i.MX 6UltraLite Applications Processor Reference Manual

Document Number: IMX6ULRM
Rev. 1, 04/2016

Contents

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

Chapter 1
Introduction

1.1 About This Document....147

1.1.1 Audience....147

1.1.2 Organization....148

1.1.3 Suggested Reading....148

1.1.3.1 General Information....148

1.1.3.2 Related Documentation....148

1.1.4 Conventions....148

1.1.5 Register Access....150

1.1.5.1 Register Diagram Field Access Type Legend....150

1.1.5.2 Register Macro Usage....151

1.1.6 Signal Conventions....152

1.1.7 Acronyms and Abbreviations....153

1.2 Introduction....155

1.3 Target Applications....156

1.4 Features....156

1.5 Architectural Overview....159

1.5.1 Simplified Block Diagram....159

1.5.2 Architectural Partitioning....160

1.5.3 Endianness Support....162

1.5.4 Memory Interfaces....162

Chapter 2
Memory Maps

2.1 Memory system overview....163

2.2 ARM Platform Memory Map....163

2.3 DMA memory map....168

Chapter 3

Section number Title Page

Interrupts and DMA Events

3.1 Overview....169
3.2 Cortex A7 interrupts....169
3.3 SDMA event mapping....173

Chapter 4
External Signals and Pin Multiplexing

4.1 Overview....177
4.1.1 Muxing Options....177

Chapter 5
Fusemap

5.1 Boot Fusemap....201
5.2 Lock Fusemap....212
5.3 Fusemap Descriptions Table....212

Chapter 6
External Memory Controllers

6.1 Overview....221
6.2 Multi-mode DDR controller (MMDC) overview and feature summary....221
6.3 EIM-PSRAM/NOR Flash controller overview....222
6.3.1 EIM features....222
6.3.2 EIM boot scenarios....223
6.3.3 EIM boot configuration....223
6.3.4 OneNAND requirements....224

Chapter 7
System Debug

7.1 Overview....225
7.2 Chip and ARM Platform Debug Architecture....225
7.2.1 Debug Features....226
7.2.2 Debug System components....226
7.2.2.1 AMBA trace bus (ATB)....227
7.2.2.2 ATB replicator....227

7.2.2.3 Embedded Cross Triggering ... 227
7.2.2.3.1 Cross-Trigger Matrix (CTM) ... 228
7.2.2.3.2 Cross-Trigger Interface (CTI) ... 229
7.2.2.4 Debug Access Port (DAP) ... 229
7.2.3 Chip-Specific SJC Features ... 230
7.2.3.1 JTAG Disable Mode ... 230
7.2.3.2 JTAG ID ... 230
7.2.4 System JTAG Controller - SJC ... 230
7.2.5 System JTAG controller main features ... 231
7.2.6 SJC TAP Port ... 231
7.2.7 SJC main blocks ... 231
7.3 Smart DMA (SDMA) core ... 232
7.3.1 SDMA On Chip Emulation Module (OnCE) Feature Summary ... 232
7.3.1.1 Other SDMA Debug Functionality ... 233
7.3.1.2 SDMA ROM Patching ... 234
7.4 Miscellaneous ... 234
7.4.1 Clock/Reset/Power ... 234
7.5 Supported tools ... 234

Chapter 8
System Boot

8.1 Overview ... 237
8.2 Boot modes ... 238
8.2.1 Boot mode pin settings ... 239
8.2.2 High level boot sequence ... 239
8.2.3 Boot From Fuses Mode (BOOT_MODE[1:0] = 00b) ... 240
8.2.4 Serial Downloader ... 241
8.2.5 Internal Boot Mode (BOOT_MODE[1:0] = 0b10) ... 243
8.2.6 Boot security settings ... 243
8.3 Device Configuration ... 244

8.3.1 Boot eFUSE Descriptions ... 244
8.3.2 GPIO Boot Overrides ... 246
8.3.3 Device Configuration Data (DCD) ... 247
8.4 Device Initialization ... 248
8.4.1 Internal ROM /RAM memory map ... 248
8.4.2 Boot Block Activation ... 249
8.4.3 Clocks at Boot Time ... 249
8.4.4 Enabling MMU and Caches ... 251
8.4.5 Exception Handling ... 252
8.4.6 Interrupt Handling During Boot ... 252
8.4.7 Persistent Bits ... 253
8.5 Boot Devices (Internal Boot) ... 253
8.5.1 NOR Flash/OneNAND using EIM Interface ... 254
8.5.1.1 NOR Flash Boot Operation ... 255
8.5.1.2 OneNAND Flash Boot Operation ... 255
8.5.1.3 IOMUX Configuration for EIM Devices ... 256
8.5.2 NAND Flash ... 257
8.5.2.1 NAND eFUSE Configuration ... 257
8.5.2.2 NAND Flash Boot Flow and Boot Control Blocks (BCB) ... 259
8.5.2.3 Firmware Configuration Block ... 263
8.5.2.4 Discovered Bad Block Table (DBBT) ... 266
8.5.2.5 Bad Block Handling in the ROM ... 266
8.5.2.6 Read Retry Handling in the ROM ... 267
8.5.2.7 Toggle Mode DDR NAND Boot ... 269
8.5.2.7.1 GPMI and BCH Clocks Configuration ... 269
8.5.2.7.2 Setup DMA for DDR Transfers ... 270
8.5.2.7.3 Reconfigure Timing and Speed Using Values in FCB ... 270
8.5.2.8 Typical NAND Page Organization ... 271
8.5.2.8.1 BCH ECC Page Organization ... 271

8.5.2.8.2 Metadata....272
8.5.2.9 IOMUX Configuration for NAND....272
8.5.3 Expansion Device....273
8.5.3.1 Expansion Device eFUSE Configuration....273
8.5.3.2 MMC and eMMC Boot....276
8.5.3.3 SD, eSD and SDXC....285
8.5.3.4 IOMUX Configuration for SD/MMC....285
8.5.3.5 Redundant Boot Support for Expansion Device....286
8.5.4 Serial ROM through SPI....288
8.5.4.1 Serial ROM eFUSE Configuration....288
8.5.4.2 ECSPI Boot....289
8.5.4.2.1 ECSPI IOMUX Pin Configuration....290
8.6 QuadSPI Serial Flash Memory Boot....291
8.6.1 QuadSPI eFUSE Configuration....291
8.6.2 QuadSPI Serial Flash BOOT Operation....291
8.6.3 QuadSPI Configuration Parameters....292
8.6.4 IOMUX Configuration for QSPI Devices....295
8.6.5 QuadSPI boot flow chart....296
8.7 Program image....297
8.7.1 Image Vector Table and Boot Data....297
8.7.1.1 Image Vector Table Structure....298
8.7.1.2 Boot Data Structure....299
8.7.2 Device Configuration Data (DCD)....299
8.7.2.1 Write Data Command....300
8.7.2.2 Check Data Command....302
8.7.2.3 NOP Command....303
8.7.2.4 Unlock Command....304
8.8 Plugin Image....304
8.9 Serial Downloader....305

8.9.1 USB....308
8.9.1.1 USB Configuration Details....308
8.9.1.2 IOMUX Configuration for USB....309
8.9.2 UART....309
8.9.2.1 UART Configuration Details....309
8.9.2.2 UART eFUSE Configuration....309
8.9.2.3 IOMUX Configuration for UART....310
8.9.3 Serial Download protocol....310
8.9.3.1 SDP Command....311
8.9.3.1.1 READ REGISTER....311
8.9.3.1.2 WRITE REGISTER....312
8.9.3.1.3 WRITE_FILE....313
8.9.3.1.4 ERROR_STATUS....314
8.9.3.1.5 DCD WRITE....314
8.9.3.1.6 SKIP_DCD_HEADER....315
8.9.3.1.7 JUMP ADDRESS....316
8.10 Recovery Devices....317
8.11 USB Low Power Boot....317
8.12 SD/MMC Manufacture Mode....319
8.13 High Assurance Boot (HAB)....320
8.13.1 HAB API Vector Table Addresses....322

Chapter 9
Multimedia

9.1 Display and graphics subsystem....323
9.1.1 PiXel Processing Pipeline....324
9.1.2 LCD Interface....324
9.1.3 Parallel CMOS Sensor Interface....324
9.2 Audio subsystem....325
9.2.1 Audio Subsystem Module Overview....326

9.2.2 Medium Quality Sound (MQS)....327
9.2.3 Synchronous Audio Interface (SAI)....327
9.2.4 Sony/Philips Digital Interface (SPDIF)....327
9.2.5 Asynchronous Sample Rate Converter (ASRC)....328

Chapter 10
Clock and Power Management

10.1 Introduction....331
10.2 Device Power Management Architecture Components....331
10.2.1 Centralized components of clock generation and management....332
10.2.2 Centralized components of power generation, distribution and management....333
10.2.3 Reset generation and distribution system....333
10.2.4 Power and clock management framework....333
10.3 Clock Management....334
10.3.1 Centralized components of clock management system....334
10.3.2 Clock generation....336
10.3.2.1 Crystal Oscillator (XTALOSC)....337
10.3.2.2 Low Voltage Differential Signaling (LVDS) I/O ports....337
10.3.2.3 PLLs....337
10.3.2.3.1 General PLL Control and Status Functions....338
10.3.2.4 CCM....340
10.3.2.5 Low Power Clock Gating unit (LPCG)....340
10.3.3 Peripheral components of clock management system....341
10.3.3.1 Interface and functional clock....341
10.3.3.2 Block level clock management....342
10.3.3.2.1 Master clock protocol....342
10.3.3.2.2 Slave clock protocol....343
10.3.3.3 Clock Domain(s)....343
10.3.3.4 Domain level clock management....343
10.3.3.5 Domain dependencies....343

10.4 Power management....344
10.4.1 Centralized Components of Power Management System....344
10.4.1.1 Integrated PMU....345
10.4.1.1.1 Digital LDO Regulators....347
10.4.1.1.2 Analog LDO regulators....348
10.4.1.1.3 USB LDO....348
10.4.1.1.4 SNVS regulator....348
10.4.1.2 GPC - General Power Controller....349
10.4.1.3 SRC - System reset Controller....349
10.4.1.4 Power domain(s)....350
10.4.1.4.1 Power distribution....351
10.4.1.4.2 Domain Memory and domain logic state retention in case of Power Gating. 352
10.4.1.4.3 Power Gating Domain Management....352
10.4.1.4.3.1 Cortex-A7 Core Platform....353
10.4.1.4.3.2 SOC_PD....353
10.4.1.4.3.3 SoC....354
10.4.1.4.4 Power Gating domain dependencies....354
10.4.1.5 Voltage domains....355
10.4.1.6 Voltage domain management....355
10.4.1.6.1 Dynamic....355
10.4.1.6.1.1 DVFS....355
10.4.1.6.1.2 Voltage Scaling....357
10.4.1.6.2 Static....357
10.4.1.6.2.1 Standby Leakage reduction (SLR)....357
10.4.1.6.2.2 ANALOG PHYs IPs -....357
10.4.1.6.3 Voltage domain dependencies....357
10.4.1.6.4 IO voltage....358
10.4.1.7 System domains layout....358
10.4.2 Power management techniques....360

10.4.2.1 Power saving techniques....361
10.4.2.2 Thermal-aware power management....361
10.4.2.3 Peripheral Power management....362
10.4.2.3.1 Main memory power management....362
10.4.2.3.2 Video-Graphics system power management....363
10.4.2.3.3 IO power reduction....363
10.4.3 Examples of External Power Supply Interface....364
10.5 ONOFF (Button)....366

Chapter 11
System Security

11.1 Overview....369
11.2 Central Security Unit (CSU)....370
11.2.1 CSU Overview....370
11.2.2 CSU Features....370
11.2.3 CSU Functional Description....371
11.2.3.1 CSU Peripheral Access Policy....371
11.3 Cryptographic Acceleration and Assurance Module (CAAM)....372
11.3.1 CAAM Overview....372
11.4 Secure Non-Volatile Storage (SNVS)....373
11.4.1 SNVS Overview....373
11.4.2 Tamper Detection....373
11.5 High Assurance Boot (HAB)....374
11.6 System JTAG Controller (SJC)....375
11.7 Bus Encryption Engine (BEE)....375

Chapter 12
ARM Cortex A7 Platform (CA7)

12.1 Overview....377
12.2 External Signals....378
12.3 Clocks....378

12.4 Platform Configuration..... 378
12.5 Low-Power and Performance..... 379

Chapter 13
Analog-to-Digital Converter (ADC)

13.1 Overview.....381
13.1.1 Features..... 381
13.1.2 ADC I/F block diagram.....382
13.1.3 ADC block diagram..... 382
13.1.4 ADC module interface..... 383
13.1.5 Modes of Operation..... 383
13.2 External Signals..... 383
13.3 Functional Description.....384
13.3.1 Clock Select and Divide Control..... 385
13.3.2 Voltage Reference Selection 386
13.3.3 Hardware Triggering and Channel Selection.....387
13.3.4 Conversion Control..... 388
13.3.4.1 Initiating Conversions..... 388
13.3.4.2 Completing Conversions.....389
13.3.4.3 Aborting Conversions..... 390
13.3.4.4 Power Control..... 391
13.3.4.5 Sample Time and Total Conversion Time..... 391
13.3.4.6 Conversion Time Examples..... 394
13.3.4.6.1 Typical conversion time configuration..... 394
13.3.4.6.2 Long conversion time configuration.....394
13.3.4.6.3 Short conversion time configuration.....395
13.3.4.7 Hardware Average Function..... 395
13.3.5 Automatic Compare Function.....396
13.3.6 Calibration Function..... 397
13.3.7 User Defined Offset Function 398

13.3.8 Temperature Sensor ... 399
13.3.9 MCU Wait Mode Operation ... 399
13.3.10 MCU Stop Mode Operation ... 400
13.3.10.1 Stop Mode With ADACK Disabled ... 400
13.3.10.2 Stop Mode With ADACK Enabled ... 400
13.4 Initialization Information ... 401
13.4.1 ADC Module Initialization Example ... 401
13.4.1.1 Initialization Sequence ... 401
13.4.1.2 Pseudo-Code Example ... 401
13.5 Application Information ... 403
13.5.1 Sources of Error ... 403
13.5.1.1 Sampling Error ... 403
13.5.1.2 Pin Leakage Error ... 403
13.5.1.3 Noise-Induced Errors ... 403
13.5.1.4 Code Width and Quantization Error ... 404
13.5.1.5 Linearity Errors ... 405
13.5.1.6 Code Jitter, Non-Monotonicity, and Missing Codes ... 406
13.6 Memory map and register definition ... 406
13.6.1 Control register for hardware triggers (ADC*x*_HC0) ... 408
13.6.2 Control register for hardware triggers (ADC*x*_HC1) ... 409
13.6.3 Status register for HW triggers (ADC*x*_HS) ... 411
13.6.4 Data result register for HW triggers (ADC*x*_R0) ... 412
13.6.5 Data result register for HW triggers (ADC*x*_R1) ... 413
13.6.6 Configuration register (ADC*x*_CFG) ... 414
13.6.7 General control register (ADC*x*_GC) ... 416
13.6.8 General status register (ADC*x*_GS) ... 418
13.6.9 Compare value register (ADC*x*_CV) ... 419
13.6.10 Offset correction value register (ADC*x*_OFS) ... 420
13.6.11 Calibration value register (ADC*x*_CAL) ... 421

Chapter 14
AHB to IP Bridge (AIPSTZ)

14.1 Overview....423
14.1.1 Features....423
14.2 Clocks....423
14.3 Functional Description....424
14.4 Access Protections....425
14.5 Access Support....425
14.6 Initialization Information....426
14.6.1 Security Block....426
14.7 AIPSTZ Memory Map/Register Definition....427
14.7.1 Master Priviledge Registers (AIPSTZ_MPR)....428
14.7.2 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR)....431
14.7.3 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR1)....434
14.7.4 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR2)....437
14.7.5 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR3)....440
14.7.6 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR4)....443

Chapter 15
AHB-to-APBH Bridge with DMA (APBH-Bridge-DMA)

15.1 Overview....445
15.2 Clocks....446
15.3 APBH DMA....447
15.4 NAND Read Status Polling Example....452
15.5 APBH Memory Map/Register Definition....454
15.5.1 AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0*n*)....460
15.5.2 AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1*n*)....461
15.5.3 AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2*n*)....465
15.5.4 AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL*n*)....470
15.5.5 AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)....471

15.5.6 AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)....472
15.5.7 AHB to APBH DMA Debug Register (APBH_DEBUG)....473
15.5.8 APBH DMA Channel n Current Command Address Register (APBH_CH*n*_CURCMDAR)....474
15.5.9 APBH DMA Channel n Next Command Address Register (APBH_CH*n*_NXTCMDAR)....475
15.5.10 APBH DMA Channel n Command Register (APBH_CH*n*_CMD)....475
15.5.11 APBH DMA Channel n Buffer Address Register (APBH_CH*n*_BAR)....477
15.5.12 APBH DMA Channel n Semaphore Register (APBH_CH*n*_SEMA)....478
15.5.13 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG1)....479
15.5.14 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG2)....482
15.5.15 APBH Bridge Version Register (APBH_VERSION)....483

Chapter 16
Asynchronous Sample Rate Converter (ASRC)

16.1 Overview....485
16.1.1 Features....487
16.1.2 Modes of Operation....488
16.1.2.1 Data Transfer Schemes....488
16.1.2.1.1 Data Input Modes....488
16.1.2.1.2 Data Output Modes....489
16.1.2.2 Word Alignment Supported....490
16.1.2.2.1 Input Data Alignment Modes....491
16.1.2.2.2 Output Data Alignment Modes....491
16.2 Clocks....492
16.3 Interrupts....492
16.4 DMA requests....493
16.5 Functional Description....493
16.5.1 Algorithm Description....493
16.5.1.1 Signal processing flow....493
16.5.1.2 Operation of the Filter....497
16.5.1.2.1 Support of Physical Clocks....497

16.6 Startup Procedure....497
16.7 ASRC Memory Map/Register Definition....502
16.7.1 ASRC Control Register (ASRC_ASRCTR)....504
16.7.2 ASRC Interrupt Enable Register (ASRC_ASRIER)....507
16.7.3 ASRC Channel Number Configuration Register (ASRC_ASRCNCR)....508
16.7.4 ASRC Filter Configuration Status Register (ASRC_ASRCFG)....510
16.7.5 ASRC Clock Source Register (ASRC_ASRCSR)....512
16.7.6 ASRC Clock Divider Register 1 (ASRC_ASRCDR1)....515
16.7.7 ASRC Clock Divider Register 2 (ASRC_ASRCDR2)....516
16.7.8 ASRC Status Register (ASRC_ASRSTR)....518
16.7.9 ASRC Parameter Register n (ASRC_ASRPMn*n*)....521
16.7.10 ASRC ASRC Task Queue FIFO Register 1 (ASRC_ASRTFR1)....522
16.7.11 ASRC Channel Counter Register (ASRC_ASRCCR)....523
16.7.12 ASRC Data Input Register for Pair *x* (ASRC_ASRDI*n*)....524
16.7.13 ASRC Data Output Register for Pair *x* (ASRC_ASRDO*n*)....524
16.7.14 ASRC Ideal Ratio for Pair A-High Part (ASRC_ASRIDRHA)....525
16.7.15 ASRC Ideal Ratio for Pair A -Low Part (ASRC_ASRIDRLA)....526
16.7.16 ASRC Ideal Ratio for Pair B-High Part (ASRC_ASRIDRHB)....526
16.7.17 ASRC Ideal Ratio for Pair B-Low Part (ASRC_ASRIDRLB)....527
16.7.18 ASRC Ideal Ratio for Pair C-High Part (ASRC_ASRIDRHC)....527
16.7.19 ASRC Ideal Ratio for Pair C-Low Part (ASRC_ASRIDRLC)....528
16.7.20 ASRC 76 kHz Period in terms of ASRC processing clock (ASRC_ASR76K)....529
16.7.21 ASRC 56 kHz Period in terms of ASRC processing clock (ASRC_ASR56K)....530
16.7.22 ASRC Misc Control Register for Pair A (ASRC_ASRMCRA)....531
16.7.23 ASRC FIFO Status Register for Pair A (ASRC_ASRFSTA)....533
16.7.24 ASRC Misc Control Register for Pair B (ASRC_ASRMCRB)....534
16.7.25 ASRC FIFO Status Register for Pair B (ASRC_ASRFSTB)....536
16.7.26 ASRC Misc Control Register for Pair C (ASRC_ASRMCRC)....537
16.7.27 ASRC FIFO Status Register for Pair C (ASRC_ASRFSTC)....539

16.7.28 ASRC Misc Control Register 1 for Pair X (ASRC_ASRMCR1*n*)....540

Chapter 17
40-BIT Correcting ECC Accelerator (BCH)

17.1 Overview....543

17.2 Operation....545

17.2.1 BCH Limitations and Assumptions....546

17.2.2 Flash Page Layout....547

17.2.3 Determining the ECC layout for a device....549

17.2.3.1 4K+218 flash, 10 bytes metadata, 512 byte data blocks, separate metadata, Assuming GF(213)....549

17.2.3.2 4K+128 flash, 10 bytes metadata, 1024 byte data blocks, separate metadata, assuming GF(213) for data and GF(214) for metadata....550

17.2.4 Data Buffers in System Memory....550

17.3 Memory to Memory (Loopback) Operation....553

17.4 Programming the BCH/GPMI Interfaces....554

17.4.1 BCH Encoding for NAND Writes....555

17.4.1.1 DMA Structure Code Example....557

17.4.1.2 Using the BCH Encoder....562

17.4.2 BCH Decoding for NAND Reads....563

17.4.2.1 DMA Structure Code Example....567

17.4.2.2 Using the Decoder....570

17.4.3 Interrupts....572

17.5 Behavior During Reset....573

17.6 BCH Memory Map/Register Definition....574

17.6.1 Hardware BCH ECC Accelerator Control Register (BCH_CTRL*n*)....578

17.6.2 Hardware ECC Accelerator Status Register 0 (BCH_STATUS0*n*)....580

17.6.3 Hardware ECC Accelerator Mode Register (BCH_MODE*n*)....582

17.6.4 Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR*n*)....583

17.6.5 Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR*n*)....583

17.6.6 Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR*n*)....584

17.6.7 Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT*n*)..... 584
17.6.8 Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0*n*).....585
17.6.9 Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1*n*).....587
17.6.10 Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0*n*).....588
17.6.11 Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1*n*).....590
17.6.12 Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0*n*).....591
17.6.13 Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1*n*).....593
17.6.14 Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0*n*).....594
17.6.15 Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1*n*).....596
17.6.16 Hardware BCH ECC Debug Register0 (BCH_DEBUG0*n*)..... 597
17.6.17 KES Debug Read Register (BCH_DBGKESREAD*n*)..... 599
17.6.18 Chien Search Debug Read Register (BCH_DBGCSFEREAD*n*)..... 600
17.6.19 Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD*n*).....600
17.6.20 Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD*n*).....601
17.6.21 Block Name Register (BCH_BLOCKNAME*n*).....601
17.6.22 BCH Version Register (BCH_VERSION*n*)..... 602
17.6.23 Hardware BCH ECC Debug Register 1 (BCH_DEBUG1*n*)..... 603

Chapter 18
Clock Controller Module (CCM)

18.1 Overview.....605
18.1.1 Features..... 605
18.1.2 CCM Block Diagram..... 606
18.2 External Signals..... 608
18.3 CCM Clock Tree.....608
18.4 System Clocks.....610
18.5 Functional Description.....623
18.5.1 Clock Generation..... 623
18.5.1.1 External Low Frequency Clock - CKIL 623
18.5.1.1.1 CKIL synchronizing to IPG_CLK.....623

18.5.1.2 External High Frequency Clock - CKIH and internal oscillator....623
18.5.1.3 PLL reference clock....623
18.5.1.3.1 ARM PLL....624
18.5.1.3.2 USB PLLs....624
18.5.1.3.3 System PLL....624
18.5.1.3.4 Audio / Video PLL....625
18.5.1.3.5 Ethernet PLL....625
18.5.1.4 Phase Fractional Dividers (PFD)....626
18.5.1.5 CCM internal clock generation....626
18.5.1.5.1 Clock Switcher....626
18.5.1.5.2 PLL bypass procedure....628
18.5.1.5.3 PLL clock change....628
18.5.1.5.4 Clock Root Generator....628
18.5.1.5.5 Initial values controlled by the System JTAG Controller (SJC)....630
18.5.1.5.6 Divider change handshake....630
18.5.1.6 Disabling / Enabling PLLs....630
18.5.1.7 Clock Switching Multiplexers....631
18.5.1.8 Low Power Clock Gating module (LPCG)....632
18.5.1.8.1 MMDC handshake....633
18.5.2 DVFS support....634
18.5.3 Power modes....634
18.5.3.1 RUN mode....634
18.5.3.2 WAIT mode....634
18.5.3.2.1 Entering WAIT mode....634
18.5.3.2.2 Exiting WAIT mode....635
18.5.3.3 STOP mode....635
18.5.3.3.1 Entering STOP mode....635
18.5.3.3.2 Exiting STOP mode....636
18.6 CCM Memory Map/Register Definition....637

18.6.1 CCM Control Register (CCM_CCR)....639
18.6.2 CCM Control Divider Register (CCM_CCDR)....640
18.6.3 CCM Status Register (CCM_CSR)....642
18.6.4 CCM Clock Switcher Register (CCM_CCSR)....643
18.6.5 CCM Arm Clock Root Register (CCM_CACRR)....644
18.6.6 CCM Bus Clock Divider Register (CCM_CBCDR)....645
18.6.7 CCM Bus Clock Multiplexer Register (CCM_CBCMR)....648
18.6.8 CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)....650
18.6.9 CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)....652
18.6.10 CCM Serial Clock Divider Register 1 (CCM_CSCDR1)....654
18.6.11 CCM SAI1 Clock Divider Register (CCM_CS1CDR)....656
18.6.12 CCM SAI2 Clock Divider Register (CCM_CS2CDR)....657
18.6.13 CCM D1 Clock Divider Register (CCM_CDCDR)....659
18.6.14 CCM HSC Clock Divider Register (CCM_CHSCCDR)....660
18.6.15 CCM Serial Clock Divider Register 2 (CCM_CSCDR2)....662
18.6.16 CCM Serial Clock Divider Register 3 (CCM_CSCDR3)....663
18.6.17 CCM Wakeup Detector Register (CCM_CWDR)....664
18.6.17 CCM Divider Handshake In-Process Register (CCM_CDHIPR)....665
18.6.18 CCM Low Power Control Register (CCM_CLPCR)....668
18.6.19 CCM Interrupt Status Register (CCM_CISR)....670
18.6.20 CCM Interrupt Mask Register (CCM_CIMR)....673
18.6.21 CCM Clock Output Source Register (CCM_CCOSR)....675
18.6.22 CCM General Purpose Register (CCM_CGPR)....677
18.6.23 CCM Clock Gating Register 0 (CCM_CCGR0)....678
18.6.24 CCM Clock Gating Register 1 (CCM_CCGR1)....680
18.6.25 CCM Clock Gating Register 2 (CCM_CCGR2)....682
18.6.26 CCM Clock Gating Register 3 (CCM_CCGR3)....683
18.6.27 CCM Clock Gating Register 4 (CCM_CCGR4)....684
18.6.28 CCM Clock Gating Register 5 (CCM_CCGR5)....686

18.6.29 CCM Clock Gating Register 6 (CCM_CCGR6)....687
18.6.30 CCM Module Enable Overide Register (CCM_CMEOR)....689
18.7 CCM Analog Memory Map/Register Definition....690
18.7.1 Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM*n*)....694
18.7.2 Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1*n*)....696
18.7.3 Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2*n*)....698
18.7.4 Analog System PLL Control Register (CCM_ANALOG_PLL_SYS*n*)....700
18.7.5 528MHz System PLL Spread Spectrum Register (CCM_ANALOG_PLL_SYS_SS)....702
18.7.6 Numerator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_NUM)....702
18.7.7 Denominator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_DENOM)....703
18.7.8 Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO*n*)....704
18.7.9 Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)....706
18.7.10 Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)....707
18.7.11 Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO*n*)....708
18.7.12 Numerator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_NUM)....710
18.7.13 Denominator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_DENOM)....711
18.7.14 Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET*n*)....712
18.7.15 480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480*n*)....714
18.7.16 528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528*n*)....716
18.7.17 Miscellaneous Register 0 (CCM_ANALOG_MISC0*n*)....719
18.7.18 Miscellaneous Register 1 (CCM_ANALOG_MISC1*n*)....723
18.7.19 Miscellaneous Register 2 (CCM_ANALOG_MISC2*n*)....726

Chapter 19
CMOS Sensor Interface (CSI)

19.1 Overview....731
19.2 External Signals....732

19.3 Clocks ... 734
19.4 Principles of Operation ... 734
19.4.1 Data Transfer with the Embedded DMA Controllers ... 736
19.4.2 Gated Clock Mode ... 737
19.4.3 Non-Gated Clock Mode ... 737
19.4.4 CCIR656 Interlace Mode ... 738
19.4.5 CCIR656 Progressive Mode ... 741
19.4.6 Error Correction for CCIR656 Coding ... 741
19.5 Interrupt Generation ... 742
19.5.1 Start Of Frame Interrupt (SOF_INT) ... 742
19.5.2 End Of Frame Interrupt (EOF_INT) ... 742
19.5.3 Change Of Field Interrupt (COF_INT) ... 742
19.5.4 CCIR Error Interrupt (ECC_INT) ... 743
19.5.5 RxFIFO Full Interrupt (RxFF_INT) ... 743
19.5.6 Statistic FIFO Full Interrupt (STATFF_INT) ... 743
19.5.7 RxFIFO Overrun Interrupt (RFF_OR_INT) ... 743
19.5.8 Statistic FIFO Overrun Interrupt (SFF_OR_INT) ... 743
19.5.9 Frame Buffer1 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB1) ... 744
19.5.10 Frame Buffer2 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB2) ... 744
19.5.11 Statistic FIFO DMA Transfer Done Interrupt (DMA_TSF_DONE_SFF) ... 744
19.5.12 AHB Bus Response Error Interrupt (HRESP_ERR_INT) ... 744
19.5.13 DMA Field 0 Transfer Done Interrupt (DMA_FIELD0_DONE) ... 744
19.5.14 DMA Field 1 Transfer Done Interrupt (DMA_FIELD1_DONE) ... 744
19.5.15 Base Address Change Error Interrupt (BASEADDR_CHANGE_ERROR) ... 745
19.6 Data Packing Style ... 745
19.6.1 RX FIFO Path ... 746
19.6.1.1 Bayer Data ... 746
19.6.1.2 RGB565 Data ... 746
19.6.1.3 RGB888 Data ... 747

19.6.2 STAT FIFO Path....749
19.7 CSI Memory Map/Register Definition....749
19.7.1 CSI Control Register 1 (CSI_CSICR1)....751
19.7.2 CSI Control Register 2 (CSI_CSICR2)....755
19.7.3 CSI Control Register 3 (CSI_CSICR3)....757
19.7.4 CSI Statistic FIFO Register (CSI_CSISTATFIFO)....759
19.7.5 CSI RX FIFO Register (CSI_CSIRFIFO)....760
19.7.6 CSI RX Count Register (CSI_CSIRXCNT)....760
19.7.7 CSI Status Register (CSI_CSISR)....761
19.7.8 CSI DMA Start Address Register - for STATFIFO (CSI_CSIDMASA_STATFIFO)....764
19.7.9 CSI DMA Transfer Size Register - for STATFIFO (CSI_CSIDMATS_STATFIFO)....764
19.7.10 CSI DMA Start Address Register - for Frame Buffer1 (CSI_CSIDMASA_FB1)....765
19.7.11 CSI DMA Transfer Size Register - for Frame Buffer2 (CSI_CSIDMASA_FB2)....766
19.7.12 CSI Frame Buffer Parameter Register (CSI_CSIFBUF_PARA)....766
19.7.13 CSI Image Parameter Register (CSI_CSIIMAG_PARA)....767
19.7.14 CSI Control Register 18 (CSI_CSICR18)....768
19.7.15 CSI Control Register 19 (CSI_CSICR19)....770

Chapter 20
Enhanced Configurable SPI (ECSPI)

20.1 Overview....771
20.1.1 Features....772
20.1.2 Modes and Operations....772
20.2 External Signals....773
20.3 Clocks....774
20.4 Functional Description....774
20.4.1 Master Mode....775
20.4.2 Slave Mode....775
20.4.3 Low Power Modes....775
20.4.4 Operations....775

20.4.4.1 Typical Master Mode....775
20.4.4.1.1 Master Mode with SPI_RDY....776
20.4.4.1.2 Master Mode with Wait States....778
20.4.4.1.3 Master Mode with SS_CTL[3:0] Control....778
20.4.4.1.4 Master Mode with Phase Control....779
20.4.4.2 Typical Slave Mode....780
20.4.5 Reset....781
20.4.6 Interrupts....781
20.4.7 DMA....782
20.4.8 Byte Order....783
20.5 Initialization....784
20.6 Applications....784
20.7 ECSPI Memory Map/Register Definition....785
20.7.1 Receive Data Register (ECSPI*x*_RXDATA)....787
20.7.2 Transmit Data Register (ECSPI*x*_TXDATA)....788
20.7.3 Control Register (ECSPI*x*_CONREG)....788
20.7.4 Config Register (ECSPI*x*_CONFIGREG)....791
20.7.5 Interrupt Control Register (ECSPI*x*_INTREG)....793
20.7.6 DMA Control Register (ECSPI*x*_DMAREG)....794
20.7.7 Status Register (ECSPI*x*_STATREG)....796
20.7.8 Sample Period Control Register (ECSPI*x*_PERIODREG)....797
20.7.9 Test Control Register (ECSPI*x*_TESTREG)....799
20.7.10 Message Data Register (ECSPI*x*_MSGDATA)....800

Chapter 21
External Interface Module (EIM)

21.1 Overview....801
21.1.1 Features....803
21.1.2 Modes of Operation....803
21.1.2.1 Asynchronous Mode....804

21.1.2.2 Asynchronous Page Read Mode 804
21.1.2.3 Multiplexed Address/Data Mode 804
21.1.2.4 Burst Clock Mode 805
21.1.2.5 Low Power Modes 805
21.1.2.6 Boot Mode 806
21.2 External Signals 806
21.2.1 Other Important Block I/O Signals Internal to the SoC 806
21.3 Clocks 806
21.4 Chip Select Memory Map 807
21.5 Functional Description 807
21.5.1 Bus Sizing Configuration 808
21.5.1.1 8 BIT PORT SUPPORT 808
21.5.1.1.1 MOTOROLA 68000 808
21.5.1.1.2 INTEL 386 808
21.5.2 EIM Operational Modes 809
21.5.3 Burst Mode (Synchronous) Memory Operation 809
21.5.4 Burst Clock Divisor (BCD) 810
21.5.5 Burst Clock Start (BCS) 811
21.5.6 Multiplexed Address/Data Mode Support 811
21.5.7 Mixed Master/Memory Burst Modes Support 811
21.5.8 AXI (Master) Bus Cycles Support 812
21.5.9 WAIT_B Signal, RWSC and WWSC bit fields Usage 814
21.5.10 IPS Register Interface 814
21.5.11 MRS Set for PSRAM 815
21.5.12 EIM Access Termination 815
21.5.13 Error Conditions 815
21.5.14 DTACK Mode 816
21.5.15 EIM_GRANT / EIM_BUSY Handshake Description 816
21.5.16 LPMD / LPACK Handshake Description 817

21.5.17 Endianness ... 817
21.5.18 Strobe Signal Use ... 818
21.6 Initialization Information ... 819
21.6.1 Booting from EIM ... 819
21.7 Typical Application ... 819
21.7.1 Access to Intel Sibley Flash ... 820
21.7.1.1 Intel Sibley Flash Asynchronous Mode Configuration ... 820
21.7.1.2 Intel Sibley Flash Synchronous Mode Configuration ... 820
21.7.1.3 Intel Sibley Flash Utility ... 820
21.7.2 Access to MDOC Device ... 821
21.7.2.1 MDOC Device Boot ... 821
21.7.2.2 MDOC Device Asynchronous Mode Configuration ... 821
21.7.2.3 MDOC Device Utility ... 821
21.7.3 Access to Micron PSRAM ... 821
21.7.3.1 Micron PSRAM Asynchronous Mode Configuration ... 822
21.7.3.2 Micron PSRAM Synchronous Mode Configuration ... 822
21.7.4 Access to Samsung OneNAND ... 822
21.7.4.1 Samsung OneNAND Boot ... 822
21.7.4.2 Samsung OneNAND Asynchronous Mode Configuration ... 823
21.7.4.3 Samsung OneNAND Synchronous Mode Configuration ... 823
21.7.4.4 Samsung OneNAND Utility ... 823
21.7.5 Access to Samsung UtRAM ... 824
21.7.5.1 Samsung UtRAM Asynchronous Mode Configuration ... 824
21.7.5.2 Samsung UtRAM Synchronous Mode Configuration ... 824
21.7.6 Access to Spansion Flash ... 824
21.7.6.1 Spansion Flash Asynchronous Mode Configuration ... 824
21.7.6.2 Spansion Flash Synchronous Mode Configuration ... 825
21.7.6.3 Spansion Flash Utility ... 825
21.7.7 8 bit support ... 826

21.8 External Bus Timing Diagrams ... 827
21.8.1 Asynchronous Read Memory Accesses Timing Diagram ... 827
21.8.2 Asynchronous Write Memory Accesses Timing Diagram ... 828
21.8.3 Asynchronous Read/Write Memory Accesses Timing Diagram ... 829
21.8.4 Asynchronous Read/Write Using RAL, WAL and CSREC ... 831
21.8.5 Consecutive Asynchronous Write Memory Accesses Timing Diagram ... 832
21.8.6 Consecutive Asynchronous Read Memory Accesses Timing Diagram ... 835
21.8.7 Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=0 ... 837
21.8.8 Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=1 ... 838
21.8.9 Burst (Synchronous Mode) Write Memory Access Timing - BCD=1 ... 839
21.8.10 Asynchronous Page Mode Access ... 841
21.8.11 DTACK Mode - AXI Single Access ... 841
21.8.12 DTACK Mode - AXI Single Write Access ... 844
21.8.13 DTACK Mode - AXI Burst Access ... 845
21.9 EIM Memory Map/Register Definition ... 846
21.9.1 Chip Select n General Configuration Register 1 (EIM_CS*n*GCR1) ... 849
21.9.2 Chip Select n General Configuration Register 2 (EIM_CS*n*GCR2) ... 853
21.9.3 Chip Select n Read Configuration Register 1 (EIM_CS*n*RCR1) ... 854
21.9.4 Chip Select n Read Configuration Register 2 (EIM_CS*n*RCR2) ... 857
21.9.5 Chip Select n Write Configuration Register 1 (EIM_CS*n*WCR1) ... 858
21.9.6 Chip Select n Write Configuration Register 2 (EIM_CS*n*WCR2) ... 861
21.9.7 EIM Configuration Register (EIM_WCR) ... 862

Chapter 22
10/100-Mbps Ethernet MAC (ENET)

22.1 Introduction ... 865
22.2 Overview ... 865
22.2.1 Features ... 866
22.2.1.1 Ethernet MAC features ... 866
22.2.1.2 IP protocol performance optimization features ... 867

22.2.1.3 IEEE 1588 features ... 868
22.2.2 Block diagram ... 868
22.3 External Signals ... 869
22.4 Clocks ... 878
22.5 Memory map/register definition ... 879
22.5.1 Interrupt Event Register (ENET*x*_EIR) ... 888
22.5.2 Interrupt Mask Register (ENET*x*_EIMR) ... 891
22.5.3 Receive Descriptor Active Register (ENET*x*_RDAR) ... 894
22.5.4 Transmit Descriptor Active Register (ENET*x*_TDAR) ... 894
22.5.5 Ethernet Control Register (ENET*x*_ECR) ... 896
22.5.6 MII Management Frame Register (ENET*x*_MMFR) ... 898
22.5.7 MII Speed Control Register (ENET*x*_MSCR) ... 898
22.5.8 MIB Control Register (ENET*x*_MIBC) ... 901
22.5.9 Receive Control Register (ENET*x*_RCR) ... 902
22.5.10 Transmit Control Register (ENET*x*_TCR) ... 905
22.5.11 Physical Address Lower Register (ENET*x*_PALR) ... 907
22.5.12 Physical Address Upper Register (ENET*x*_PAUR) ... 907
22.5.13 Opcode/Pause Duration Register (ENET*x*_OPD) ... 908
22.5.14 Transmit Interrupt Coalescing Register (ENET*x*_TXIC) ... 908
22.5.15 Receive Interrupt Coalescing Register (ENET*x*_RXIC) ... 909
22.5.16 Descriptor Individual Upper Address Register (ENET*x*_IAUR) ... 910
22.5.17 Descriptor Individual Lower Address Register (ENET*x*_IALR) ... 911
22.5.18 Descriptor Group Upper Address Register (ENET*x*_GAUR) ... 911
22.5.19 Descriptor Group Lower Address Register (ENET*x*_GALR) ... 912
22.5.20 Transmit FIFO Watermark Register (ENET*x*_TFWR) ... 912
22.5.21 Receive Descriptor Ring Start Register (ENET*x*_RDSR) ... 913
22.5.22 Transmit Buffer Descriptor Ring Start Register (ENET*x*_TDSR) ... 914
22.5.23 Maximum Receive Buffer Size Register (ENET*x*_MRBR) ... 915
22.5.24 Receive FIFO Section Full Threshold (ENET*x*_RSFL) ... 916

22.5.25 Receive FIFO Section Empty Threshold (ENETx_RSEM)....916
22.5.26 Receive FIFO Almost Empty Threshold (ENETx_RAEM)....917
22.5.27 Receive FIFO Almost Full Threshold (ENETx_RAFL)....917
22.5.28 Transmit FIFO Section Empty Threshold (ENETx_TSEM)....918
22.5.29 Transmit FIFO Almost Empty Threshold (ENETx_TAEM)....918
22.5.30 Transmit FIFO Almost Full Threshold (ENETx_TAFL)....919
22.5.31 Transmit Inter-Packet Gap (ENETx_TIPG)....919
22.5.32 Frame Truncation Length (ENETx_FTRL)....920
22.5.33 Transmit Accelerator Function Configuration (ENETx_TACC)....920
22.5.34 Receive Accelerator Function Configuration (ENETx_RACC)....921
22.5.35 Reserved Statistic Register (ENETx_RMON_T_DROP)....922
22.5.36 Tx Packet Count Statistic Register (ENETx_RMON_T_PACKETS)....923
22.5.37 Tx Broadcast Packets Statistic Register (ENETx_RMON_T_BC_PKT)....923
22.5.38 Tx Multicast Packets Statistic Register (ENETx_RMON_T_MC_PKT)....924
22.5.39 Tx Packets with CRC/Align Error Statistic Register (ENETx_RMON_T_CRC_ALIGN)....924
22.5.40 Tx Packets Less Than Bytes and Good CRC Statistic Register (ENETx_RMON_T_UNDERSIZE)....924
22.5.41 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENETx_RMON_T_OVERSIZE)....925
22.5.42 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENETx_RMON_T_FRAG)....925
22.5.43 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENETx_RMON_T_JAB)....926
22.5.44 Tx Collision Count Statistic Register (ENETx_RMON_T_COL)....926
22.5.45 Tx 64-Byte Packets Statistic Register (ENETx_RMON_T_P64)....927
22.5.46 Tx 65- to 127-byte Packets Statistic Register (ENETx_RMON_T_P65TO127)....927
22.5.47 Tx 128- to 255-byte Packets Statistic Register (ENETx_RMON_T_P128TO255)....928
22.5.48 Tx 256- to 511-byte Packets Statistic Register (ENETx_RMON_T_P256TO511)....928
22.5.49 Tx 512- to 1023-byte Packets Statistic Register (ENETx_RMON_T_P512TO1023)....929
22.5.50 Tx 1024- to 2047-byte Packets Statistic Register (ENETx_RMON_T_P1024TO2047)....929
22.5.51 Tx Packets Greater Than 2048 Bytes Statistic Register (ENETx_RMON_T_P_GTE2048)....930
22.5.52 Tx Octets Statistic Register (ENETx_RMON_T_OCTETS)....930
22.5.53 Reserved Statistic Register (ENETx_IEEE_T_DROP)....930

22.5.54 Frames Transmitted OK Statistic Register (ENET*x*_IEEE_T_FRAME_OK)..... 931
22.5.55 Frames Transmitted with Single Collision Statistic Register (ENET*x*_IEEE_T_1COL).....931
22.5.56 Frames Transmitted with Multiple Collisions Statistic Register (ENET*x*_IEEE_T_MCOL)..... 932
22.5.57 Frames Transmitted after Deferral Delay Statistic Register (ENET*x*_IEEE_T_DEF).....932
22.5.58 Frames Transmitted with Late Collision Statistic Register (ENET*x*_IEEE_T_LCOL)..... 932
22.5.59 Frames Transmitted with Excessive Collisions Statistic Register (ENET*x*_IEEE_T_EXCOL).....933
22.5.60 Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET*x*_IEEE_T_MACERR)..... 933
22.5.61 Frames Transmitted with Carrier Sense Error Statistic Register (ENET*x*_IEEE_T_CSERR).....934
22.5.62 Reserved Statistic Register (ENET*x*_IEEE_T_SQE)..... 934
22.5.63 Flow Control Pause Frames Transmitted Statistic Register (ENET*x*_IEEE_T_FDXFC).....934
22.5.64 Octet Count for Frames Transmitted w/o Error Statistic Register (ENET*x*_IEEE_T_OCTETS_OK)..... 935
22.5.65 Rx Packet Count Statistic Register (ENET*x*_RMON_R_PACKETS)..... 935
22.5.66 Rx Broadcast Packets Statistic Register (ENET*x*_RMON_R_BC_PKT)..... 936
22.5.67 Rx Multicast Packets Statistic Register (ENET*x*_RMON_R_MC_PKT)..... 936
22.5.68 Rx Packets with CRC/Align Error Statistic Register (ENET*x*_RMON_R_CRC_ALIGN)..... 936
22.5.69 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET*x*_RMON_R_UNDERSIZE)..... 937
22.5.70 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET*x*_RMON_R_OVERSIZE)..... 937
22.5.71 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET*x*_RMON_R_FRAG).....938
22.5.72 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET*x*_RMON_R_JAB)..938
22.5.73 Reserved Statistic Register (ENET*x*_RMON_R_RESVD_0)..... 938
22.5.74 Rx 64-Byte Packets Statistic Register (ENET*x*_RMON_R_P64)..... 939
22.5.75 Rx 65- to 127-Byte Packets Statistic Register (ENET*x*_RMON_R_P65TO127)..... 939
22.5.76 Rx 128- to 255-Byte Packets Statistic Register (ENET*x*_RMON_R_P128TO255)..... 940
22.5.77 Rx 256- to 511-Byte Packets Statistic Register (ENET*x*_RMON_R_P256TO511)..... 940
22.5.78 Rx 512- to 1023-Byte Packets Statistic Register (ENET*x*_RMON_R_P512TO1023)..... 940
22.5.79 Rx 1024- to 2047-Byte Packets Statistic Register (ENET*x*_RMON_R_P1024TO2047)..... 941
22.5.80 Rx Packets Greater than 2048 Bytes Statistic Register (ENET*x*_RMON_R_P_GTE2048)..... 941

22.5.81 Rx Octets Statistic Register (ENET*x*_RMON_R_OCTETS)..... 942
22.5.82 Frames not Counted Correctly Statistic Register (ENET*x*_IEEE_R_DROP)..... 942
22.5.83 Frames Received OK Statistic Register (ENET*x*_IEEE_R_FRAME_OK)..... 942
22.5.84 Frames Received with CRC Error Statistic Register (ENET*x*_IEEE_R_CRC)..... 943
22.5.85 Frames Received with Alignment Error Statistic Register (ENET*x*_IEEE_R_ALIGN).....943
22.5.86 Receive FIFO Overflow Count Statistic Register (ENET*x*_IEEE_R_MACERR).....944
22.5.87 Flow Control Pause Frames Received Statistic Register (ENET*x*_IEEE_R_FDXFC)..... 944
22.5.88 Octet Count for Frames Received without Error Statistic Register (ENET*x*_IEEE_R_OCTETS_OK)... 944
22.5.89 Adjustable Timer Control Register (ENET*x*_ATCR).....945
22.5.90 Timer Value Register (ENET*x*_ATVR)..... 947
22.5.91 Timer Offset Register (ENET*x*_ATOFF)..... 947
22.5.92 Timer Period Register (ENET*x*_ATPER).....947
22.5.93 Timer Correction Register (ENET*x*_ATCOR)..... 948
22.5.94 Time-Stamping Clock Period Register (ENET*x*_ATINC)..... 948
22.5.95 Timestamp of Last Transmitted Frame (ENET*x*_ATSTMP)..... 949
22.5.96 Timer Global Status Register (ENET*x*_TGSR).....949
22.5.97 Timer Control Status Register (ENET*x*_TCSR*n*).....950
22.5.98 Timer Compare Capture Register (ENET*x*_TCCR*n*)..... 952
22.6 Functional description.....952
22.6.1 Ethernet MAC frame formats..... 952
22.6.1.1 Pause Frames.....954
22.6.1.2 Magic packets..... 955
22.6.2 IP and higher layers frame format.....955
22.6.2.1 Ethernet types.....955
22.6.2.2 IPv4 datagram format.....956
22.6.2.3 IPv6 datagram format.....957
22.6.2.4 Internet Control Message Protocol (ICMP) datagram format..... 958
22.6.2.5 User Datagram Protocol (UDP) datagram format.....958
22.6.2.6 TCP datagram format.....959

22.6.3 IEEE 1588 message formats ... 960
22.6.3.1 Transport encapsulation ... 960
22.6.3.1.1 UDP/IP ... 960
22.6.3.1.2 Native Ethernet (PTPv2) ... 960
22.6.3.2 PTP header ... 961
22.6.3.2.1 PTPv1 header ... 961
22.6.3.2.2 PTPv2 header ... 962
22.6.4 MAC receive ... 963
22.6.4.1 Collision detection in half-duplex mode ... 964
22.6.4.2 Preamble processing ... 964
22.6.4.3 MAC address check ... 965
22.6.4.3.1 Unicast address check ... 965
22.6.4.3.2 Multicast and unicast address resolution ... 965
22.6.4.3.3 Broadcast address reject ... 966
22.6.4.3.4 Miss-bit implementation ... 966
22.6.4.4 Frame length/type verification: payload length check ... 967
22.6.4.5 Frame length/type verification: frame length check ... 967
22.6.4.6 VLAN frames processing ... 967
22.6.4.7 Pause frame termination ... 968
22.6.4.8 CRC check ... 968
22.6.4.9 Frame padding removal ... 968
22.6.5 MAC transmit ... 969
22.6.5.1 Frame payload padding ... 970
22.6.5.2 MAC address insertion ... 970
22.6.5.3 CRC-32 generation ... 970
22.6.5.4 Inter-packet gap (IPG) ... 971
22.6.5.5 Collision detection and handling — half-duplex operation only ... 971
22.6.6 Full-duplex flow control operation ... 973
22.6.6.1 Remote device congestion ... 973

22.6.6.2 Local device/FIFO congestion....974
22.6.7 Magic packet detection....975
22.6.7.1 Sleep mode....975
22.6.7.2 Magic packet detection....975
22.6.7.3 Wakeup....975
22.6.8 IP accelerator functions....976
22.6.8.1 Checksum calculation....976
22.6.8.2 Additional padding processing....977
22.6.8.3 32-bit Ethernet payload alignment....977
22.6.8.3.1 Receive processing....978
22.6.8.3.2 Transmit processing....978
22.6.8.4 Received frame discard....978
22.6.8.5 IPv4 fragments....979
22.6.8.6 IPv6 support....979
22.6.8.6.1 Receive processing....979
22.6.8.6.2 Transmit processing....980
22.6.9 Resets and stop controls....980
22.6.9.1 Hardware reset....980
22.6.9.2 Soft reset....980
22.6.9.3 Hardware freeze....981
22.6.9.4 Graceful stop....981
22.6.9.4.1 Graceful transmit stop (GTS)....982
22.6.9.4.2 Graceful receive stop (GRS)....982
22.6.9.4.3 Graceful stop interrupt (GRA)....983
22.6.10 IEEE 1588 functions....983
22.6.10.1 Adjustable timer module....984
22.6.10.1.1 Adjustable timer implementation....984
22.6.10.2 Transmit timestamping....986
22.6.10.3 Receive timestamping....986

22.6.10.4 Time synchronization....986
22.6.10.5 Input Capture and Output Compare....987
22.6.10.5.1 Input capture....987
22.6.10.5.2 Output compare....987
22.6.10.5.3 DMA requests....987
22.6.11 FIFO thresholds....987
22.6.11.1 Receive FIFO....988
22.6.11.2 Transmit FIFO....989
22.6.12 Loopback options....990
22.6.13 Legacy buffer descriptors....991
22.6.13.1 Legacy receive buffer descriptor....991
22.6.13.2 Legacy transmit buffer descriptor....991
22.6.14 Enhanced buffer descriptors....992
22.6.14.1 Enhanced receive buffer descriptor....992
22.6.14.2 Enhanced transmit buffer descriptor....996
22.6.15 Client FIFO application interface....998
22.6.15.1 Data structure description....999
22.6.15.2 Data structure examples....1000
22.6.15.3 Frame status....1001
22.6.16 FIFO protection....1001
22.6.16.1 Transmit FIFO underflow....1002
22.6.16.2 Transmit FIFO overflow....1002
22.6.16.3 Receive FIFO overflow....1003
22.6.17 PHY management interface....1004
22.6.17.1 MDIO clause 22 frame format....1004
22.6.17.2 MDIO clause 45 frame format....1005
22.6.17.3 MDIO clock generation....1006
22.6.17.4 MDIO operation....1006
22.6.18 Ethernet interfaces....1007

22.6.18.1 RMII interface....1007
22.6.18.2 MII Interface — transmit....1008
22.6.18.2.1 Transmit with collision — half-duplex....1009
22.6.18.3 MII interface — receive....1010
22.6.19 Interrupt coalescence....1011
22.6.19.1 Interrupt coalescence setup....1011
22.6.19.2 Updating the frame count threshold on-the-fly....1012
22.6.19.3 Updating the timer threshold on-the-fly....1012

Chapter 23
Enhanced Periodic Interrupt Timer (EPIT)

23.1 Overview....1013
23.1.1 EPIT features....1013
23.1.2 EPIT modes and operations....1014
23.2 External signals....1014
23.3 Clocks....1014
23.4 Functional Description....1016
23.4.1 Operating modes....1016
23.4.1.1 Operating in set-and-forget mode....1016
23.4.1.2 Operating in free-running mode....1016
23.4.2 Operations....1017
23.4.3 Compare Event....1017
23.4.3.1 Counter Value Overwrite....1018
23.4.3.2 Low-Power Mode Behavior....1019
23.4.3.3 Debug Mode Behavior....1019
23.5 Initialization/ Application Information....1019
23.5.1 Change of Clock Source....1019
23.6 EPIT Memory Map/Register Definition....1020
23.6.1 Control register (EPIT*x*_CR)....1020
23.6.2 Status register (EPIT*x*_SR)....1023

23.6.3 Load register (EPIT*x*_LR)..... 1023
23.6.4 Compare register (EPIT*x*_CMPR)..... 1024
23.6.5 Counter register (EPIT*x*_CNR)..... 1024

Chapter 24
Flexible Controller Area Network (FLEXCAN)

24.1 Overview..... 1025
24.1.1 Block Diagram..... 1025
24.1.2 FLEXCAN Module Features..... 1027
24.1.3 Modes of Operation..... 1028
24.2 External Signals..... 1029
24.3 Clocks..... 1030
24.4 Message Buffer Structure..... 1030
24.5 Rx FIFO Structure..... 1034
24.6 Functional Description..... 1038
24.6.1 Functional Overview..... 1038
24.6.2 Transmit Process..... 1038
24.6.3 Arbitration process..... 1039
24.6.3.1 Lowest Mailbox number first..... 1040
24.6.3.2 Highest Mailbox priority first..... 1040
24.6.3.2.1 Local Priority disabled..... 1040
24.6.3.2.2 Local Priority enabled..... 1041
24.6.4 Receive Process..... 1043
24.6.5 Matching Process..... 1044
24.6.6 Move Process..... 1049
24.6.6.1 Move-in..... 1049
24.6.6.2 Move-out..... 1050
24.6.7 Data Coherence..... 1050
24.6.7.1 Transmission Abort Mechanism..... 1051
24.6.7.2 Message Buffer Inactivation..... 1052

24.6.7.3 Message Buffer Lock Mechanism ... 1052
24.6.8 Rx FIFO ... 1054
24.6.9 CAN Protocol Related Features ... 1055
24.6.9.1 Remote Frames ... 1055
24.6.9.2 Overload Frames ... 1056
24.6.9.3 Time Stamp ... 1056
24.6.9.4 Protocol Timing ... 1056
24.6.9.5 Arbitration and Matching Timing ... 1059
24.6.10 Modes of Operation Details ... 1061
24.6.10.1 Freeze Mode ... 1061
24.6.10.2 Module Disable Mode ... 1062
24.6.10.3 Stop Mode ... 1063
24.6.11 Interrupts ... 1064
24.7 Initialization/Application Information ... 1065
24.7.1 FLEXCAN Initialization Sequence ... 1065
24.8 FLEXCAN Memory Map/Register Definition ... 1066
24.8.1 Module Configuration Register (FLEXCAN*x*_MCR) ... 1068
24.8.2 Control 1 Register (FLEXCAN*x*_CTRL1) ... 1073
24.8.3 Free Running Timer Register (FLEXCAN*x*_TIMER) ... 1076
24.8.4 Rx Mailboxes Global Mask Register (FLEXCAN*x*_RXMGMASK) ... 1076
24.8.5 Rx Buffer 14 Mask Register (FLEXCAN*x*_RX14MASK) ... 1077
24.8.6 Rx Buffer 15 Mask Register (FLEXCAN*x*_RX15MASK) ... 1078
24.8.7 Error Counter Register (FLEXCAN*x*_ECR) ... 1079
24.8.8 Error and Status 1 Register (FLEXCAN*x*_ESR1) ... 1080
24.8.9 Interrupt Masks 2 Register (FLEXCAN*x*_IMASK2) ... 1084
24.8.10 Interrupt Masks 1 Register (FLEXCAN*x*_IMASK1) ... 1084
24.8.11 Interrupt Flags 2 Register (FLEXCAN*x*_IFLAG2) ... 1085
24.8.12 Interrupt Flags 1 Register (FLEXCAN*x*_IFLAG1) ... 1085
24.8.13 Control 2 Register (FLEXCAN*x*_CTRL2) ... 1087

24.8.14 Error and Status 2 Register (FLEXCAN*x*_ESR2)..... 1093
24.8.15 CRC Register (FLEXCAN*x*_CRCR)..... 1095
24.8.16 Rx FIFO Global Mask Register (FLEXCAN*x*_RXFGMASK)..... 1096
24.8.17 Rx FIFO Information Register (FLEXCAN*x*_RXFIR)..... 1097
24.8.18 Rx Individual Mask Registers (FLEXCAN*x*_RXIMR0_RXIMR63)..... 1098
24.8.19 Glitch Filter Width Registers (FLEXCAN*x*_GFWR)..... 1098

Chapter 25
General Power Controller (GPC)

25.1 Overview..... 1101
25.2 Clocks..... 1101
25.3 DVFS overview..... 1101
25.3.1 Features..... 1102
25.4 DVFS output event/interrupt configuration..... 1102
25.4.1 Interrupts..... 1102
25.4.2 DVFS Change Request Sequence Diagrams..... 1103
25.5 Power Gating Control (PGC)..... 1104
25.5.1 Overview..... 1105
25.5.1.1 Features..... 1106
25.6 GPC Interrupt Controller (INTC)..... 1106
25.6.1 Interrupt Controller features..... 1106
25.7 GPC Memory Map/Register Definition..... 1107
25.7.1 GPC Interface control register (GPC_CNTR)..... 1107
25.7.2 GPC Power Gating Register (GPC_PGR)..... 1110
25.7.3 IRQ masking register 1 (GPC_IMR1)..... 1110
25.7.4 IRQ masking register 2 (GPC_IMR2)..... 1111
25.7.5 IRQ masking register 3 (GPC_IMR3)..... 1111
25.7.6 IRQ masking register 4 (GPC_IMR4)..... 1112
25.7.7 IRQ status resister 1 (GPC_ISR1)..... 1112
25.7.8 IRQ status resister 2 (GPC_ISR2)..... 1113

25.7.9 IRQ status resister 3 (GPC_ISR3)..... 1113
25.7.10 IRQ status resister 4 (GPC_ISR4)..... 1114
25.8 PGC Memory Map/Register Definition..... 1114
25.8.1 PGC Control Register (PGC_MEGA_CTRL)..... 1115
25.8.2 Power Up Sequence Control Register (PGC_MEGA_PUPSCR)..... 1116
25.8.3 Pull Down Sequence Control Register (PGC_MEGA_PDNSCR)..... 1116
25.8.4 Power Gating Controller Status Register (PGC_MEGA_SR)..... 1117
25.8.5 PGC Control Register (PGC_CPU_CTRL)..... 1118
25.8.6 Power Up Sequence Control Register (PGC_CPU_PUPSCR)..... 1118
25.8.7 Pull Down Sequence Control Register (PGC_CPU_PDNSCR)..... 1119
25.8.8 Power Gating Controller Status Register (PGC_CPU_SR)..... 1120
25.9 DVFSC Memory Map/Register Definition..... 1120
25.9.1 DVFS Thresholds (DVFSC_THRS)..... 1121
25.9.2 DVFS Counters thresholds (DVFSC_COUN)..... 1122
25.9.3 DVFS general purpose bits weight (DVFSC_SIG1)..... 1122
25.9.4 DVFS general purpose bits weight (DVFSC_DVFSSIG0)..... 1123
25.9.5 DVFS general purpose bit 0 weight counter (DVFSC_DVFSGPC0)..... 1124
25.9.6 DVFS general purpose bit 1 weight counter (DVFSC_DVFSGPC1)..... 1125
25.9.7 DVFS general purpose bits enables (DVFSC_DVFSGPBT)..... 1126
25.9.8 DVFS EMAC settings (DVFSC_DVFSEMAC)..... 1128
25.9.9 DVFS Control (DVFSC_CNTR)..... 1129
25.9.10 DVFS Load Tracking Register 0, portion 0 (DVFSC_DVFSLTR0_0)..... 1132
25.9.11 DVFS Load Tracking Register 0, portion 1 (DVFSC_DVFSLTR0_1)..... 1133
25.9.12 DVFS Load Tracking Register 1, portion 0 (DVFSC_DVFSLTR1_0)..... 1133
25.9.13 DVFS Load Tracking Register 3, portion 1 (DVFSC_DVFSLTR1_1)..... 1134
25.9.14 DVFS pattern 0 length (DVFSC_DVFSPT0)..... 1135
25.9.15 DVFS pattern 1 length (DVFSC_DVFSPT1)..... 1135
25.9.16 DVFS pattern 2 length (DVFSC_DVFSPT2)..... 1136
25.9.17 DVFS pattern 3 length (DVFSC_DVFSPT3)..... 1137

Chapter 26
General Purpose Input/Output (GPIO)

26.1 Overview....1139
26.1.1 Block Diagram....1141
26.1.2 Features....1142
26.2 External Signals....1143
26.3 Clocks....1146
26.4 GPIO Functional Description....1146
26.4.1 GPIO Function....1147
26.4.2 GPIO pad structure....1147
26.4.2.1 Input Driver....1148
26.4.2.1.1 Schmitt trigger....1148
26.4.2.1.2 Input keeper....1149
26.4.2.2 Output Driver....1149
26.4.2.2.1 Drive strength....1150
26.4.2.2.2 Output keeper....1150
26.4.2.2.3 PU / PD / Keeper Logic....1151
26.4.2.2.4 Open drain....1151
26.4.3 GPIO Programming....1152
26.4.3.1 GPIO Read Mode....1152
26.4.3.2 GPIO Write Mode....1153
26.4.4 Interrupt Control Unit....1153
26.5 GPIO Memory Map/Register Definition....1153
26.5.1 GPIO data register (GPIO*x*_DR)....1155
26.5.2 GPIO direction register (GPIO*x*_GDIR)....1156
26.5.3 GPIO pad status register (GPIO*x*_PSR)....1156
26.5.4 GPIO interrupt configuration register1 (GPIO*x*_ICR1)....1157
26.5.5 GPIO interrupt configuration register2 (GPIO*x*_ICR2)....1161
26.5.6 GPIO interrupt mask register (GPIO*x*_IMR)....1164

26.5.7 GPIO interrupt status register (GPIO*x*_ISR)..... 1165
26.5.8 GPIO edge select register (GPIO*x*_EDGE_SEL)..... 1166

Chapter 27
General Purpose Media Interface (GPMI)

27.1 Overview.....1167
27.2 External Signals..... 1168
27.3 Clocks..... 1169
27.4 GPMI NAND Mode.....1170
27.4.1 Multiple NAND Support.....1170
27.4.2 GPMI NAND Timing and Clocking.....1171
27.4.3 Basic NAND Timing..... 1171
27.4.3.1 NAND Asynchronous Timing..... 1171
27.4.3.2 NAND Asynchronous EDO Mode Timing.....1173
27.4.3.3 NAND ONFI Source Synchronous Mode Timing.....1176
27.4.3.4 NAND Toggle Mode Timing.....1181
27.4.4 Hardware BCH Interface..... 1188
27.5 Behavior During Reset.....1189
27.6 GPMI Memory Map/Register Definition.....1189
27.6.1 GPMI Control Register 0 Description (GPMI_CTRL0*n*).....1191
27.6.2 GPMI Compare Register Description (GPMI_COMPARE)..... 1193
27.6.3 GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL*n*).....1194
27.6.4 GPMI Integrated ECC Transfer Count Register Description (GPMI_ECCCOUNT)..... 1195
27.6.5 GPMI Payload Address Register Description (GPMI_PAYLOAD)..... 1195
27.6.6 GPMI Auxiliary Address Register Description (GPMI_AUXILIARY)..... 1196
27.6.7 GPMI Control Register 1 Description (GPMI_CTRL1*n*).....1197
27.6.8 GPMI Timing Register 0 Description (GPMI_TIMING0).....1199
27.6.9 GPMI Timing Register 1 Description (GPMI_TIMING1).....1200
27.6.10 GPMI Timing Register 2 Description (GPMI_TIMING2).....1201
27.6.11 GPMI DMA Data Transfer Register Description (GPMI_DATA).....1202

27.6.12 GPMI Status Register Description (GPMI_STAT)..... 1202
27.6.13 GPMI Debug Information Register Description (GPMI_DEBUG)..... 1205
27.6.14 GPMI Version Register Description (GPMI_VERSION)..... 1205
27.6.15 GPMI Debug2 Information Register Description (GPMI_DEBUG2)..... 1206
27.6.16 GPMI Debug3 Information Register Description (GPMI_DEBUG3)..... 1209
27.6.17 GPMI Double Rate Read DLL Control Register Description (GPMI_READ_DDR_DLL_CTRL)..... 1209
27.6.18 GPMI Double Rate Write DLL Control Register Description (GPMI_WRITE_DDR_DLL_CTRL)..... 1211
27.6.19 GPMI Double Rate Read DLL Status Register Description (GPMI_READ_DDR_DLL_STS)..... 1213
27.6.20 GPMI Double Rate Write DLL Status Register Description (GPMI_WRITE_DDR_DLL_STS)..... 1214

Chapter 28
General Purpose Timer (GPT)

28.1 Overview..... 1217
28.1.1 Features..... 1219
28.1.2 Modes and Operation..... 1219
28.2 External Signals..... 1219
28.2.1 External Clock Input..... 1221
28.2.2 Input Capture Trigger Signals..... 1221
28.2.3 Output Compare Signals..... 1221
28.3 Signals..... 0
28.3.1 External Clock Input..... 0
28.3.2 Input Capture Trigger Signals..... 0
28.3.3 Output Compare Signals..... 0
28.4 Clocks..... 1221
28.5 Functional Description..... 1223
28.5.1 Operating Modes..... 1223
28.5.1.1 Restart Mode..... 1223
28.5.1.2 Free-Run Mode..... 1224
28.5.2 Operation..... 1224
28.5.2.1 Input Capture..... 1225

28.5.2.2 Output Compare ... 1226
28.5.2.3 Interrupts ... 1226
28.5.2.4 Low Power Mode Behavior ... 1227
28.5.2.5 Debug Mode Behavior ... 1227
28.6 Initialization/ Application Information ... 1228
28.6.1 Selecting the Clock Source ... 1228
28.7 GPT Memory Map/Register Definition ... 1228
28.7.1 GPT Control Register (GPT*x*_CR) ... 1230
28.7.2 GPT Prescaler Register (GPT*x*_PR) ... 1234
28.7.3 GPT Status Register (GPT*x*_SR) ... 1235
28.7.4 GPT Interrupt Register (GPT*x*_IR) ... 1236
28.7.5 GPT Output Compare Register 1 (GPT*x*_OCR1) ... 1237
28.7.6 GPT Output Compare Register 2 (GPT*x*_OCR2) ... 1238
28.7.7 GPT Output Compare Register 3 (GPT*x*_OCR3) ... 1238
28.7.8 GPT Input Capture Register 1 (GPT*x*_ICR1) ... 1239
28.7.9 GPT Input Capture Register 2 (GPT*x*_ICR2) ... 1239
28.7.10 GPT Counter Register (GPT*x*_CNT) ... 1240

Chapter 29
I2C Controller (I2C)

29.1 Overview ... 1241
29.1.1 Features ... 1243
29.1.2 Modes and operations ... 1244
29.2 External Signals ... 1244
29.3 Clocks ... 1245
29.4 Functional description ... 1245
29.4.1 I2C system configuration ... 1246
29.4.2 Arbitration procedure ... 1246
29.4.3 Clock synchronization ... 1246
29.4.4 Handshaking ... 1247

29.4.5 Clock stretching ... 1247
29.4.6 Peripheral bus accesses ... 1248
29.4.7 Generation of transfer error on IP bus ... 1248
29.4.8 Reset ... 1248
29.4.9 Interrupts ... 1248
29.4.10 Byte order ... 1248
29.5 Initialization ... 1249
29.5.1 Initialization sequence ... 1249
29.5.2 Generation of Start ... 1249
29.5.3 Post-transfer software response ... 1249
29.5.4 Generation of Stop ... 1250
29.5.5 Generation of Repeated Start ... 1250
29.5.6 Slave mode ... 1251
29.5.7 Arbitration lost ... 1251
29.6 Software restriction ... 1257
29.7 I2C Memory Map/Register Definition ... 1258
29.7.1 I2C Address Register (I2C*x*_IADR) ... 1259
29.7.2 I2C Frequency Divider Register (I2C*x*_IFDR) ... 1259
29.7.3 I2C Control Register (I2C*x*_I2CR) ... 1261
29.7.4 I2C Status Register (I2C*x*_I2SR) ... 1262
29.7.5 I2C Data I/O Register (I2C*x*_I2DR) ... 1264

Chapter 30
IOMUX Controller (IOMUXC)

30.1 Overview ... 1265
30.1.1 Features ... 1266
30.2 Clocks ... 1267
30.3 Functional description ... 1267
30.3.1 ALT6 and ALT7 extended muxing modes ... 1268
30.3.2 SW Loopback through SION bit ... 1269

30.3.3 Daisy chain - multi pads driving same module input pin..... 1269
30.4 IOMUXC GPR Memory Map/Register Definition.....1270
30.4.1 GPR0 General Purpose Register (IOMUXC_GPR_GPR0).....1271
30.4.2 GPR1 General Purpose Register (IOMUXC_GPR_GPR1).....1274
30.4.3 GPR2 General Purpose Register (IOMUXC_GPR_GPR2).....1277
30.4.4 GPR3 General Purpose Register (IOMUXC_GPR_GPR3).....1280
30.4.5 GPR4 General Purpose Register (IOMUXC_GPR_GPR4).....1283
30.4.6 GPR5 General Purpose Register (IOMUXC_GPR_GPR5).....1286
30.4.7 GPR9 General Purpose Register (IOMUXC_GPR_GPR9).....1288
30.4.8 GPR10 General Purpose Register (IOMUXC_GPR_GPR10).....1289
30.4.9 GPR14 General Purpose Register (IOMUXC_GPR_GPR14).....1290
30.5 IOMUXC Memory Map/Register Definition..... 1291
30.5.1 SW_MUX_CTL_PAD_BOOT_MODE0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0)..... 1311
30.5.2 SW_MUX_CTL_PAD_BOOT_MODE1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1)..... 1312
30.5.3 SW_MUX_CTL_PAD_SNVS_TAMPER0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER0)..... 1313
30.5.4 SW_MUX_CTL_PAD_SNVS_TAMPER1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER1)..... 1314
30.5.5 SW_MUX_CTL_PAD_SNVS_TAMPER2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER2)..... 1315
30.5.6 SW_MUX_CTL_PAD_SNVS_TAMPER3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER3)..... 1316
30.5.7 SW_MUX_CTL_PAD_SNVS_TAMPER4 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER4)..... 1317
30.5.8 SW_MUX_CTL_PAD_SNVS_TAMPER5 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER5)..... 1318
30.5.9 SW_MUX_CTL_PAD_SNVS_TAMPER6 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER6)..... 1319

30.5.10 SW_MUX_CTL_PAD_SNVS_TAMPER7 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER7)..... 1320
30.5.11 SW_MUX_CTL_PAD_SNVS_TAMPER8 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER8)..... 1321
30.5.12 SW_MUX_CTL_PAD_SNVS_TAMPER9 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER9)..... 1322
30.5.13 SW_MUX_CTL_PAD_JTAG_MOD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_MOD)..... 1323
30.5.14 SW_MUX_CTL_PAD_JTAG_TMS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TMS)..... 1324
30.5.15 SW_MUX_CTL_PAD_JTAG_TDO SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TDO)..... 1325
30.5.16 SW_MUX_CTL_PAD_JTAG_TDI SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TDI)..... 1326
30.5.17 SW_MUX_CTL_PAD_JTAG_TCK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TCK)..... 1327
30.5.18 SW_MUX_CTL_PAD_JTAG_TRST_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TRST_B)..... 1328
30.5.19 SW_MUX_CTL_PAD_GPIO1_IO00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO00)..... 1329
30.5.20 SW_MUX_CTL_PAD_GPIO1_IO01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO01)..... 1330
30.5.21 SW_MUX_CTL_PAD_GPIO1_IO02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO02)..... 1331
30.5.22 SW_MUX_CTL_PAD_GPIO1_IO03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO03)..... 1332
30.5.23 SW_MUX_CTL_PAD_GPIO1_IO04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO04)..... 1333
30.5.24 SW_MUX_CTL_PAD_GPIO1_IO05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO05)..... 1334
30.5.25 SW_MUX_CTL_PAD_GPIO1_IO06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO06)..... 1335

30.5.26 SW_MUX_CTL_PAD_GPIO1_IO07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO07)..... 1336

30.5.27 SW_MUX_CTL_PAD_GPIO1_IO08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO08)..... 1337

30.5.28 SW_MUX_CTL_PAD_GPIO1_IO09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09)..... 1338

30.5.29 SW_MUX_CTL_PAD_UART1_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA).....1339

30.5.30 SW_MUX_CTL_PAD_UART1_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA)..... 1340

30.5.31 SW_MUX_CTL_PAD_UART1_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_CTS_B)..... 1341

30.5.32 SW_MUX_CTL_PAD_UART1_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RTS_B)..... 1342

30.5.33 SW_MUX_CTL_PAD_UART2_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA).....1343

30.5.34 SW_MUX_CTL_PAD_UART2_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RX_DATA)..... 1344

30.5.35 SW_MUX_CTL_PAD_UART2_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_CTS_B)..... 1345

30.5.36 SW_MUX_CTL_PAD_UART2_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RTS_B)..... 1346

30.5.37 SW_MUX_CTL_PAD_UART3_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_TX_DATA).....1347

30.5.38 SW_MUX_CTL_PAD_UART3_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA)..... 1348

30.5.39 SW_MUX_CTL_PAD_UART3_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B)..... 1349

30.5.40 SW_MUX_CTL_PAD_UART3_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B)..... 1350

30.5.41 SW_MUX_CTL_PAD_UART4_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART4_TX_DATA).....1351

30.5.42 SW_MUX_CTL_PAD_UART4_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART4_RX_DATA)..... 1352
30.5.43 SW_MUX_CTL_PAD_UART5_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART5_TX_DATA).....1353
30.5.44 SW_MUX_CTL_PAD_UART5_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART5_RX_DATA)..... 1354
30.5.45 SW_MUX_CTL_PAD_ENET1_RX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA0)..... 1355
30.5.46 SW_MUX_CTL_PAD_ENET1_RX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA1)..... 1356
30.5.47 SW_MUX_CTL_PAD_ENET1_RX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_EN).....1357
30.5.48 SW_MUX_CTL_PAD_ENET1_TX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA0)..... 1358
30.5.49 SW_MUX_CTL_PAD_ENET1_TX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA1)..... 1359
30.5.50 SW_MUX_CTL_PAD_ENET1_TX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_EN)..... 1360
30.5.51 SW_MUX_CTL_PAD_ENET1_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_CLK)..... 1361
30.5.52 SW_MUX_CTL_PAD_ENET1_RX_ER SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_ER)..... 1362
30.5.53 SW_MUX_CTL_PAD_ENET2_RX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA0)..... 1363
30.5.54 SW_MUX_CTL_PAD_ENET2_RX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA1)..... 1364
30.5.55 SW_MUX_CTL_PAD_ENET2_RX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_EN).....1365
30.5.56 SW_MUX_CTL_PAD_ENET2_TX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA0)..... 1366
30.5.57 SW_MUX_CTL_PAD_ENET2_TX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA1)..... 1367

30.5.58 SW_MUX_CTL_PAD_ENET2_TX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_EN)..... 1368

30.5.59 SW_MUX_CTL_PAD_ENET2_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_CLK)..... 1369

30.5.60 SW_MUX_CTL_PAD_ENET2_RX_ER SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_ER)..... 1370

30.5.61 SW_MUX_CTL_PAD_LCD_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_CLK)..... 1371

30.5.62 SW_MUX_CTL_PAD_LCD_ENABLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_ENABLE)..... 1372

30.5.63 SW_MUX_CTL_PAD_LCD_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC)..... 1373

30.5.64 SW_MUX_CTL_PAD_LCD_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC)..... 1374

30.5.65 SW_MUX_CTL_PAD_LCD_RESET SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_RESET)..... 1375

30.5.66 SW_MUX_CTL_PAD_LCD_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00)..... 1376

30.5.67 SW_MUX_CTL_PAD_LCD_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01)..... 1377

30.5.68 SW_MUX_CTL_PAD_LCD_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA02)..... 1378

30.5.69 SW_MUX_CTL_PAD_LCD_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03)..... 1379

30.5.70 SW_MUX_CTL_PAD_LCD_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04)..... 1380

30.5.71 SW_MUX_CTL_PAD_LCD_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA05)..... 1381

30.5.72 SW_MUX_CTL_PAD_LCD_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06)..... 1382

30.5.73 SW_MUX_CTL_PAD_LCD_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07)..... 1383

30.5.74 SW_MUX_CTL_PAD_LCD_DATA08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08)....1384
30.5.75 SW_MUX_CTL_PAD_LCD_DATA09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA09)....1385
30.5.76 SW_MUX_CTL_PAD_LCD_DATA10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10)....1386
30.5.77 SW_MUX_CTL_PAD_LCD_DATA11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11)....1387
30.5.78 SW_MUX_CTL_PAD_LCD_DATA12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12)....1388
30.5.79 SW_MUX_CTL_PAD_LCD_DATA13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA13)....1389
30.5.80 SW_MUX_CTL_PAD_LCD_DATA14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14)....1390
30.5.81 SW_MUX_CTL_PAD_LCD_DATA15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15)....1391
30.5.82 SW_MUX_CTL_PAD_LCD_DATA16 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16)....1392
30.5.83 SW_MUX_CTL_PAD_LCD_DATA17 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA17)....1393
30.5.84 SW_MUX_CTL_PAD_LCD_DATA18 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18)....1394
30.5.85 SW_MUX_CTL_PAD_LCD_DATA19 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19)....1395
30.5.86 SW_MUX_CTL_PAD_LCD_DATA20 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA20)....1396
30.5.87 SW_MUX_CTL_PAD_LCD_DATA21 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21)....1397
30.5.88 SW_MUX_CTL_PAD_LCD_DATA22 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22)....1398
30.5.89 SW_MUX_CTL_PAD_LCD_DATA23 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA23)....1399

30.5.90 SW_MUX_CTL_PAD_NAND_RE_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_RE_B)....1400
30.5.91 SW_MUX_CTL_PAD_NAND_WE_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_WE_B).... 1401
30.5.92 SW_MUX_CTL_PAD_NAND_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA00).... 1402
30.5.93 SW_MUX_CTL_PAD_NAND_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA01).... 1403
30.5.94 SW_MUX_CTL_PAD_NAND_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA02).... 1404
30.5.95 SW_MUX_CTL_PAD_NAND_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA03).... 1405
30.5.96 SW_MUX_CTL_PAD_NAND_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA04).... 1406
30.5.97 SW_MUX_CTL_PAD_NAND_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA05).... 1407
30.5.98 SW_MUX_CTL_PAD_NAND_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA06).... 1408
30.5.99 SW_MUX_CTL_PAD_NAND_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA07).... 1409
30.5.100 SW_MUX_CTL_PAD_NAND_ALE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_ALE)....1410
30.5.101 SW_MUX_CTL_PAD_NAND_WP_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_WP_B)....1411
30.5.102 SW_MUX_CTL_PAD_NAND_READY_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_READY_B).... 1412
30.5.103 SW_MUX_CTL_PAD_NAND_CE0_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CE0_B)....1413
30.5.104 SW_MUX_CTL_PAD_NAND_CE1_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CE1_B)....1414
30.5.105 SW_MUX_CTL_PAD_NAND_CLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CLE)....1415

30.5.106 SW_MUX_CTL_PAD_NAND_DQS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DQS)..... 1416
30.5.107 SW_MUX_CTL_PAD_SD1_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CMD).....1417
30.5.108 SW_MUX_CTL_PAD_SD1_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CLK).....1418
30.5.109 SW_MUX_CTL_PAD_SD1_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0).....1419
30.5.110 SW_MUX_CTL_PAD_SD1_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1).....1420
30.5.111 SW_MUX_CTL_PAD_SD1_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2).....1421
30.5.112 SW_MUX_CTL_PAD_SD1_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3).....1422
30.5.113 SW_MUX_CTL_PAD_CSI_MCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK)..... 1423
30.5.114 SW_MUX_CTL_PAD_CSI_PIXCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK)..... 1424
30.5.115 SW_MUX_CTL_PAD_CSI_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC)..... 1425
30.5.116 SW_MUX_CTL_PAD_CSI_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC)..... 1426
30.5.117 SW_MUX_CTL_PAD_CSI_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA00).....1427
30.5.118 SW_MUX_CTL_PAD_CSI_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA01).....1428
30.5.119 SW_MUX_CTL_PAD_CSI_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA02).....1429
30.5.120 SW_MUX_CTL_PAD_CSI_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA03).....1430
30.5.121 SW_MUX_CTL_PAD_CSI_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA04).....1431

30.5.122 SW_MUX_CTL_PAD_CSI_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA05)....1432
30.5.123 SW_MUX_CTL_PAD_CSI_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA06)....1433
30.5.124 SW_MUX_CTL_PAD_CSI_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA07)....1434
30.5.125 SW_PAD_CTL_PAD_DRAM_ADDR00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR00)....1435
30.5.126 SW_PAD_CTL_PAD_DRAM_ADDR01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR01)....1438
30.5.127 SW_PAD_CTL_PAD_DRAM_ADDR02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR02)....1441
30.5.128 SW_PAD_CTL_PAD_DRAM_ADDR03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR03)....1444
30.5.129 SW_PAD_CTL_PAD_DRAM_ADDR04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR04)....1447
30.5.130 SW_PAD_CTL_PAD_DRAM_ADDR05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR05)....1450
30.5.131 SW_PAD_CTL_PAD_DRAM_ADDR06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR06)....1453
30.5.132 SW_PAD_CTL_PAD_DRAM_ADDR07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR07)....1456
30.5.133 SW_PAD_CTL_PAD_DRAM_ADDR08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR08)....1459
30.5.134 SW_PAD_CTL_PAD_DRAM_ADDR09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR09)....1462
30.5.135 SW_PAD_CTL_PAD_DRAM_ADDR10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR10)....1465
30.5.136 SW_PAD_CTL_PAD_DRAM_ADDR11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR11)....1468
30.5.137 SW_PAD_CTL_PAD_DRAM_ADDR12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR12)....1471

30.5.138 SW_PAD_CTL_PAD_DRAM_ADDR13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR13)..... 1474

30.5.139 SW_PAD_CTL_PAD_DRAM_ADDR14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR14)..... 1477

30.5.140 SW_PAD_CTL_PAD_DRAM_ADDR15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR15)..... 1480

30.5.141 SW_PAD_CTL_PAD_DRAM_DQM0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM0)..... 1483

30.5.142 SW_PAD_CTL_PAD_DRAM_DQM1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM1)..... 1486

30.5.143 SW_PAD_CTL_PAD_DRAM_RAS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_RAS_B)..... 1489

30.5.144 SW_PAD_CTL_PAD_DRAM_CAS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CAS_B)..... 1492

30.5.145 SW_PAD_CTL_PAD_DRAM_CS0_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CS0_B)..... 1495

30.5.146 SW_PAD_CTL_PAD_DRAM_CS1_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CS1_B)..... 1498

30.5.147 SW_PAD_CTL_PAD_DRAM_SDWE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDWE_B)..... 1501

30.5.148 SW_PAD_CTL_PAD_DRAM_ODT0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT0)..... 1504

30.5.149 SW_PAD_CTL_PAD_DRAM_ODT1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT1)..... 1507

30.5.150 SW_PAD_CTL_PAD_DRAM_SDBA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA0)..... 1510

30.5.151 SW_PAD_CTL_PAD_DRAM_SDBA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA1)..... 1513

30.5.152 SW_PAD_CTL_PAD_DRAM_SDBA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA2)..... 1516

30.5.153 SW_PAD_CTL_PAD_DRAM_SDCKE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE0)..... 1519

30.5.154 SW_PAD_CTL_PAD_DRAM_SDCKE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE1)..... 1522

30.5.155 SW_PAD_CTL_PAD_DRAM_SDCLK0_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK0_P)..... 1525

30.5.156 SW_PAD_CTL_PAD_DRAM_SDQS0_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS0_P)..... 1528

30.5.157 SW_PAD_CTL_PAD_DRAM_SDQS1_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS1_P)..... 1531

30.5.158 SW_PAD_CTL_PAD_DRAM_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET)..... 1534

30.5.159 SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_TEST_MODE)..... 1536

30.5.160 SW_PAD_CTL_PAD_POR_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_POR_B)..... 1538

30.5.161 SW_PAD_CTL_PAD_ONOFF SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ONOFF)..... 1540

30.5.162 SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ)..... 1542

30.5.163 SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ)..... 1544

30.5.164 SW_PAD_CTL_PAD_BOOT_MODE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE0)..... 1546

30.5.165 SW_PAD_CTL_PAD_BOOT_MODE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE1)..... 1548

30.5.166 SW_PAD_CTL_PAD_SNVS_TAMPER0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER0)..... 1550

30.5.167 SW_PAD_CTL_PAD_SNVS_TAMPER1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER1)..... 1552

30.5.168 SW_PAD_CTL_PAD_SNVS_TAMPER2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER2)..... 1554

30.5.169 SW_PAD_CTL_PAD_SNVS_TAMPER3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER3)..... 1556

30.5.170 SW_PAD_CTL_PAD_SNVS_TAMPER4 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER4)..... 1558

30.5.171 SW_PAD_CTL_PAD_SNVS_TAMPER5 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER5)..... 1560

30.5.172 SW_PAD_CTL_PAD_SNVS_TAMPER6 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER6)..... 1562

30.5.173 SW_PAD_CTL_PAD_SNVS_TAMPER7 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER7)..... 1564

30.5.174 SW_PAD_CTL_PAD_SNVS_TAMPER8 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER8)..... 1566

30.5.175 SW_PAD_CTL_PAD_SNVS_TAMPER9 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER9)..... 1568

30.5.176 SW_PAD_CTL_PAD_JTAG_MOD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD)..... 1570

30.5.177 SW_PAD_CTL_PAD_JTAG_TMS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS)..... 1572

30.5.178 SW_PAD_CTL_PAD_JTAG_TDO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO)..... 1574

30.5.179 SW_PAD_CTL_PAD_JTAG_TDI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI)..... 1576

30.5.180 SW_PAD_CTL_PAD_JTAG_TCK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK)..... 1578

30.5.181 SW_PAD_CTL_PAD_JTAG_TRST_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B)..... 1580

30.5.182 SW_PAD_CTL_PAD_GPIO1_IO00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO00)..... 1582

30.5.183 SW_PAD_CTL_PAD_GPIO1_IO01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO01)..... 1584

30.5.184 SW_PAD_CTL_PAD_GPIO1_IO02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO02)..... 1586

30.5.185 SW_PAD_CTL_PAD_GPIO1_IO03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO03)..... 1588

30.5.186 SW_PAD_CTL_PAD_GPIO1_IO04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO04)....1590

30.5.187 SW_PAD_CTL_PAD_GPIO1_IO05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO05)....1592

30.5.188 SW_PAD_CTL_PAD_GPIO1_IO06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO06)....1594

30.5.189 SW_PAD_CTL_PAD_GPIO1_IO07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO07)....1596

30.5.190 SW_PAD_CTL_PAD_GPIO1_IO08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08)....1598

30.5.191 SW_PAD_CTL_PAD_GPIO1_IO09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09)....1600

30.5.192 SW_PAD_CTL_PAD_UART1_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA)....1602

30.5.193 SW_PAD_CTL_PAD_UART1_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA)....1604

30.5.194 SW_PAD_CTL_PAD_UART1_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_CTS_B)....1606

30.5.195 SW_PAD_CTL_PAD_UART1_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RTS_B)....1608

30.5.196 SW_PAD_CTL_PAD_UART2_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA)....1610

30.5.197 SW_PAD_CTL_PAD_UART2_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA)....1612

30.5.198 SW_PAD_CTL_PAD_UART2_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_CTS_B)....1614

30.5.199 SW_PAD_CTL_PAD_UART2_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RTS_B)....1616

30.5.200 SW_PAD_CTL_PAD_UART3_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA)....1618

30.5.201 SW_PAD_CTL_PAD_UART3_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA)....1620

30.5.202 SW_PAD_CTL_PAD_UART3_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B)..... 1622

30.5.203 SW_PAD_CTL_PAD_UART3_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B)..... 1624

30.5.204 SW_PAD_CTL_PAD_UART4_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART4_TX_DATA)..... 1626

30.5.205 SW_PAD_CTL_PAD_UART4_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART4_RX_DATA)..... 1628

30.5.206 SW_PAD_CTL_PAD_UART5_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART5_TX_DATA)..... 1630

30.5.207 SW_PAD_CTL_PAD_UART5_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART5_RX_DATA)..... 1632

30.5.208 SW_PAD_CTL_PAD_ENET1_RX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA0)..... 1634

30.5.209 SW_PAD_CTL_PAD_ENET1_RX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA1)..... 1636

30.5.210 SW_PAD_CTL_PAD_ENET1_RX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_EN)..... 1638

30.5.211 SW_PAD_CTL_PAD_ENET1_TX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA0)..... 1640

30.5.212 SW_PAD_CTL_PAD_ENET1_TX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA1)..... 1642

30.5.213 SW_PAD_CTL_PAD_ENET1_TX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_EN)..... 1644

30.5.214 SW_PAD_CTL_PAD_ENET1_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK)..... 1646

30.5.215 SW_PAD_CTL_PAD_ENET1_RX_ER SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_ER)..... 1648

30.5.216 SW_PAD_CTL_PAD_ENET2_RX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA0)..... 1650

30.5.217 SW_PAD_CTL_PAD_ENET2_RX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA1)..... 1652

30.5.218 SW_PAD_CTL_PAD_ENET2_RX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_EN)..... 1654

30.5.219 SW_PAD_CTL_PAD_ENET2_TX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA0).....1656

30.5.220 SW_PAD_CTL_PAD_ENET2_TX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA1).....1658

30.5.221 SW_PAD_CTL_PAD_ENET2_TX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_EN)..... 1660

30.5.222 SW_PAD_CTL_PAD_ENET2_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_CLK).....1662

30.5.223 SW_PAD_CTL_PAD_ENET2_RX_ER SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_ER)..... 1664

30.5.224 SW_PAD_CTL_PAD_LCD_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_CLK)..... 1666

30.5.225 SW_PAD_CTL_PAD_LCD_ENABLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE)..... 1668

30.5.226 SW_PAD_CTL_PAD_LCD_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC).....1670

30.5.227 SW_PAD_CTL_PAD_LCD_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC).....1672

30.5.228 SW_PAD_CTL_PAD_LCD_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_RESET)..... 1674

30.5.229 SW_PAD_CTL_PAD_LCD_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00)..... 1676

30.5.230 SW_PAD_CTL_PAD_LCD_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01)..... 1678

30.5.231 SW_PAD_CTL_PAD_LCD_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02)..... 1680

30.5.232 SW_PAD_CTL_PAD_LCD_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03)..... 1682

30.5.233 SW_PAD_CTL_PAD_LCD_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04)..... 1684

30.5.234 SW_PAD_CTL_PAD_LCD_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05)..... 1686

30.5.235 SW_PAD_CTL_PAD_LCD_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06)..... 1688

30.5.236 SW_PAD_CTL_PAD_LCD_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07)..... 1690

30.5.237 SW_PAD_CTL_PAD_LCD_DATA08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08)..... 1692

30.5.238 SW_PAD_CTL_PAD_LCD_DATA09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09)..... 1694

30.5.239 SW_PAD_CTL_PAD_LCD_DATA10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10)..... 1696

30.5.240 SW_PAD_CTL_PAD_LCD_DATA11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11)..... 1698

30.5.241 SW_PAD_CTL_PAD_LCD_DATA12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12)..... 1700

30.5.242 SW_PAD_CTL_PAD_LCD_DATA13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13)..... 1702

30.5.243 SW_PAD_CTL_PAD_LCD_DATA14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14)..... 1704

30.5.244 SW_PAD_CTL_PAD_LCD_DATA15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15)..... 1706

30.5.245 SW_PAD_CTL_PAD_LCD_DATA16 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16)..... 1708

30.5.246 SW_PAD_CTL_PAD_LCD_DATA17 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17)..... 1710

30.5.247 SW_PAD_CTL_PAD_LCD_DATA18 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18)..... 1712

30.5.248 SW_PAD_CTL_PAD_LCD_DATA19 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19)..... 1714

30.5.249 SW_PAD_CTL_PAD_LCD_DATA20 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20)..... 1716

30.5.250 SW_PAD_CTL_PAD_LCD_DATA21 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21)..... 1718

30.5.251 SW_PAD_CTL_PAD_LCD_DATA22 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22)..... 1720

30.5.252 SW_PAD_CTL_PAD_LCD_DATA23 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23)..... 1722

30.5.253 SW_PAD_CTL_PAD_NAND_RE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_RE_B)..... 1724

30.5.254 SW_PAD_CTL_PAD_NAND_WE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_WE_B)..... 1726

30.5.255 SW_PAD_CTL_PAD_NAND_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA00)..... 1728

30.5.256 SW_PAD_CTL_PAD_NAND_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA01)..... 1730

30.5.257 SW_PAD_CTL_PAD_NAND_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA02)..... 1732

30.5.258 SW_PAD_CTL_PAD_NAND_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA03)..... 1734

30.5.259 SW_PAD_CTL_PAD_NAND_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA04)..... 1736

30.5.260 SW_PAD_CTL_PAD_NAND_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA05)..... 1738

30.5.261 SW_PAD_CTL_PAD_NAND_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA06)..... 1740

30.5.262 SW_PAD_CTL_PAD_NAND_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA07)..... 1742

30.5.263 SW_PAD_CTL_PAD_NAND_ALE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_ALE)..... 1744

30.5.264 SW_PAD_CTL_PAD_NAND_WP_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_WP_B)..... 1746

30.5.265 SW_PAD_CTL_PAD_NAND_READY_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_READY_B)..... 1748

30.5.266 SW_PAD_CTL_PAD_NAND_CE0_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CE0_B)..... 1750

30.5.267 SW_PAD_CTL_PAD_NAND_CE1_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CE1_B)..... 1752

30.5.268 SW_PAD_CTL_PAD_NAND_CLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CLE)..... 1754

30.5.269 SW_PAD_CTL_PAD_NAND_DQS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DQS)..... 1756

30.5.270 SW_PAD_CTL_PAD_SD1_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CMD)..... 1758

30.5.271 SW_PAD_CTL_PAD_SD1_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CLK)..... 1760

30.5.272 SW_PAD_CTL_PAD_SD1_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0)..... 1762

30.5.273 SW_PAD_CTL_PAD_SD1_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1)..... 1764

30.5.274 SW_PAD_CTL_PAD_SD1_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2)..... 1766

30.5.275 SW_PAD_CTL_PAD_SD1_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3)..... 1768

30.5.276 SW_PAD_CTL_PAD_CSI_MCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK)..... 1770

30.5.277 SW_PAD_CTL_PAD_CSI_PIXCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK)..... 1772

30.5.278 SW_PAD_CTL_PAD_CSI_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC)..... 1774

30.5.279 SW_PAD_CTL_PAD_CSI_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC)..... 1776

30.5.280 SW_PAD_CTL_PAD_CSI_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA00)..... 1778

30.5.281 SW_PAD_CTL_PAD_CSI_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA01)..... 1780

30.5.282 SW_PAD_CTL_PAD_CSI_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA02)..... 1782
30.5.283 SW_PAD_CTL_PAD_CSI_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA03)..... 1784
30.5.284 SW_PAD_CTL_PAD_CSI_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA04)..... 1786
30.5.285 SW_PAD_CTL_PAD_CSI_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA05)..... 1788
30.5.286 SW_PAD_CTL_PAD_CSI_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA06)..... 1790
30.5.287 SW_PAD_CTL_PAD_CSI_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA07)..... 1792
30.5.288 SW_PAD_CTL_GRP_ADDDS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_ADDDS)..... 1793
30.5.289 SW_PAD_CTL_GRP_DDRMODE_CTL SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRMODE_CTL)..... 1794
30.5.290 SW_PAD_CTL_GRP_B0DS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_B0DS)..... 1795
30.5.291 SW_PAD_CTL_GRP_DDRPK SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRPK)..... 1796
30.5.292 SW_PAD_CTL_GRP_CTLDS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_CTLDS)..... 1796
30.5.293 SW_PAD_CTL_GRP_B1DS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_B1DS)..... 1797
30.5.294 SW_PAD_CTL_GRP_DDRHYS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRHYS)..... 1798
30.5.295 SW_PAD_CTL_GRP_DDRPKE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRPKE)..... 1799
30.5.296 SW_PAD_CTL_GRP_DDRMODE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRMODE)..... 1800
30.5.297 SW_PAD_CTL_GRP_DDR_TYPE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDR_TYPE)..... 1801
30.5.298 USB_OTG1_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_ID_SELECT_INPUT). 1802
30.5.299 USB_OTG2_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_ID_SELECT_INPUT). 1802
30.5.300 CCM_PMIC_READY_SELECT_INPUT DAISY Register (IOMUXC_CCM_PMIC_READY_SELECT_INPUT)..... 1803
30.5.301 CSI_DATA02_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA02_SELECT_INPUT)..... 1804
30.5.302 CSI_DATA03_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA03_SELECT_INPUT)..... 1805

30.5.303 CSI_DATA05_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA05_SELECT_INPUT).....1806
30.5.304 CSI_DATA00_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA00_SELECT_INPUT).....1807
30.5.305 CSI_DATA01_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA01_SELECT_INPUT).....1808
30.5.306 CSI_DATA04_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA04_SELECT_INPUT).....1809
30.5.307 CSI_DATA06_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA06_SELECT_INPUT).....1810
30.5.308 CSI_DATA07_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA07_SELECT_INPUT).....1811
30.5.309 CSI_DATA08_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA08_SELECT_INPUT).....1812
30.5.310 CSI_DATA09_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA09_SELECT_INPUT).....1813
30.5.311 CSI_DATA10_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA10_SELECT_INPUT).....1814
30.5.312 CSI_DATA11_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA11_SELECT_INPUT).....1815
30.5.313 CSI_DATA12_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA12_SELECT_INPUT).....1816
30.5.314 CSI_DATA13_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA13_SELECT_INPUT).....1817
30.5.315 CSI_DATA14_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA14_SELECT_INPUT).....1818
30.5.316 CSI_DATA15_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA15_SELECT_INPUT).....1819
30.5.317 CSI_DATA16_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA16_SELECT_INPUT).....1820
30.5.318 CSI_DATA17_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA17_SELECT_INPUT).....1821
30.5.319 CSI_DATA18_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA18_SELECT_INPUT).....1822
30.5.320 CSI_DATA19_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA19_SELECT_INPUT).....1823
30.5.321 CSI_DATA20_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA20_SELECT_INPUT).....1824
30.5.322 CSI_DATA21_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA21_SELECT_INPUT).....1825
30.5.323 CSI_DATA22_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA22_SELECT_INPUT).....1826
30.5.324 CSI_DATA23_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA23_SELECT_INPUT).....1827
30.5.325 CSI_HSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_HSYNC_SELECT_INPUT).....1828
30.5.326 CSI_PIXCLK_SELECT_INPUT DAISY Register (IOMUXC_CSI_PIXCLK_SELECT_INPUT)..... 1829
30.5.327 CSI_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_VSYNC_SELECT_INPUT).....1830
30.5.328 CSI_FIELD_SELECT_INPUT DAISY Register (IOMUXC_CSI_FIELD_SELECT_INPUT)..... 1831
30.5.329 ECSPI1_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SCLK_SELECT_INPUT)... 1832
30.5.330 ECSPI1_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MISO_SELECT_INPUT)....1833
30.5.331 ECSPI1_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MOSI_SELECT_INPUT)....1834

30.5.332 ECSPI1_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SS0_B_SELECT_INPUT). 1835
30.5.333 ECSPI2_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SCLK_SELECT_INPUT)... 1836
30.5.334 ECSPI2_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MISO_SELECT_INPUT)....1837
30.5.335 ECSPI2_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MOSI_SELECT_INPUT)....1838
30.5.336 ECSPI2_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SS0_B_SELECT_INPUT). 1839
30.5.337 ECSPI3_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SCLK_SELECT_INPUT)... 1840
30.5.338 ECSPI3_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MISO_SELECT_INPUT)....1841
30.5.339 ECSPI3_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MOSI_SELECT_INPUT)....1842
30.5.340 ECSPI3_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SS0_B_SELECT_INPUT). 1843
30.5.341 ECSPI4_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SCLK_SELECT_INPUT)... 1844
30.5.342 ECSPI4_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MISO_SELECT_INPUT)....1845
30.5.343 ECSPI4_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MOSI_SELECT_INPUT)....1846
30.5.344 ECSPI4_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SS0_B_SELECT_INPUT). 1847
30.5.345 ENET1_REF_CLK1_SELECT_INPUT DAISY Register (IOMUXC_ENET1_REF_CLK1_SELECT_INPUT).....1847
30.5.346 ENET1_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET1_MAC0_MDIO_SELECT_INPUT)..... 1848
30.5.347 ENET2_REF_CLK2_SELECT_INPUT DAISY Register (IOMUXC_ENET2_REF_CLK2_SELECT_INPUT).....1849
30.5.348 ENET2_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET2_MAC0_MDIO_SELECT_INPUT)..... 1850
30.5.349 FLEXCAN1_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN1_RX_SELECT_INPUT)..... 1850
30.5.350 FLEXCAN2_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN2_RX_SELECT_INPUT)..... 1851
30.5.351 GPT1_CAPTURE1_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CAPTURE1_SELECT_INPUT).....1852
30.5.352 GPT1_CAPTURE2_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CAPTURE2_SELECT_INPUT).....1853
30.5.353 GPT1_CLK_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CLK_SELECT_INPUT)..... 1854

30.5.354 GPT2_CAPTURE1_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CAPTURE1_SELECT_INPUT)....1855
30.5.355 GPT2_CAPTURE2_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CAPTURE2_SELECT_INPUT)....1856
30.5.356 GPT2_CLK_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CLK_SELECT_INPUT)....1857
30.5.357 I2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SCL_SELECT_INPUT)....1857
30.5.358 I2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SDA_SELECT_INPUT)....1858
30.5.359 I2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SCL_SELECT_INPUT)....1859
30.5.360 I2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SDA_SELECT_INPUT)....1859
30.5.361 I2C3_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SCL_SELECT_INPUT)....1860
30.5.362 I2C3_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SDA_SELECT_INPUT)....1861
30.5.363 I2C4_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SCL_SELECT_INPUT)....1861
30.5.364 I2C4_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SDA_SELECT_INPUT)....1862
30.5.365 KPP_COL0_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL0_SELECT_INPUT)....1863
30.5.366 KPP_COL1_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL1_SELECT_INPUT)....1864
30.5.367 KPP_COL2_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL2_SELECT_INPUT)....1865
30.5.368 KPP_ROW0_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW0_SELECT_INPUT)....1866
30.5.369 KPP_ROW1_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW1_SELECT_INPUT)....1867
30.5.370 KPP_ROW2_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW2_SELECT_INPUT)....1868
30.5.371 LCD_BUSY_SELECT_INPUT DAISY Register (IOMUXC_LCD_BUSY_SELECT_INPUT)....1869
30.5.372 SAI1_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_MCLK_SELECT_INPUT)....1870
30.5.373 SAI1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA_SELECT_INPUT)....1871
30.5.374 SAI1_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_BCLK_SELECT_INPUT)....1872
30.5.375 SAI1_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_SYNC_SELECT_INPUT)....1873
30.5.376 SAI2_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_MCLK_SELECT_INPUT)....1874
30.5.377 SAI2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_DATA_SELECT_INPUT)....1875

30.5.378 SAI2_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_BCLK_SELECT_INPUT)..... 1876
30.5.379 SAI2_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_SYNC_SELECT_INPUT)..... 1877
30.5.380 SAI3_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_MCLK_SELECT_INPUT)..... 1878
30.5.381 SAI3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_DATA_SELECT_INPUT)..... 1879
30.5.382 SAI3_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_BCLK_SELECT_INPUT)..... 1880
30.5.383 SAI3_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_SYNC_SELECT_INPUT)..... 1881
30.5.384 SDMA_EVENTS0_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS0_SELECT_INPUT)..... 1881
30.5.385 SDMA_EVENTS1_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS1_SELECT_INPUT)..... 1882
30.5.386 SPDIF_IN_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_IN_SELECT_INPUT)..... 1883
30.5.387 SPDIF_EXT_CLK_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_EXT_CLK_SELECT_INPUT)..... 1884
30.5.388 UART1_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART1_RTS_B_SELECT_INPUT) 1884
30.5.389 UART1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART1_RX_DATA_SELECT_INPUT)..... 1885
30.5.390 UART2_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART2_RTS_B_SELECT_INPUT) 1886
30.5.391 UART2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART2_RX_DATA_SELECT_INPUT)..... 1886
30.5.392 UART3_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART3_RTS_B_SELECT_INPUT) 1887
30.5.393 UART3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART3_RX_DATA_SELECT_INPUT)..... 1888
30.5.394 UART4_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART4_RTS_B_SELECT_INPUT) 1888
30.5.395 UART4_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART4_RX_DATA_SELECT_INPUT)..... 1889
30.5.396 UART5_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART5_RTS_B_SELECT_INPUT) 1890

30.5.397 UART5_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART5_RX_DATA_SELECT_INPUT) 1890
30.5.398 UART6_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART6_RTS_B_SELECT_INPUT) 1891
30.5.399 UART6_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART6_RX_DATA_SELECT_INPUT) 1892
30.5.400 UART7_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART7_RTS_B_SELECT_INPUT) 1892
30.5.401 UART7_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART7_RX_DATA_SELECT_INPUT) 1893
30.5.402 UART8_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART8_RTS_B_SELECT_INPUT) 1894
30.5.403 UART8_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART8_RX_DATA_SELECT_INPUT) 1894
30.5.404 USB_OTG2_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_OC_SELECT_INPUT) 1895
30.5.405 USB_OTG_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG_OC_SELECT_INPUT).. 1896
30.5.406 USDHC1_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_CD_B_SELECT_INPUT) 1896
30.5.407 USDHC1_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_WP_SELECT_INPUT)..... 1897
30.5.408 USDHC2_CLK_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CLK_SELECT_INPUT).. 1898
30.5.409 USDHC2_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CD_B_SELECT_INPUT) 1898
30.5.410 USDHC2_CMD_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CMD_SELECT_INPUT) 1899
30.5.411 USDHC2_DATA0_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA0_SELECT_INPUT) 1900
30.5.412 USDHC2_DATA1_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA1_SELECT_INPUT) 1900
30.5.413 USDHC2_DATA2_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA2_SELECT_INPUT) 1901
30.5.414 USDHC2_DATA3_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA3_SELECT_INPUT) 1902
30.5.415 USDHC2_DATA4_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA4_SELECT_INPUT) 1902

30.5.416 USDHC2_DATA5_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA5_SELECT_INPUT)....1903
30.5.417 USDHC2_DATA6_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA6_SELECT_INPUT)....1904
30.5.418 USDHC2_DATA7_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA7_SELECT_INPUT)....1904
30.5.419 USDHC2_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_WP_SELECT_INPUT)....1905

Chapter 31
Keypad Port (KPP)

31.1 Overview....1907
31.1.1 Features....1909
31.1.2 Modes and Operations....1909
31.2 Clocks....1909
31.3 External Signals....1909
31.3.1 Input Pins....1910
31.3.2 Output Pins....1911
31.3.3 Generation of Transfer Error Signal on Peripheral Bus....1911
31.4 Functional Description....1911
31.4.1 Keypad Matrix Construction....1912
31.4.2 Keypad Port Configuration....1912
31.4.3 Keypad Matrix Scanning....1912
31.4.4 Keypad Standby....1913
31.4.5 Glitch Suppression on Keypad Inputs....1913
31.4.6 Multiple Key Closures....1915
31.4.6.1 Ghost Key Problem and Correction....1917
31.4.7 3-Point Contact Keys Support....1919
31.5 Initialization/Application Information....1920
31.5.1 Typical Keypad Configuration and Scanning Sequence....1921
31.5.2 Key Press Interrupt Scanning Sequence....1921
31.5.3 Additional Comments....1921

31.6 KPP Memory Map/Register Definition ... 1922
31.6.1 Keypad Control Register (KPP_KPCR) ... 1922
31.6.2 Keypad Status Register (KPP_KPSR) ... 1923
31.6.3 Keypad Data Direction Register (KPP_KDDR) ... 1925
31.6.4 Keypad Data Register (KPP_KPDR) ... 1925

Chapter 32
Enhanced LCD Interface (eLCDIF)

32.1 Overview ... 1927
32.2 External Signals ... 1927
32.3 Clocks ... 1928
32.4 Functional Description ... 1929
32.4.1 Bus Interface Mechanisms ... 1930
32.4.1.1 Bus Master Operation in Write/Display Modes ... 1931
32.4.1.2 System Bus Master Performance ... 1931
32.4.2 Write Data Path ... 1932
32.4.3 Read Data Path ... 1938
32.4.4 eLCDIF Interrupts ... 1943
32.4.5 Initializing the eLCDIF ... 1943
32.4.5.1 Write Modes ... 1943
32.4.5.2 MPU Read Mode ... 1944
32.4.6 MPU Interface ... 1945
32.4.6.1 Code Example to Initialize the eLCDIF in MPU Write Mode ... 1947
32.4.7 VSYNC Interface ... 1947
32.4.7.1 Code Example to Initialize eLCDIF in VSYNC Mode ... 1948
32.4.8 DOTCLK Interface ... 1949
32.4.8.1 Code Example ... 1951
32.4.9 CSI HANDSHAKE INTERFACE ... 1951
32.4.10 Alpha Blending Interface ... 1952
32.4.11 ITU-R BT.656 Digital Video Interface (DVI) ... 1952

32.4.12 eLCDIF Pin Usage by Interface Mode 1954
32.5 Behavior During Reset 1957
32.6 ELCDIF Memory Map/Register Definition 1957
32.6.1 eLCDIF General Control Register (LCDIF_RL*n*) 1960
32.6.2 eLCDIF General Control1 Register (LCDIF_CTRL1*n*) 1963
32.6.3 eLCDIF General Control2 Register (LCDIF_CTRL2*n*) 1965
32.6.4 eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT) 1968
32.6.5 LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF) 1968
32.6.6 LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF) 1969
32.6.7 LCD Interface Timing Register (LCDIF_TIMING) 1969
32.6.8 eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0*n*) 1970
32.6.9 eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1) 1972
32.6.10 LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2) 1972
32.6.11 eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3) 1973
32.6.12 eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4) 1974
32.6.13 Digital Video Interface Control0 Register (LCDIF_DVICTRL0) 1975
32.6.14 Digital Video Interface Control1 Register (LCDIF_DVICTRL1) 1975
32.6.15 Digital Video Interface Control2 Register (LCDIF_DVICTRL2) 1976
32.6.16 Digital Video Interface Control3 Register (LCDIF_DVICTRL3) 1977
32.6.17 Digital Video Interface Control4 Register (LCDIF_DVICTRL4) 1978
32.6.18 RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF_CSC_COEFF0) 1979
32.6.19 RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF_CSC_COEFF1) 1980
32.6.20 RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF_CSC_COEFF2) 1981
32.6.21 RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF_CSC_COEFF3) 1981
32.6.22 RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF_CSC_COEFF4) 1982
32.6.23 RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF_CSC_OFFSET) 1983
32.6.24 RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF_CSC_LIMIT) 1983
32.6.25 LCD Interface Data Register (LCDIF_DATA) 1984
32.6.26 Bus Master Error Status Register (LCDIF_BM_ERROR_STAT) 1985

32.6.27 CRC Status Register (LCDIF_CRC_STAT)..... 1985
32.6.28 LCD Interface Status Register (LCDIF_STAT)..... 1986
32.6.29 LCD Interface Version Register (LCDIF_VERSION)..... 1988
32.6.30 LCD Interface Debug0 Register (LCDIF_DEBUG0)..... 1988
32.6.31 LCD Interface Debug1 Register (LCDIF_DEBUG1)..... 1991
32.6.32 LCD Interface Debug2 Register (LCDIF_DEBUG2)..... 1992
32.6.33 eLCDIF Threshold Register (LCDIF_THRES)..... 1992
32.6.34 eLCDIF AS Buffer Control Register (LCDIF_AS_CTRL)..... 1994
32.6.35 Alpha Surface Buffer Pointer (LCDIF_AS_BUF)..... 1996
32.6.36 LCDIF_AS_NEXT_BUF..... 1997
32.6.37 eLCDIF Overlay Color Key Low (LCDIF_AS_CLRKEYLOW)..... 1997
32.6.38 eLCDIF Overlay Color Key High (LCDIF_AS_CLRKEYHIGH)..... 1998
32.6.39 LCD working insync mode with CSI for VSYNC delay (LCDIF_SYNC_DELAY)..... 1998
32.6.40 eLCDIF Interface Debug3 Register (LCDIF_DEBUG3)..... 1999
32.6.41 LCD Interface Debug4 (LCDIF_DEBUG4)..... 2000
32.6.42 LCD Interface Debug5 (LCDIF_DEBUG5)..... 2001

Chapter 33
Multi Mode DDR Controller (MMDC)

33.1 Overview..... 2003
33.1.1 MMDC feature summary..... 2004
33.2 External Signals..... 2006
33.3 Clocks..... 2007
33.4 Functional Description..... 2007
33.4.1 Write/Read data flow..... 2007
33.4.1.1 Write data flow..... 2007
33.4.1.2 Read data flow..... 2008
33.4.2 MMDC initialization..... 2008
33.4.3 Configuring the MMDC registers..... 2010
33.4.4 MMDC Address Space..... 2010

33.4.4.1 Address decoding 2010
33.4.4.2 Chip select settings..... 2013
33.4.4.2.1 Creating 4 Gbyte address space with 2 Gbyte CS density..... 2013
33.4.4.2.2 Creating 2 Gbyte address spaces with 1 Gbyte CS density..... 2013
33.4.4.3 Translation of AXI accesses to DDR accessess..... 2014
33.4.4.3.1 Example 1..... 2014
33.4.4.3.2 Example 2..... 2015
33.4.4.3.3 Example 3..... 2016
33.4.4.3.4 Example 4..... 2016
33.4.4.3.5 Example 5..... 2017
33.4.4.4 Address mirroring 2018
33.4.5 LPDDR2 and DDR3 pin mux mapping..... 2018
33.4.6 Power Saving and Clock Frequency Change modes..... 2019
33.4.6.1 Power saving general..... 2020
33.4.6.2 Self refresh and Frequency change entry/exit..... 2021
33.4.7 Reset 2022
33.4.7.1 Hard reset..... 2022
33.4.7.2 Warm reset..... 2022
33.4.7.3 Software reset 2023
33.4.8 Refresh Scheme..... 2024
33.4.9 Burst Length options towards DDR..... 2024
33.4.10 Exclusive accesses handling..... 2025
33.4.11 AXI Error Handling..... 2026
33.5 Performance..... 2026
33.5.1 Arbitration and reordering mechanism..... 2026
33.5.1.1 Arbitration General..... 2026
33.5.1.2 Real time channel mode..... 2027
33.5.1.3 Dynamic scoring mode (Arbitration Winning Conditions)..... 2027
33.5.1.4 Guarding (aging) mechanism..... 2028

33.5.2 Prediction mechanism 2029
33.5.3 Special Optimization for accesses towards DDR3 2029
33.6 MMDC Debug 2030
33.6.1 Hardware debug monitor 2030
33.6.2 Step By Step (SBS) software monitor 2031
33.7 MMDC Profiling 2031
33.8 LPDDR2 Refresh Rate Update and Timing Derating 2032
33.9 Changing MMDC SDCLK Frequency 2033
33.9.1 Changing MMDC SDCLK Frequency Using DDR3/DDR3L Memories 2033
33.9.1.1 High Frequency to Low Frequency Transition (DLL On to DLL Off) 2036
33.9.1.2 Low Frequency to High Frequency Transition (DLL Off to DLL On) 2037
33.9.2 Changing MMDC Frequency Using LPDDR2 2038
33.9.2.1 High Frequency to Low Frequency Transition 2038
33.9.2.2 Low Frequency to High Frequency Transition 2039
33.10 ODT Configuration 2040
33.11 Calibration Process 2041
33.11.1 Delay-line 2042
33.11.2 ZQ calibration 2042
33.11.2.1 ZQ automatic (hardware) calibration process 2043
33.11.2.1.1 ZQ automatic Pull-up calibration 2044
33.11.2.1.2 ZQ automatic Pull-down calibration 2044
33.11.2.2 ZQ software calibration process 2045
33.11.2.3 ZQ calibration commands 2045
33.11.3 Read DQS Gating Calibration 2045
33.11.3.1 Hardware DQS Gating Calibration 2045
33.11.3.1.1 Hardware DQS Calibration with MPR 2046
33.11.3.1.2 Hardware DQS Calibration with pre-defined value 2046
33.11.3.2 SW read DQS gating Calibration 2049
33.11.3.2.1 SW read Calibration with MPR 2049

33.11.3.2.2 SW read Calibration with pre-defined value..... 2049
33.11.4 Read Calibration..... 2051
33.11.4.1 Hardware (automatic) Read Calibration..... 2052
33.11.4.1.1 Hardware (automatic) Calibration with MPR (DDR3) /DQ Calibration (LPDDR2)..... 2052
33.11.4.1.2 Hardware (automatic) Calibration with pre-defined value..... 2052
33.11.4.2 SW Read Calibration..... 2054
33.11.4.2.1 Calibration with MPR(DDR3)/DQ calibration(LPDDR2)..... 2054
33.11.4.2.2 Calibration with pre-defined value..... 2055
33.11.5 Write Calibration..... 2056
33.11.5.1 HW (automatic) Write Calibration..... 2057
33.11.5.2 SW Write Calibration..... 2058
33.11.6 Write leveling Calibration..... 2060
33.11.6.1 Hardware Write Leveling Calibration..... 2060
33.11.6.2 SW Write Leveling Calibration..... 2062
33.11.7 Write fine tuning..... 2063
33.11.8 Read fine tuning..... 2063
33.11.9 ZQ Fine Tuning..... 2063
33.11.10 Duty cycle adjustment..... 2064
33.12 MMDC Memory Map/Register Definition..... 2064
33.12.1 MMDC Core Control Register (MMDC_MDCTL)..... 2067
33.12.2 MMDC Core Power Down Control Register (MMDC_MDPDC)..... 2069
33.12.3 MMDC Core ODT Timing Control Register (MMDC_MDOTC)..... 2071
33.12.4 MMDC Core Timing Configuration Register 0 (MMDC_MDCFG0)..... 2073
33.12.5 MMDC Core Timing Configuration Register 1 (MMDC_MDCFG1)..... 2075
33.12.6 MMDC Core Timing Configuration Register 2 (MMDC_MDCFG2)..... 2077
33.12.7 MMDC Core Miscellaneous Register (MMDC_MDMISC)..... 2079
33.12.8 MMDC Core Special Command Register (MMDC_MDSCR)..... 2082
33.12.9 MMDC Core Refresh Control Register (MMDC_MDREF)..... 2085

33.12.10 MMDC Core Read/Write Command Delay Register (MMDC_MDRWD)..... 2087
33.12.11 MMDC Core Out of Reset Delays Register (MMDC_MDOR)..... 2089
33.12.12 MMDC Core MRR Data Register (MMDC_MDMRR)..... 2090
33.12.13 MMDC Core Timing Configuration Register 3 (MMDC_MDCFG3LP)..... 2091
33.12.14 MMDC Core MR4 Derating Register (MMDC_MDMR4)..... 2093
33.12.15 MMDC Core Address Space Partition Register (MMDC_MDASP)..... 2095
33.12.16 MMDC Core AXI Reordering Control Register (MMDC_MAARCR)..... 2096
33.12.17 MMDC Core Power Saving Control and Status Register (MMDC_MAPSR)..... 2099
33.12.18 MMDC Core Exclusive ID Monitor Register0 (MMDC_MAEXIDR0)..... 2101
33.12.19 MMDC Core Exclusive ID Monitor Register1 (MMDC_MAEXIDR1)..... 2102
33.12.20 MMDC Core Debug and Profiling Control Register 0 (MMDC_MADPCR0)..... 2103
33.12.21 MMDC Core Debug and Profiling Control Register 1 (MMDC_MADPCR1)..... 2104
33.12.22 MMDC Core Debug and Profiling Status Register 0 (MMDC_MADPSR0)..... 2105
33.12.23 MMDC Core Debug and Profiling Status Register 1 (MMDC_MADPSR1)..... 2105
33.12.24 MMDC Core Debug and Profiling Status Register 2 (MMDC_MADPSR2)..... 2106
33.12.25 MMDC Core Debug and Profiling Status Register 3 (MMDC_MADPSR3)..... 2106
33.12.26 MMDC Core Debug and Profiling Status Register 4 (MMDC_MADPSR4)..... 2107
33.12.27 MMDC Core Debug and Profiling Status Register 5 (MMDC_MADPSR5)..... 2107
33.12.28 MMDC Core Step By Step Address Register (MMDC_MASBS0)..... 2108
33.12.29 MMDC Core Step By Step Address Attributes Register (MMDC_MASBS1)..... 2108
33.12.30 MMDC Core General Purpose Register (MMDC_MAGENP)..... 2109
33.12.31 MMDC PHY ZQ HW control register (MMDC_MPZQHWCTRL)..... 2110
33.12.32 MMDC PHY ZQ SW control register (MMDC_MPZQSWCTRL)..... 2113
33.12.33 MMDC PHY Write Leveling Configuration and Error Status Register (MMDC_MPWLGCR)..... 2115
33.12.34 MMDC PHY Write Leveling Delay Control Register 0 (MMDC_MPWLDECTRL0)..... 2118
33.12.35 MMDC PHY Write Leveling delay-line Status Register (MMDC_MPWLDLST)..... 2121
33.12.36 MMDC PHY ODT control register (MMDC_MPODTCTRL)..... 2122
33.12.37 MMDC PHY Read DQ Byte0 Delay Register (MMDC_MPRDDQBY0DL)..... 2124
33.12.38 MMDC PHY Read DQ Byte1 Delay Register (MMDC_MPRDDQBY1DL)..... 2127

33.12.39 MMDC PHY Write DQ Byte0 Delay Register (MMDC_MPWRDQBY0DL)..... 2130
33.12.40 MMDC PHY Write DQ Byte1 Delay Register (MMDC_MPWRDQBY1DL)..... 2132
33.12.41 MMDC PHY Read DQS Gating Control Register 0 (MMDC_MPDGCTRL0)..... 2134
33.12.42 MMDC PHY Read DQS Gating delay-line Status Register (MMDC_MPDGDLST0)..... 2137
33.12.43 MMDC PHY Read delay-lines Configuration Register (MMDC_MPRDDLCTL)..... 2139
33.12.44 MMDC PHY Read delay-lines Status Register (MMDC_MPRDDLST)..... 2141
33.12.45 MMDC PHY Write delay-lines Configuration Register (MMDC_MPWRDLCTL)..... 2143
33.12.46 MMDC PHY Write delay-lines Status Register (MMDC_MPWRDLST)..... 2145
33.12.47 MMDC PHY CK Control Register (MMDC_MPSDCTRL)..... 2146
33.12.48 MMDC ZQ LPDDR2 HW Control Register (MMDC_MPZQLP2CTL)..... 2147
33.12.49 MMDC PHY Read Delay HW Calibration Control Register (MMDC_MPRDDLHWCTL)..... 2149
33.12.50 MMDC PHY Write Delay HW Calibration Control Register (MMDC_MPWRDLHWCTL)..... 2152
33.12.51 MMDC PHY Read Delay HW Calibration Status Register 0 (MMDC_MPRDDLHWST0)..... 2154
33.12.52 MMDC PHY Write Delay HW Calibration Status Register 0 (MMDC_MPWRDLHWST0)..... 2155
33.12.53 MMDC PHY Write Leveling HW Error Register (MMDC_MPWLHWERR)..... 2156
33.12.54 MMDC PHY Read DQS Gating HW Status Register 0 (MMDC_MPDGHWST0)..... 2156
33.12.55 MMDC PHY Read DQS Gating HW Status Register 1 (MMDC_MPDGHWST1)..... 2157
33.12.56 MMDC PHY Pre-defined Compare Register 1 (MMDC_MPPDCMPR1)..... 2157
33.12.57 MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MMDC_MPPDCMPR2)..... 2159
33.12.58 MMDC PHY SW Dummy Access Register (MMDC_MPSWDAR0)..... 2162
33.12.59 MMDC PHY SW Dummy Read Data Register 0 (MMDC_MPSWDRDR0)..... 2164
33.12.60 MMDC PHY SW Dummy Read Data Register 1 (MMDC_MPSWDRDR1)..... 2164
33.12.61 MMDC PHY SW Dummy Read Data Register 2 (MMDC_MPSWDRDR2)..... 2165
33.12.62 MMDC PHY SW Dummy Read Data Register 3 (MMDC_MPSWDRDR3)..... 2165
33.12.63 MMDC PHY SW Dummy Read Data Register 4 (MMDC_MPSWDRDR4)..... 2165
33.12.64 MMDC PHY SW Dummy Read Data Register 5 (MMDC_MPSWDRDR5)..... 2166
33.12.65 MMDC PHY SW Dummy Read Data Register 6 (MMDC_MPSWDRDR6)..... 2166
33.12.66 MMDC PHY SW Dummy Read Data Register 7 (MMDC_MPSWDRDR7)..... 2167

33.12.67 MMDC PHY Measure Unit Register (MMDC_MPMUR0)....2167
33.12.68 MMDC Write CA delay-line controller (MMDC_MPWRCADL)....2168
33.12.69 MMDC Duty Cycle Control Register (MMDC_MPDCCR)....2170

Chapter 34
Medium Quality Sound (MQS)

34.1 Overview....2173
34.2 Block Diagram....2173
34.3 External Signals....2174
34.4 Programmability....2175
34.5 Usage Model....2175

Chapter 35
On-Chip OTP Controller (OCOTP_CTRL)

35.1 Overview....2177
35.1.1 Features....2177
35.2 Clocks....2177
35.3 Top-Level Symbol and Functional Overview....2178
35.3.1 Operation....2178
35.3.1.1 Shadow Register Reload....2179
35.3.1.2 Fuse and Shadow register read....2179
35.3.1.3 Fuse and Shadow Register Writes....2179
35.3.1.4 Write Postamble....2181
35.3.2 Fuse Shadow Memory Footprint....2181
35.3.3 OTP Read/Write Timing Parameters....2183
35.3.4 Hardware Visible Fuses....2183
35.3.5 Behavior During Reset....2183
35.3.6 Secure JTAG control....2183
35.4 Fuse Map....2184
35.5 OCOTP Memory Map/Register Definition....2184
35.5.1 OTP Controller Control Register (OCOTP_CTRL*n*)....2190

35.5.2 OTP Controller Timing Register (OCOTP_TIMING)..... 2192
35.5.3 OTP Controller Write Data Register (OCOTP_DATA)..... 2193
35.5.4 OTP Controller Write Data Register (OCOTP_READ_CTRL)..... 2193
35.5.5 OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA)..... 2194
35.5.6 Sticky bit Register (OCOTP_SW_STICKY)..... 2195
35.5.7 Software Controllable Signals Register (OCOTP_SCS*n*)..... 2196
35.5.8 OTP Controller CRC test address (OCOTP_CRC_ADDR)..... 2197
35.5.9 OTP Controller CRC Value Register (OCOTP_CRC_VALUE)..... 2198
35.5.10 OTP Controller Version Register (OCOTP_VERSION)..... 2198
35.5.11 Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)..... 2199
35.5.12 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)..... 2201
35.5.13 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)..... 2202
35.5.14 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)..... 2202
35.5.15 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)..... 2203
35.5.16 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)..... 2203
35.5.17 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)..... 2204
35.5.18 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)..... 2204
35.5.19 Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)..... 2205
35.5.20 Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)..... 2205
35.5.21 Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)..... 2206
35.5.22 Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3)..... 2206
35.5.23 Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4)..... 2207
35.5.24 Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_ANA0)..... 2207
35.5.25 Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_ANA1)..... 2208
35.5.26 Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_ANA2)..... 2208
35.5.27 Value of OTP Bank2 Word0 (OTPMK Key) (OCOTP_OTPMK0)..... 2209
35.5.28 Value of OTP Bank2 Word1 (OTPMK Key) (OCOTP_OTPMK1)..... 2209
35.5.29 Value of OTP Bank2 Word2 (OTPMK Key) (OCOTP_OTPMK2)..... 2210
35.5.30 Value of OTP Bank2 Word3 (OTPMK Key) (OCOTP_OTPMK3)..... 2210

35.5.31 Value of OTP Bank2 Word4 (OTPMK Key) (OCOTP_OTPMK4)....2211
35.5.32 Value of OTP Bank2 Word5 (OTPMK Key) (OCOTP_OTPMK5)....2211
35.5.33 Value of OTP Bank2 Word6 (OTPMK Key) (OCOTP_OTPMK6)....2212
35.5.34 Value of OTP Bank2 Word7 (OTPMK Key) (OCOTP_OTPMK7)....2212
35.5.35 Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0).... 2213
35.5.36 Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1).... 2213
35.5.37 Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2).... 2214
35.5.38 Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3).... 2214
35.5.39 Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4).... 2215
35.5.40 Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5).... 2215
35.5.41 Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6).... 2216
35.5.42 Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7).... 2216
35.5.43 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0).... 2217
35.5.44 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1).... 2217
35.5.45 Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0).... 2218
35.5.46 Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1).... 2218
35.5.47 Value of OTP Bank4 Word4 (MAC Address) (OCOTP_RESERVED) (OCOTP_MAC).... 2218
35.5.48 Value of OTP Bank4 Word5 (CRC Key) (OCOTP_CRC)....2219
35.5.49 Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (OCOTP_GP1)....2219
35.5.50 Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (OCOTP_GP2)....2220
35.5.51 Value of OTP Bank5 Word0 (SW GP) (OCOTP_SW_GP0).... 2220
35.5.52 Value of OTP Bank5 Word1 (SW GP) (OCOTP_SW_GP1).... 2221
35.5.53 Value of OTP Bank5 Word2 (SW GP) (OCOTP_SW_GP2).... 2221
35.5.54 Value of OTP Bank5 Word3 (SW GP) (OCOTP_SW_GP3).... 2221
35.5.55 Value of OTP Bank5 Word4 (SW GP) (OCOTP_SW_GP4).... 2222
35.5.56 Value of OTP Bank5 Word5 (Misc Conf) (OCOTP_MISC_CONF).... 2222
35.5.57 Value of OTP Bank5 Word6 (Field Return) (OCOTP_FIELD_RETURN).... 2223
35.5.58 Value of OTP Bank5 Word7 (SRK Revoke) (OCOTP_SRK_REVOKE).... 2223
35.5.59 Value of OTP Bank6 Word0 (ROM Patch) (OCOTP_ROM_PATCH0).... 2224

35.5.60 Value of OTP Bank6 Word1 (ROM Patch) (OCOTP_ROM_PATCH1)..... 2224
35.5.61 Value of OTP Bank6 Word2 (ROM Patch) (OCOTP_ROM_PATCH2)..... 2224
35.5.62 Value of OTP Bank6 Word3 (ROM Patch) (OCOTP_ROM_PATCH3)..... 2225
35.5.63 Value of OTP Bank6 Word4 (ROM Patch) (OCOTP_ROM_PATCH4)..... 2225
35.5.64 Value of OTP Bank6 Word5 (ROM Patch) (OCOTP_ROM_PATCH5)..... 2226
35.5.65 Value of OTP Bank6 Word6 (ROM Patch) (OCOTP_ROM_PATCH6)..... 2226
35.5.66 Value of OTP Bank6 Word7 (ROM Patch) (OCOTP_ROM_PATCH7)..... 2227
35.5.67 Value of OTP Bank7 Word0 (ROM Patch) (OCOTP_ROM_PATCH8)..... 2227
35.5.68 Value of OTP Bank7 Word1 (ROM Patch) (OCOTP_ROM_PATCH9)..... 2227
35.5.69 Value of OTP Bank7 Word2 (ROM Patch) (OCOTP_ROM_PATCH10)..... 2228
35.5.70 Value of OTP Bank7 Word3 (ROM Patch) (OCOTP_ROM_PATCH11)..... 2228
35.5.71 Value of OTP Bank7 Word4 (ROM Patch) (OCOTP_ROM_PATCH12)..... 2229
35.5.72 Value of OTP Bank7 Word5 (ROM Patch) (OCOTP_ROM_PATCH13)..... 2229
35.5.73 Value of OTP Bank7 Word6 (ROM Patch) (OCOTP_ROM_PATCH14)..... 2230
35.5.74 Value of OTP Bank7 Word7 (ROM Patch) (OCOTP_ROM_PATCH15)..... 2230
35.5.75 Value of OTP Bank8 Word0 (General Purpose Customer Defined Info) (OCOTP_GP30).....2230
35.5.76 Value of OTP Bank8 Word1 (General Purpose Customer Defined Info) (OCOTP_GP31).....2231
35.5.77 Value of OTP Bank8 Word2 (General Purpose Customer Defined Info) (OCOTP_GP32).....2231
35.5.78 Value of OTP Bank8 Word3 (General Purpose Customer Defined Info) (OCOTP_GP33).....2232
35.5.79 Value of OTP Bank8 Word4 (General Purpose Customer Defined Info) (OCOTP_GP34).....2232
35.5.80 Value of OTP Bank8 Word5 (General Purpose Customer Defined Info) (OCOTP_GP35).....2233
35.5.81 Value of OTP Bank8 Word6 (General Purpose Customer Defined Info) (OCOTP_GP36).....2233
35.5.82 Value of OTP Bank8 Word7 (General Purpose Customer Defined Info) (OCOTP_GP37).....2233
35.5.83 Value of OTP Bank9 Word0 (General Purpose Customer Defined Info) (OCOTP_GP38).....2234
35.5.84 Value of OTP Bank9 Word1 (General Purpose Customer Defined Info) (OCOTP_GP39).....2234
35.5.85 Value of OTP Bank9 Word2 (General Purpose Customer Defined Info) (OCOTP_GP310).....2235
35.5.86 Value of OTP Bank9 Word3 (General Purpose Customer Defined Info) (OCOTP_GP311).....2235
35.5.87 Value of OTP Bank9 Word4 (General Purpose Customer Defined Info) (OCOTP_GP312).....2236
35.5.88 Value of OTP Bank9 Word5 (General Purpose Customer Defined Info) (OCOTP_GP313).....2236

35.5.89 Value of OTP Bank9 Word6 (General Purpose Customer Defined Info) (OCOTP_GP314).....2236
35.5.90 Value of OTP Bank9 Word7 (General Purpose Customer Defined Info) (OCOTP_GP315).....2237
35.5.91 Value of OTP Bank10 Word0 (General Purpose Customer Defined Info) (OCOTP_GP40).....2237
35.5.92 Value of OTP Bank10 Word1 (General Purpose Customer Defined Info) (OCOTP_GP41).....2238
35.5.93 Value of OTP Bank10 Word2 (General Purpose Customer Defined Info) (OCOTP_GP42).....2238
35.5.94 Value of OTP Bank10 Word3 (General Purpose Customer Defined Info) (OCOTP_GP43).....2239
35.5.95 Value of OTP Bank10 Word4 (General Purpose Customer Defined Info) (OCOTP_GP44).....2239
35.5.96 Value of OTP Bank10 Word5 (General Purpose Customer Defined Info) (OCOTP_GP45).....2239
35.5.97 Value of OTP Bank10 Word6 (General Purpose Customer Defined Info) (OCOTP_GP46).....2240
35.5.98 Value of OTP Bank10 Word7 (General Purpose Customer Defined Info) (OCOTP_GP47).....2240
35.5.99 Value of OTP Bank11 Word0 (General Purpose Customer Defined Info) (OCOTP_GP48).....2241
35.5.100 Value of OTP Bank11 Word1 (General Purpose Customer Defined Info) (OCOTP_GP49).....2241
35.5.101 Value of OTP Bank11 Word2 (General Purpose Customer Defined Info) (OCOTP_GP410).....2242
35.5.102 Value of OTP Bank11 Word3 (General Purpose Customer Defined Info) (OCOTP_GP411).....2242
35.5.103 Value of OTP Bank11 Word4 (General Purpose Customer Defined Info) (OCOTP_GP412).....2242
35.5.104 Value of OTP Bank11 Word5 (General Purpose Customer Defined Info) (OCOTP_GP413).....2243
35.5.105 Value of OTP Bank11 Word6 (General Purpose Customer Defined Info) (OCOTP_GP414).....2243
35.5.106 Value of OTP Bank11 Word7 (General Purpose Customer Defined Info) (OCOTP_GP415).....2244
35.5.107 Value of OTP Bank12 Word0 (General Purpose Customer Defined Info) (OCOTP_GP50).....2244
35.5.108 Value of OTP Bank12 Word1 (General Purpose Customer Defined Info) (OCOTP_GP51).....2245
35.5.109 Value of OTP Bank12 Word2 (General Purpose Customer Defined Info) (OCOTP_GP52).....2245
35.5.110 Value of OTP Bank12 Word3 (General Purpose Customer Defined Info) (OCOTP_GP53).....2245
35.5.111 Value of OTP Bank12 Word4 (General Purpose Customer Defined Info) (OCOTP_GP54).....2246
35.5.112 Value of OTP Bank12 Word5 (General Purpose Customer Defined Info) (OCOTP_GP55).....2246
35.5.113 Value of OTP Bank12 Word6 (General Purpose Customer Defined Info) (OCOTP_GP56).....2247
35.5.114 Value of OTP Bank12 Word7 (General Purpose Customer Defined Info) (OCOTP_GP57).....2247
35.5.115 Value of OTP Bank13 Word0 (General Purpose Customer Defined Info) (OCOTP_GP58).....2248
35.5.116 Value of OTP Bank13 Word1 (General Purpose Customer Defined Info) (OCOTP_GP59).....2248
35.5.117 Value of OTP Bank13 Word2 (General Purpose Customer Defined Info) (OCOTP_GP510).....2248

35.5.118 Value of OTP Bank13 Word3 (General Purpose Customer Defined Info) (OCOTP_GP511).....2249
35.5.119 Value of OTP Bank13 Word4 (General Purpose Customer Defined Info) (OCOTP_GP512).....2249
35.5.120 Value of OTP Bank13 Word5 (General Purpose Customer Defined Info) (OCOTP_GP513).....2250
35.5.121 Value of OTP Bank13 Word6 (General Purpose Customer Defined Info) (OCOTP_GP514).....2250
35.5.122 Value of OTP Bank13 Word7 (General Purpose Customer Defined Info) (OCOTP_GP515).....2251
35.5.123 Value of OTP Bank14 Word0 (General Purpose Customer Defined Info) (OCOTP_GP60).....2251
35.5.124 Value of OTP Bank14 Word1 (General Purpose Customer Defined Info) (OCOTP_GP61).....2251
35.5.125 Value of OTP Bank14 Word2 (General Purpose Customer Defined Info) (OCOTP_GP62).....2252
35.5.126 Value of OTP Bank14 Word3 (General Purpose Customer Defined Info) (OCOTP_GP63).....2252
35.5.127 Value of OTP Bank14 Word4 (General Purpose Customer Defined Info) (OCOTP_GP64).....2253
35.5.128 Value of OTP Bank14 Word5 (General Purpose Customer Defined Info) (OCOTP_GP65).....2253
35.5.129 Value of OTP Bank14 Word6 (General Purpose Customer Defined Info) (OCOTP_GP66).....2254
35.5.130 Value of OTP Bank14 Word7 (General Purpose Customer Defined Info) (OCOTP_GP67).....2254
35.5.131 Value of OTP Bank15 Word0 (General Purpose Customer Defined Info) (OCOTP_GP70).....2254
35.5.132 Value of OTP Bank15 Word1 (General Purpose Customer Defined Info) (OCOTP_GP71).....2255
35.5.133 Value of OTP Bank15 Word2 (General Purpose Customer Defined Info) (OCOTP_GP72).....2255
35.5.134 Value of OTP Bank15 Word3 (General Purpose Customer Defined Info) (OCOTP_GP73).....2256
35.5.135 Value of OTP Bank15 Word4 (General Purpose Customer Defined Info) (OCOTP_GP80).....2256
35.5.136 Value of OTP Bank15 Word5 (General Purpose Customer Defined Info) (OCOTP_GP81).....2257
35.5.137 Value of OTP Bank15 Word6 (General Purpose Customer Defined Info) (OCOTP_GP82).....2257
35.5.138 Value of OTP Bank15 Word7 (General Purpose Customer Defined Info) (OCOTP_GP83).....2257

Chapter 36
On-Chip RAM Memory Controller (OCRAM)

36.1 Overview.....2259
36.2 Basic Functions.....2260
36.2.1 Read/Write Arbitration..... 2260
36.3 Advanced Features.....2261
36.3.1 Read Data Wait State..... 2261
36.3.2 Read Address Pipeline..... 2261

36.3.3 Write Data Pipeline....2262
36.3.4 Write Address Pipeline....2262
36.4 Programmable Registers....2263

Chapter 37
Power Management Unit (PMU)

37.1 Overview....2265
37.2 Digital LDO Regulators....2267
37.3 Analog LDO Regulators....2268
37.3.1 LDO 1P1....2268
37.3.2 LDO 2P5....2268
37.3.3 Low Power Operation....2269
37.4 USB LDO Regulator....2270
37.5 SNVS Regulator....2270
37.6 PMU Memory Map/Register Definition....2270
37.6.1 Regulator 1P1 Register (PMU_REG_1P1)....2272
37.6.2 Regulator 3P0 Register (PMU_REG_3P0)....2275
37.6.3 Regulator 2P5 Register (PMU_REG_2P5)....2277
37.6.4 Digital Regulator Core Register (PMU_REG_CORE)....2279
37.6.5 Miscellaneous Register 0 (PMU_MISC0)....2281
37.6.6 Miscellaneous Register 1 (PMU_MISC1*n*)....2284
37.6.7 Miscellaneous Control Register (PMU_MISC2*n*)....2287
37.6.8 Low Power Control Register (PMU_LOWPWR_CTRL*n*)....2291

Chapter 38
Pulse Width Modulation (PWM)

38.1 Overview....2295
38.2 External Signals....2296
38.3 Clocks....2298
38.4 Functional Description....2299
38.4.1 Operation....2299

38.4.1.1 FIFO ... 2300
38.4.1.2 Rollover and Compare Event ... 2300
38.4.1.3 Low Power Mode Behavior ... 2300
38.4.1.4 Debug Mode Behavior ... 2301
38.5 Enable Sequence for the PWM ... 2301
38.6 Disable Sequence for the PWM ... 2301
38.7 PWM Memory Map/Register Definition ... 2302
38.7.1 PWM Control Register (PWM*x*_PWMCR) ... 2304
38.7.2 PWM Status Register (PWM*x*_PWMSR) ... 2306
38.7.3 PWM Interrupt Register (PWM*x*_PWMIR) ... 2307
38.7.4 PWM Sample Register (PWM*x*_PWMSAR) ... 2308
38.7.5 PWM Period Register (PWM*x*_PWMPR) ... 2309
38.7.6 PWM Counter Register (PWM*x*_PWMCNR) ... 2310

Chapter 39
Pixel Pipeline (PXP)

39.1 Overview ... 2311
39.2 Clocks ... 2312
39.3 Top-level architecture ... 2312
39.3.1 Processing Details ... 2314
39.3.2 Scaling Operation ... 2315
39.3.3 Decimation Image Scaling ... 2316
39.3.4 Bilinear Image Scaling Filter ... 2318
39.3.5 YUV 4:2:2 Image Scaling ... 2320
39.3.6 YUV 4:2:0 Image Scaling ... 2321
39.3.7 RGB/YUV444 Image Scaling ... 2323
39.3.8 Color Space Conversion (CSC) ... 2323
39.3.9 CSC1 Operation ... 2324
39.3.10 YUV versus YCbCr Support ... 2325
39.3.11 CSC2 operation ... 2325

39.3.12 Alpha Blending/Color Key 2325
39.3.13 Alpha Blend..... 2326
39.3.14 Color Key..... 2327
39.3.15 LUT..... 2327
39.3.16 Lookup Modes..... 2328
39.3.17 DIRECT_Y8..... 2328
39.3.18 DIRECT_RGB444..... 2328
39.3.19 DIRECT_RGB454..... 2328
39.3.20 CACHE_RGB565..... 2329
39.3.21 Output Modes..... 2330
39.3.22 Y8 2330
39.3.23 RGBW4444CFA..... 2331
39.3.23.1 CFA Correction..... 2331
39.3.24 RGB888..... 2332
39.3.25 Rotation..... 2332
39.3.26 Output Buffer..... 2335
39.3.27 Address calculator..... 2335
39.3.28 Block size selection..... 2335
39.3.29 Interlaced Video Support..... 2335
39.3.30 LCDIF Handshake..... 2336
39.3.31 LCDIF Abort..... 2339
39.3.32 Theory of Operation..... 2339
39.3.33 Pixel Handling..... 2340
39.3.34 Output Buffer Composition..... 2341
39.3.35 PS Image Processing..... 2341
39.3.36 Letterboxing..... 2342
39.3.37 Clipping source images..... 2342
39.3.38 Color Key Processing..... 2344
39.3.39 In Place Processing (PS buffer is destination buffer)..... 2346

39.3.40 Alpha Surface (AS) Processing....2346
39.3.41 Alpha Handling....2346
39.3.42 Color Key Processing (AS_CTRL)....2346
39.4 Output Image Processing....2347
39.4.1 Output Image Size....2347
39.4.2 Output Format....2347
39.4.3 Rotation/Flip operations....2347
39.5 Queuing PXP transactions....2348
39.6 Error Handling....2348
39.6.1 Known PXP Limitations/Issues....2349
39.7 PXP Memory Map/Register Definition....2349
39.7.1 Control Register 0 (PXP_CTRL)....2352
39.7.2 Status Register (PXP_STAT)....2354
39.7.3 Output Buffer Control Register (PXP_OUT_CTRL)....2356
39.7.4 Output Frame Buffer Pointer (PXP_OUT_BUF)....2358
39.7.5 Output Frame Buffer Pointer #2 (PXP_OUT_BUF2)....2359
39.7.6 Output Buffer Pitch (PXP_OUT_PITCH)....2359
39.7.7 Output Surface Lower Right Coordinate (PXP_OUT_LRC)....2360
39.7.8 Processed Surface Upper Left Coordinate (PXP_OUT_PS_ULC)....2361
39.7.9 Processed Surface Lower Right Coordinate (PXP_OUT_PS_LRC)....2362
39.7.10 Alpha Surface Upper Left Coordinate (PXP_OUT_AS_ULC)....2363
39.7.11 Alpha Surface Lower Right Coordinate (PXP_OUT_AS_LRC)....2364
39.7.12 Processed Surface (PS) Control Register (PXP_PS_CTRL)....2365
39.7.13 PS Input Buffer Address (PXP_PS_BUF)....2366
39.7.14 PS U/Cb or 2 Plane UV Input Buffer Address (PXP_PS_UBUF)....2367
39.7.15 PS V/Cr Input Buffer Address (PXP_PS_VBUF)....2368
39.7.16 Processed Surface Pitch (PXP_PS_PITCH)....2368
39.7.17 PS Background Color (PXP_PS_BACKGROUND)....2369
39.7.18 PS Scale Factor Register (PXP_PS_SCALE)....2370

39.7.19 PS Scale Offset Register (PXP_PS_OFFSET)..... 2371
39.7.20 PS Color Key Low (PXP_PS_CLRKEYLOW).....2372
39.7.21 PS Color Key High (PXP_PS_CLRKEYHIGH)..... 2373
39.7.22 Alpha Surface Control (PXP_AS_CTRL)..... 2373
39.7.23 Alpha Surface Buffer Pointer (PXP_AS_BUF).....2376
39.7.24 Alpha Surface Pitch (PXP_AS_PITCH).....2376
39.7.25 Overlay Color Key Low (PXP_AS_CLRKEYLOW)..... 2377
39.7.26 Overlay Color Key High (PXP_AS_CLRKEYHIGH).....2377
39.7.27 Color Space Conversion Coefficient Register 0 (PXP_CSC1_COEF0)..... 2378
39.7.28 Color Space Conversion Coefficient Register 1 (PXP_CSC1_COEF1)..... 2380
39.7.29 Color Space Conversion Coefficient Register 2 (PXP_CSC1_COEF2)..... 2380
39.7.30 Color Space Conversion Control Register. (PXP_CSC2_CTRL)..... 2381
39.7.31 Color Space Conversion Coefficient Register 0 (PXP_CSC2_COEF0)..... 2383
39.7.32 Color Space Conversion Coefficient Register 1 (PXP_CSC2_COEF1)..... 2383
39.7.33 Color Space Conversion Coefficient Register 2 (PXP_CSC2_COEF2)..... 2384
39.7.34 Color Space Conversion Coefficient Register 3 (PXP_CSC2_COEF3)..... 2384
39.7.35 Color Space Conversion Coefficient Register 4 (PXP_CSC2_COEF4)..... 2385
39.7.36 Color Space Conversion Coefficient Register 5 (PXP_CSC2_COEF5)..... 2386
39.7.37 Lookup Table Control Register. (PXP_LUT_CTRL).....2386
39.7.38 Lookup Table Control Register. (PXP_LUT_ADDR)..... 2388
39.7.39 Lookup Table Data Register. (PXP_LUT_DATA)..... 2390
39.7.40 Lookup Table External Memory Address Register. (PXP_LUT_EXTMEM)..... 2390
39.7.41 Color Filter Array Register. (PXP_CFA)..... 2391
39.7.42 Histogram Control Register. (PXP_HIST_CTRL)..... 2391
39.7.43 2-level Histogram Parameter Register. (PXP_HIST2_PARAM).....2392
39.7.44 4-level Histogram Parameter Register. (PXP_HIST4_PARAM).....2393
39.7.45 8-level Histogram Parameter 0 Register. (PXP_HIST8_PARAM0).....2394
39.7.46 8-level Histogram Parameter 1 Register. (PXP_HIST8_PARAM1).....2395
39.7.47 16-level Histogram Parameter 0 Register. (PXP_HIST16_PARAM0).....2396

39.7.48 16-level Histogram Parameter 1 Register. (PXP_HIST16_PARAM1)..... 2397
39.7.49 16-level Histogram Parameter 2 Register. (PXP_HIST16_PARAM2)..... 2398
39.7.50 16-level Histogram Parameter 3 Register. (PXP_HIST16_PARAM3)..... 2399
39.7.51 PXP Power Control Register. (PXP_POWER)..... 2399
39.7.52 Next Frame Pointer (PXP_NEXT)..... 2400

Chapter 40
Quad Serial Peripheral Interface (QuadSPI)

40.1 Overview..... 2403
40.1.1 Features..... 2405
40.1.2 QuadSPI Modes of Operation..... 2406
40.1.2.1 Normal Mode..... 2406
40.1.2.2 Module Disable Mode..... 2406
40.1.3 Acronyms and Abbreviations..... 2406
40.1.4 Glossary for QuadSPI module..... 2407
40.2 External Signals..... 2408
40.2.1 Driving External Signals..... 2409
40.3 Memory Map and Register Definition..... 2411
40.3.1 Register Write Access..... 2411
40.3.2 Serial Flash Address Assignment..... 2412
40.3.3 AMBA Bus Register Memory Map..... 2413
40.3.4 AHB Bus Register Memory Map Descriptions..... 2414
40.3.4.1 AHB Bus Access Considerations..... 2414
40.3.4.2 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A..... 2415
40.3.4.3 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B..... 2416
40.3.4.4 Parallel Flash Mode..... 2417
40.4 Interrupt Signals..... 2418
40.5 Functional Description..... 2418
40.5.1 Serial Flash Access Schemes..... 2418
40.5.2 Modes of Operation..... 2419

40.5.3 Normal Mode....2420
40.5.3.1 Programmable Sequence Engine....2420
40.5.3.2 Flexible AHB buffers....2422
40.5.3.3 Suspend-Abort Mechanism....2424
40.5.3.4 Look-up Table....2425
40.5.3.5 Issuing SFM Commands....2426
40.5.3.6 Flash Programming....2427
40.5.3.7 Flash Read....2428
40.5.3.8 Byte Ordering of Serial Flash Read Data....2432
40.5.3.9 Normal Mode Interrupt and DMA Requests....2435
40.5.3.10 TX Buffer Operation....2437
40.5.3.11 Address scheme....2437
40.6 Initialization/Application Information....2438
40.6.1 Power Up and Reset....2438
40.6.2 Available Status/Flag Information....2439
40.6.2.1 IP Commands....2439
40.6.2.2 AHB Commands....2439
40.6.2.3 Overview of Error Flags....2440
40.6.2.4 IP Bus and AHB Access Command Collisions....2441
40.6.3 Exclusive Access to Serial Flash for AHB Commands....2441
40.6.3.1 RX Buffer Read via QSPI_ARDB Registers....2442
40.6.3.2 RX Buffer Read via QSPI_RBDR Registers....2442
40.6.4 Command Arbitration....2442
40.6.5 Flash Device Selection....2443
40.6.6 DMA Usage....2444
40.6.6.1 DMA Usage in Normal Mode....2444
40.6.6.1.1 Bandwidth considerations....2444
40.6.7 Parallel mode....2446
40.7 Byte Ordering - Endianness....2448

40.7.1 Programming Flash Data ... 2449
40.7.2 Reading Flash Data into the RX Buffer ... 2450
40.7.2.1 Readout of the RX Buffer via QSPI_RBDRn ... 2450
40.7.2.2 Readout of the RX Buffer via ARDBn ... 2450
40.7.3 Reading Flash Data into the AHB Buffer ... 2451
40.7.3.1 Readout of the AHB Buffer via Memory Mapped Read ... 2451
40.8 Serial Flash Devices ... 2451
40.8.1 Example Sequences ... 2451
40.8.1.1 Fast Read Sequence (Macronix/Numonyx/Spansion/Winbond) ... 2452
40.8.1.2 Fast Dual I/O DT Read Sequence (Macronix) ... 2452
40.8.1.3 Fast Read Quad Output (Winbond) ... 2453
40.8.1.4 4 x I/O Read Enhance Performance Mode (XIP) (Macronix) ... 2453
40.8.1.5 Dual Command Page Program (Numonyx) ... 2454
40.8.1.6 Sector Erase (Macronix/Spansion/Numonyx) ... 2454
40.8.1.7 Read Status Register (Macronix/Spansion/Numonyx/Winbond) ... 2454
40.8.2 Dual Die Flashes ... 2455
40.8.3 Boot initialization sequence ... 2455
40.9 Sampling of Serial Flash Input Data ... 2456
40.9.1 Internal Sampling of Serial Flash Input Data ... 2456
40.9.2 DDR Mode ... 2459
40.10 Serial Flash Data Input Timing ... 2459
40.10.1 Input timing in SDR mode with internal sampling ... 2461
40.10.2 Input timing in DDR mode with internal sampling ... 2461
40.10.3 Input timing in SDR mode with loopback DQS sampling ... 2462
40.10.4 Input timing in DDR mode with loopback DQS sampling ... 2463
40.10.5 Input timing in SDR mode with flash DQS sampling ... 2464
40.10.6 Input timing in DDR mode with flash DQS sampling ... 2465
40.10.7 Data Strobe Signal functionality ... 2466
40.11 Output timing in SDR mode ... 2466

40.12 Output timing in DDR mode 2467
40.13 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) 2468
40.13.1 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) 2468
40.13.1.1 AHB RX Data Buffer register (ARDB*n*) 2468
40.14 Peripheral Bus Register Descriptions 2470
40.14.1 Module Configuration Register (QuadSPI_MCR) 2476
40.14.2 IP Configuration Register (QuadSPI_IPCR) 2479
40.14.3 Flash Configuration Register (QuadSPI_FLSHCR) 2480
40.14.4 Buffer0 Configuration Register (QuadSPI_BUF0CR) 2480
40.14.5 Buffer1 Configuration Register (QuadSPI_BUF1CR) 2481
40.14.6 Buffer2 Configuration Register (QuadSPI_BUF2CR) 2482
40.14.7 Buffer3 Configuration Register (QuadSPI_BUF3CR) 2483
40.14.8 Buffer Generic Configuration Register (QuadSPI_BFGENCR) 2484
40.14.9 Buffer0 Top Index Register (QuadSPI_BUF0IND) 2484
40.14.10 Buffer1 Top Index Register (QuadSPI_BUF1IND) 2485
40.14.11 Buffer2 Top Index Register (QuadSPI_BUF2IND) 2486
40.14.12 Serial Flash Address Register (QuadSPI_SFAR) 2487
40.14.13 Sampling Register (QuadSPI_SMPR) 2487
40.14.14 RX Buffer Status Register (QuadSPI_RBSR) 2488
40.14.15 RX Buffer Control Register (QuadSPI_RBCT) 2489
40.14.16 TX Buffer Status Register (QuadSPI_TBSR) 2490
40.14.17 TX Buffer Data Register (QuadSPI_TBDR) 2490
40.14.18 Status Register (QuadSPI_SR) 2492
40.14.19 Flag Register (QuadSPI_FR) 2495
40.14.20 Interrupt and DMA Request Select and Enable Register (QuadSPI_RSER) 2498
40.14.21 Sequence Suspend Status Register (QuadSPI_SPNDST) 2501
40.14.22 Sequence Pointer Clear Register (QuadSPI_SPTRCLR) 2503
40.14.23 Serial Flash A1 Top Address (QuadSPI_SFA1AD) 2503
40.14.24 Serial Flash A2 Top Address (QuadSPI_SFA2AD) 2504

40.14.25 Serial Flash B1Top Address (QuadSPI_SFB1AD)..... 2504
40.14.26 Serial Flash B2Top Address (QuadSPI_SFB2AD)..... 2505
40.14.27 RX Buffer Data Register (QuadSPI_RBDR*n*)..... 2505
40.14.28 LUT Key Register (QuadSPI_LUTKEY)..... 2506
40.14.29 LUT Lock Configuration Register (QuadSPI_LCKCR)..... 2507
40.14.30 Look-up Table register (QuadSPI_LUT0)..... 2508
40.14.31 Look-up Table register (QuadSPI_LUT1)..... 2509
40.14.32 Look-up Table register (QuadSPI_LUT*n*)..... 2510

Chapter 41
ROM Controller with Patch (ROMC)

41.1 Overview..... 2511
41.1.1 Features..... 2512
41.1.2 Modes of Operation..... 2512
41.1.2.1 Low Power Mode..... 2513
41.2 Clocks..... 2513
41.3 Memory Map..... 2513
41.3.1 ROM Memory Map in detail..... 2513
41.4 Functional Description..... 2514
41.4.1 ROM Controller (ROMC) Functional Description..... 2514
41.4.1.1 Functionality overview..... 2514
41.4.2 ROMC Functional Description..... 2514
41.4.2.1 ROMC Disabling..... 2515
41.4.2.2 ROMC Event Priority..... 2515
41.4.2.3 Data Fixing..... 2515
41.4.2.4 Opcode Patching..... 2516
41.4.2.4.1 Typical Software Response to Opcode Patch..... 2517
41.4.2.5 External Boot Feature..... 2518
41.4.2.6 Alternate Masters and ROMC..... 2519
41.5 ROMCP Memory Map/Register Definition..... 2519

41.5.1 ROMC Data Registers (ROMC_ROMPATCH*n*D) 2520
41.5.2 ROMC Control Register (ROMC_ROMPATCHCNTL) 2521
41.5.3 ROMC Enable Register High (ROMC_ROMPATCHENH) 2522
41.5.4 ROMC Enable Register Low (ROMC_ROMPATCHENL) 2522
41.5.5 ROMC Address Registers (ROMC_ROMPATCH*n*A) 2523
41.5.6 ROMC Status Register (ROMC_ROMPATCHSR) 2524

Chapter 42
Synchronous Audio Interface (SAI)

42.1 Overview 2527
42.1.1 Features 2527
42.1.2 Block diagram 2527
42.1.3 Modes of operation 2528
42.1.3.1 Run mode 2528
42.1.3.2 Stop modes 2528
42.1.3.3 Debug mode 2528
42.2 External Signals 2529
42.3 Functional description 2531
42.3.1 SAI clocking 2531
42.3.1.1 Audio master clock 2531
42.3.1.2 Bit clock 2531
42.3.1.3 Bus clock 2532
42.3.2 SAI resets 2532
42.3.2.1 Software reset 2532
42.3.2.2 FIFO reset 2532
42.3.3 Synchronous modes 2533
42.3.3.1 Synchronous mode 2533
42.3.4 Frame sync configuration 2533
42.3.5 Data FIFO 2534
42.3.5.1 Data alignment 2534

42.3.5.2 FIFO pointers ... 2535
42.3.6 Word mask register ... 2536
42.3.7 Interrupts and DMA requests ... 2536
42.3.7.1 FIFO request flag ... 2536
42.3.7.2 FIFO warning flag ... 2537
42.3.7.3 FIFO error flag ... 2537
42.3.7.4 Sync error flag ... 2537
42.3.7.5 Word start flag ... 2538
42.4 Memory map and register definition ... 2538
42.4.1 SAI Transmit Control Register (I2S*x*_TCSR) ... 2541
42.4.2 SAI Transmit Configuration 1 Register (I2S*x*_TCR1) ... 2544
42.4.3 SAI Transmit Configuration 2 Register (I2S*x*_TCR2) ... 2544
42.4.4 SAI Transmit Configuration 3 Register (I2S*x*_TCR3) ... 2546
42.4.5 SAI Transmit Configuration 4 Register (I2S*x*_TCR4) ... 2547
42.4.6 SAI Transmit Configuration 5 Register (I2S*x*_TCR5) ... 2548
42.4.7 SAI Transmit Data Register (I2S*x*_TDR*n*) ... 2549
42.4.8 SAI Transmit FIFO Register (I2S*x*_TFR*n*) ... 2550
42.4.9 SAI Transmit Mask Register (I2S*x*_TMR) ... 2550
42.4.10 SAI Receive Control Register (I2S*x*_RCSR) ... 2551
42.4.11 SAI Receive Configuration 1 Register (I2S*x*_RCR1) ... 2554
42.4.12 SAI Receive Configuration 2 Register (I2S*x*_RCR2) ... 2555
42.4.13 SAI Receive Configuration 3 Register (I2S*x*_RCR3) ... 2556
42.4.14 SAI Receive Configuration 4 Register (I2S*x*_RCR4) ... 2557
42.4.15 SAI Receive Configuration 5 Register (I2S*x*_RCR5) ... 2559
42.4.16 SAI Receive Data Register (I2S*x*_RDR*n*) ... 2559
42.4.17 SAI Receive FIFO Register (I2S*x*_RFR*n*) ... 2560
42.4.18 SAI Receive Mask Register (I2S*x*_RMR) ... 2560

Chapter 43
Smart Direct Memory Access Controller (SDMA)

43.1 Overview....2563
43.1.1 Block Diagram....2563
43.1.2 Features....2565
43.2 External Signals....2567
43.3 Clocks....2567
43.4 Functional Description....2567
43.4.1 SDMA Core....2568
43.4.1.1 SDMA Core Structure....2569
43.4.1.2 Program Control Unit (PCU)....2572
43.4.1.2.1 Instruction Types....2572
43.4.1.2.2 PCU States....2573
43.4.1.3 SDMA Core Memory....2576
43.4.2 Scheduler....2576
43.4.2.1 Primary Functions....2576
43.4.2.2 Channels and DMA Requests....2577
43.4.2.2.1 Channels....2577
43.4.2.2.2 DMA Requests....2577
43.4.2.2.3 Mapping from DMA Requests to Channels and Priorities....2577
43.4.2.3 Scheduler Functional Description....2577
43.4.2.3.1 Scheduler Overview....2577
43.4.2.3.2 DMA Requests Scanning....2578
43.4.2.3.3 Mapping DMA Requests to Pending Channels....2579
43.4.2.3.4 Channel Overflow....2582
43.4.2.3.5 Runnable Channels Evaluation....2582
43.4.2.3.6 Next Channel Decision Tree....2584
43.4.2.3.7 Scheduler State Diagram....2586
43.4.2.3.8 Scheduler Pipeline Timing Diagram....2588
43.4.2.3.9 Channel-DMA Request Mapping....2588
43.4.2.3.10 Examples: How to Start a Channel....2588

43.4.2.4 Context Switching....2589
43.4.2.4.1 Context Switch Modes....2590
43.4.2.4.2 Context Switch Procedure....2590
43.4.2.4.3 Context Map in Memory....2592
43.4.3 Functional Units....2592
43.4.3.1 Burst DMA Unit....2592
43.4.3.1.1 Burst DMA Structure....2593
43.4.3.1.2 Burst DMA Registers....2594
43.4.3.1.3 Burst DMA Data Transfers....2595
43.4.3.1.3.1 Data Retrieval from the ARM platform Memory....2595
43.4.3.1.3.2 Storing Data Into the ARM platform Memory....2595
43.4.3.1.3.3 Transferring Data Between Two ARM platform Memory Locations-Burst DMA Unit....2596
43.4.3.2 Peripheral DMA Unit....2596
43.4.3.2.1 Peripheral DMA Structure....2597
43.4.3.2.2 Peripheral DMA Registers....2598
43.4.3.2.3 Peripheral DMA Data Transfers....2599
43.4.3.2.3.1 Data Retrieval from the ARM platform Memory or Peripheral....2599
43.4.3.2.3.2 Storing Data into the ARM platform Memory or Peripheral. 2599
43.4.3.2.3.3 Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit....2600
43.4.4 SDMA Security Support....2600
43.4.4.1 Locked Mode....2600
43.4.5 OnCE and PCU Debug States....2601
43.4.6 SDMA Clocks and Low Power Modes....2603
43.4.6.1 Clock Gating and Low Power Modes....2604
43.4.6.1.1 Coarse Clock Gating....2604
43.4.6.1.2 Refined Clock Gating....2605
43.4.6.1.3 Low Power Modes and User Control....2605

43.4.6.1.3.1 SLEEP Mode....2606
43.4.6.1.3.2 RUN Mode....2606
43.4.6.1.3.3 DEBUG Mode....2607
43.4.6.1.4 Stop Mode Response....2607
43.4.6.2 Reset....2607
43.4.7 Software Interface....2607
43.4.8 Initialization Information....2608
43.4.8.1 Hardware Reset....2608
43.4.8.2 Channel Script Execution....2609
43.4.8.3 Initialization and Script Execution Setup Sequence....2609
43.4.9 SDMA Programming Model....2610
43.4.9.1 State and Registers Per Channel....2610
43.4.9.2 General Purpose Registers....2611
43.4.9.3 Functional Unit State....2611
43.4.9.3.1 Program Counter Register (PC)....2611
43.4.9.3.2 Flags....2611
43.4.9.3.3 Return Program Counter (RPC)....2612
43.4.9.3.4 Loop Mode Start Program Counter (SPC)....2612
43.4.9.3.5 Loop Mode End Program Counter (EPC)....2612
43.4.9.4 Context Switching-Programming....2613
43.4.9.5 Address Space....2614
43.4.9.5.1 Instruction Memory Map....2615
43.4.9.5.2 Data Memory Map....2615
43.4.10 SDMA Initialization....2617
43.4.10.1 Hardware Reset-SDMA....2617
43.4.10.2 Standard Boot Sequence....2617
43.4.10.3 User-Defined Boot Sequence....2618
43.4.10.4 Script Loading and Context Initialization....2618
43.4.11 Instruction Description....2619

43.4.11.1 Scheduling Instructions....2619
43.4.11.2 Conditional Branch Instructions....2619
43.4.11.3 Unconditional Jump Instructions....2620
43.4.11.4 Subroutine Return Instructions....2620
43.4.11.5 Loop Instruction....2620
43.4.11.6 Miscellaneous Instructions....2621
43.4.11.7 Logic Instructions....2621
43.4.11.8 Arithmetic Instructions....2621
43.4.11.9 Compare Instructions....2622
43.4.11.10 Test Instructions....2622
43.4.11.11 Byte Permutation Instructions....2622
43.4.11.12 Bit Shift Instructions....2623
43.4.11.13 Bit Manipulation Instructions....2623
43.4.11.14 SDMA Memory Access Instructions....2623
43.4.11.15 Functional Unit Instructions....2624
43.4.11.16 Illegal Instructions....2624
43.4.11.17 Debug Instructions....2624
43.4.12 Functional Units Programming Model....2625
43.4.12.1 Burst DMA Unit Programming....2626
43.4.12.1.1 Memory Source Address Register (MSA)....2626
43.4.12.1.2 Memory Destination Address Register (MDA)....2627
43.4.12.1.3 Memory Data Buffer Register (MD)....2627
43.4.12.1.4 State Register (MS)....2628
43.4.12.1.5 Burst DMA Write (stf)....2629
43.4.12.1.6 Burst DMA Read (ldf)....2632
43.4.12.1.7 Prefetch/Flush and Auto-Flush Management-Burst DMA Unit....2633
43.4.12.1.8 Data Alignment and Endianness-Burst DMA Unit....2635
43.4.12.1.8.1 Burst DMA in Read Mode....2635
43.4.12.1.8.2 Burst DMA in Write Mode....2636

43.4.12.1.8.3 Endianness-Burst DMA Unit ... 2637
43.4.12.1.9 Burst DMA Unit Copy Mode ... 2638
43.4.12.1.10 Burst DMA Unit Error Management ... 2639
43.4.12.1.11 Conditional Yielding-Burst DMA Unit ... 2640
43.4.12.2 Peripheral DMA Unit Programming ... 2641
43.4.12.2.1 Peripheral Source Address Register (PSA) ... 2642
43.4.12.2.2 Peripheral Destination Address Register (PDA) ... 2643
43.4.12.2.3 Peripheral Data Register (PD) ... 2643
43.4.12.2.4 Peripheral State Register (PS) ... 2644
43.4.12.2.5 Peripheral DMA Write (stf)-Write Mode ... 2645
43.4.12.2.6 Peripheral DMA Read (ldf)-Read Mode ... 2648
43.4.12.2.7 Peripheral DMA Unit Copy Mode ... 2649
43.4.12.2.8 Error Management ... 2650
43.4.12.2.8.1 Immediate Errors ... 2650
43.4.12.2.8.2 Data Transfer Errors ... 2650
43.4.12.2.8.3 Read Error (First Phase) ... 2651
43.4.12.2.8.4 Write Error and Read Error (Second Phase) ... 2651
43.4.12.2.8.5 Copy Mode Errors ... 2652
43.4.12.2.8.6 Error Check Example ... 2652
43.4.12.2.9 Peripheral DMA Unit Prefetch/Flush Management ... 2653
43.4.12.3 OnCE and Real-Time Debug ... 2653
43.4.12.3.1 Memory and Register Access ... 2653
43.4.12.3.2 Hardware Breakpoints ... 2654
43.4.12.3.3 Watchpoints ... 2654
43.4.12.3.4 Software Breakpoints ... 2654
43.4.12.3.5 Core Control ... 2654
43.4.13 The OnCE Controller ... 2654
43.4.13.1 OnCE Commands ... 2655
43.4.13.2 Sending Commands to the OnCE Controller ... 2656

43.4.13.2.1 Using the JTAG Interface..... 2656
43.4.13.2.2 Using the ARM platform..... 2656
43.4.13.2.3 Conflicts Between the JTAG and the ARM platform Accesses..... 2658
43.4.13.3 Executing a Command from the OnCE..... 2658
43.4.13.3.1 Nature of the Commands..... 2658
43.4.13.3.2 Execution Request..... 2659
43.4.13.3.3 Command Execution..... 2659
43.4.13.4 Registers Descriptions..... 2661
43.4.13.4.1 Event Cell Counter Register (ECOUNT)..... 2661
43.4.13.4.2 Event Cell Address Registers (EAA or EAB)..... 2662
43.4.13.4.3 Event Cell Address Mask Register (EAM)..... 2662
43.4.13.4.4 Event Cell Data Register (ED)..... 2662
43.4.13.4.5 Event Cell Data Mask Register (EDM)..... 2662
43.4.13.4.6 Real Time Buffer Register (RTB)..... 2662
43.4.13.4.7 Event Control Register (ECTL)..... 2663
43.4.13.4.8 Trace Buffer (TB)..... 2663
43.4.13.4.9 OnCE Status Register (OSTAT)..... 2663
43.4.13.5 JTAG Interface Requirements..... 2664
43.4.13.5.1 TCK Speed Limitation..... 2664
43.4.13.5.2 Synchronization Implementation..... 2664
43.4.13.5.3 JTAG Controller Start-Up Recommended Procedure..... 2666
43.4.14 Using the OnCE..... 2666
43.4.14.1 Activating Clocks in Debug Mode..... 2666
43.4.14.2 Getting the Current Status..... 2667
43.4.14.3 Methods of Entering Debug Mode..... 2667
43.4.14.3.1 External Debug Request During Reset..... 2667
43.4.14.3.2 Debug Request During Normal Activity..... 2668
43.4.14.3.3 Software Breakpoint Instruction..... 2668
43.4.14.3.4 Event Detection Unit Matching Condition..... 2668

43.4.14.4 Executing Instructions in Debug Mode ... 2668
43.4.14.5 Command Sequences Examples ... 2668
43.4.14.5.1 Getting the SDMA Status ... 2669
43.4.14.5.2 Saving the Context ... 2669
43.4.14.5.3 Restoring the Context ... 2670
43.4.14.5.4 Accessing the Memory ... 2671
43.4.14.5.5 Resuming Program Execution ... 2672
43.4.14.5.6 Single Stepping in RAM ... 2673
43.4.14.5.7 Single Stepping in ROM ... 2673
43.4.14.6 OnCE Event Detection Unit ... 2673
43.4.14.7 Clock Gating and Reset ... 2675
43.4.14.7.1 Clocks ... 2675
43.4.14.7.2 Resets ... 2676
43.4.14.8 Real Time Features ... 2676
43.4.14.8.1 Trace Buffer ... 2676
43.4.14.8.2 Real Time Buffer ... 2678
43.4.14.8.3 Emulation Pin ... 2678
43.4.14.8.4 Real-Time Debug Outputs ... 2678
43.5 Instruction Set ... 2682
43.5.1 Instruction Encoding ... 2682
43.5.2 SDMA Instruction Set ... 2684
43.5.2.1 ADD (Addition) ... 2685
43.5.2.2 ADDI (Add with Immediate Value) ... 2686
43.5.2.3 AND (Logical AND) ... 2687
43.5.2.4 ANDI (Logical AND with Immediate Value) ... 2688
43.5.2.5 ANDN (Logical AND NOT) ... 2689
43.5.2.6 ANDNI (Logical AND with Negated Immediate Value) ... 2690
43.5.2.7 ASR1 (Arithmetic Shift Right by 1 Bit) ... 2691
43.5.2.8 BCLRI1 (Bit Clear Immediate) ... 2692

43.5.2.9 BDF (Conditional Branch if Destination Fault)....2693
43.5.2.10 BF (Conditional Branch if False)....2694
43.5.2.11 BSETI (Bit Set Immediate)....2695
43.5.2.12 BSF (Conditional Branch if Source Fault)....2696
43.5.2.13 BT (Conditional Branch if True)....2697
43.5.2.14 BTSTI (Bit Test immediate)....2698
43.5.2.15 CLRF (Clear ARM platform flags)....2699
43.5.2.16 CMPEQ (Compare for Equal)....2700
43.5.2.17 CMPEQI (Compare with Immediate for Equal)....2701
43.5.2.18 CMPHS (Compare for Higher or Same)....2702
43.5.2.19 CMPLT (Compare for Less Than)....2703
43.5.2.20 cpShReg (Update Context of PCU Registers and Flag)....2704
43.5.2.21 DONE (DONE, Yield)....2704
43.5.2.22 ILLEGAL (ILLEGAL Instruction)....2706
43.5.2.23 JMP (Unconditional Jump Immediate)....2707
43.5.2.24 JMPR (Unconditional Jump)....2707
43.5.2.25 JSR (Unconditional Jump to Subroutine Immediate)....2708
43.5.2.26 JSRR (Unconditional Jump to Subroutine)....2709
43.5.2.27 LD (Load Register)....2710
43.5.2.28 LDF (Load Register from Functional Unit)....2711
43.5.2.29 LDI (Load Register with Immediate Value)....2713
43.5.2.30 LDRPC (Load from RPC to Register)....2714
43.5.2.31 LOOP (Hardware Loop)....2715
43.5.2.32 LSL1 (Logical Shift Left by 1 Bit)....2717
43.5.2.33 LSR1 (Logical Shift Right by 1 Bit)....2718
43.5.2.34 MOV (Logical Move)....2719
43.5.2.35 NOTIFY (Notify to ARM platform)....2720
43.5.2.36 OR (Logical OR)....2721
43.5.2.37 ORI (Logical OR with Immediate Value)....2722

43.5.2.38 RET (Return from Subroutine)..... 2723
43.5.2.39 REVB (Reverse Byte Order).....2724
43.5.2.40 Reverse Low Order Bytes(REVBLO)..... 2724
43.5.2.41 ROR1 (Rotate Right by 1 Bit).....2725
43.5.2.42 RORB (Rotate Right by 1 Byte)..... 2726
43.5.2.43 SOFTBKPT (Software Breakpoint).....2727
43.5.2.44 ST (Store Register).....2727
43.5.2.45 STF (Store Register in Functional Unit)..... 2729
43.5.2.46 SUB (Subtract)..... 2732
43.5.2.47 SUBI (Subtract with Immediate)..... 2733
43.5.2.48 TST (Test with Zero)..... 2734
43.5.2.49 TSTI (Test Immediate).....2735
43.5.2.50 XOR (Logical Exclusive OR)..... 2736
43.5.2.51 XORI (Exclusive OR with Immediate).....2737
43.5.2.52 YIELD, YIELDGE (DONE, Yield).....2738
43.6 Software Restrictions.....2738
43.6.1 Unsupported Burst DMA Access Sequence..... 2738
43.7 Application Notes..... 2739
43.7.1 Data Structures for Boot Code and Channel Scripts.....2739
43.7.1.1 Buffer Descriptor Format.....2740
43.7.1.2 Buffer Descriptor Commands for Bootload scripts..... 2743
43.7.1.3 Example of Buffer Descriptors for Channel 0..... 2744
43.7.1.4 Channel Context.....2747
43.7.2 Typical Data Transfer Supported by SDMA DMA Units..... 2747
43.7.2.1 External Memory to External Memory.....2748
43.7.2.2 Peripheral to Peripheral Transfer..... 2749
43.7.2.2.1 Source and Destination Target Have the Same Data Path Width.....2749
43.7.2.2.2 Source and Destination Target Have a Different Data Path Width..... 2750
43.7.2.3 Transfer Between Peripheral and External Memory.....2751

43.7.2.3.1 Peripheral to External Memory Transfer 2751
43.7.2.3.2 External Memory to Peripheral Transfer 2753
43.7.2.4 Transfer Between External Memory and Internal Memory 2754
43.7.2.4.1 Internal Memory to Internal Memory 2754
43.7.2.4.2 Transfer Between Peripheral and Internal Memory 2754
43.8 ARM Platform Memory Map and Control Register Definitions 2754
43.8.1 ARM platform Channel 0 Pointer (SDMAARM_MC0PTR) 2760
43.8.2 Channel Interrupts (SDMAARM_INTR) 2760
43.8.3 Channel Stop/Channel Status (SDMAARM_STOP_STAT) 2760
43.8.4 Channel Start (SDMAARM_HSTART) 2761
43.8.5 Channel Event Override (SDMAARM_EVTOVR) 2761
43.8.6 Channel BP Override (SDMAARM_DSPOVR) 2762
43.8.7 Channel ARM platform Override (SDMAARM_HOSTOVR) 2762
43.8.8 Channel Event Pending (SDMAARM_EVTPEND) 2762
43.8.9 Reset Register (SDMAARM_RESET) 2763
43.8.10 DMA Request Error Register (SDMAARM_EVTERR) 2764
43.8.11 Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK) 2764
43.8.12 Schedule Status (SDMAARM_PSW) 2765
43.8.13 DMA Request Error Register (SDMAARM_EVTERRDBG) 2765
43.8.14 Configuration Register (SDMAARM_CONFIG) 2766
43.8.15 SDMA LOCK (SDMAARM_SDMA_LOCK) 2767
43.8.16 OnCE Enable (SDMAARM_ONCE_ENB) 2768
43.8.17 OnCE Data Register (SDMAARM_ONCE_DATA) 2769
43.8.18 OnCE Instruction Register (SDMAARM_ONCE_INSTR) 2769
43.8.19 OnCE Status Register (SDMAARM_ONCE_STAT) 2769
43.8.20 OnCE Command Register (SDMAARM_ONCE_CMD) 2771
43.8.21 Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR) 2772
43.8.22 Channel 0 Boot Address (SDMAARM_CHN0ADDR) 2772
43.8.23 DMA Requests (SDMAARM_EVT_MIRROR) 2773

43.8.24 DMA Requests 2 (SDMAARM_EVT_MIRROR2)..... 2773
43.8.25 Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)..... 2774
43.8.26 Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)..... 2776
43.8.27 Channel Priority Registers (SDMAARM_SDMA_CHNPRI*n*)..... 2777
43.8.28 Channel Enable RAM (SDMAARM_CHNENBL*n*)..... 2777
43.9 BP Memory Map and Control Register Definitions..... 2778
43.9.1 Channel 0 Pointer (SDMABP_DC0PTR)..... 2778
43.9.2 Channel Interrupts (SDMABP_INTR)..... 2779
43.9.3 Channel Stop/Channel Status (SDMABP_STOP_STAT)..... 2779
43.9.4 Channel Start (SDMABP_DSTART)..... 2780
43.9.5 DMA Request Error Register (SDMABP_EVTERR)..... 2780
43.9.6 Channel DSP Interrupt Mask (SDMABP_INTRMASK)..... 2781
43.9.7 DMA Request Error Register (SDMABP_EVTERRDBG)..... 2781
43.10 SDMA Internal (Core) Memory Map and Internal Register Definitions..... 2781
43.10.1 ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)..... 2783
43.10.2 Current Channel Pointer (SDMACORE_CCPTR)..... 2783
43.10.3 Current Channel Register (SDMACORE_CCR)..... 2783
43.10.4 Highest Pending Channel Register (SDMACORE_NCR)..... 2784
43.10.5 External DMA Requests Mirror (SDMACORE_EVENTS)..... 2785
43.10.6 Current Channel Priority (SDMACORE_CCPRI)..... 2786
43.10.7 Next Channel Priority (SDMACORE_NCPRI)..... 2786
43.10.8 OnCE Event Cell Counter (SDMACORE_ECOUNT)..... 2787
43.10.9 OnCE Event Cell Control Register (SDMACORE_ECTL)..... 2787
43.10.10 OnCE Event Address Register A (SDMACORE_EAA)..... 2789
43.10.11 OnCE Event Cell Address Register B (SDMACORE_EAB)..... 2789
43.10.12 OnCE Event Cell Address Mask (SDMACORE_EAM)..... 2789
43.10.13 OnCE Event Cell Data Register (SDMACORE_ED)..... 2790
43.10.14 OnCE Event Cell Data Mask (SDMACORE_EDM)..... 2790
43.10.15 OnCE Real-Time Buffer (SDMACORE_RTB)..... 2791

43.10.16 OnCE Trace Buffer (SDMACORE_TB)..... 2791
43.10.17 OnCE Status (SDMACORE_OSTAT)..... 2792
43.10.18 Channel 0 Boot Address (SDMACORE_MCHN0ADDR)..... 2794
43.10.19 ENDIAN Status Register (SDMACORE_ENDIANNESS)..... 2795
43.10.20 Lock Status Register (SDMACORE_SDMA_LOCK)..... 2796
43.10.21 External DMA Requests Mirror #2 (SDMACORE_EVENTS2)..... 2797
43.11 SDMA Peripheral Registers.....2797

Chapter 44
Subscriber Identification Module (SIM)

44.1 SIM Overview.....2799
44.1.1 Features..... 2800
44.1.2 Modes of Operation..... 2800
44.1.3 SIM Bus Interface Overview..... 2800
44.1.4 SIM Clock Generator Overview..... 2801
44.1.5 SIM Transmitter Overview..... 2801
44.1.6 SIM Receiver Overview.....2802
44.1.7 SIM Port Control Overview.....2803
44.1.8 SIM General Purpose Counter Overview..... 2803
44.1.9 SIM LRC Block Overview..... 2803
44.1.10 SIM CRC Block Overview..... 2804
44.2 External Signal Description..... 2804
44.2.1 External Signals Overview.....2804
44.2.2 Detailed Signal Descriptions.....2805
44.2.2.1 SIM_CLK.....2805
44.2.2.2 SIM_RST_B.....2805
44.2.2.3 SIM_SVEN..... 2805
44.2.2.4 SIM_TRXD.....2805
44.2.2.5 SIM_PD..... 2806
44.3 SIM Functional Description.....2806

44.3.1 SIM Bus Interface ... 2807
44.3.2 SIM Clocking ... 2810
44.3.3 SIM Clock Generator ... 2811
44.3.3.1 Scan Test ... 2812
44.3.3.2 Baud Clock Generation ... 2812
44.3.3.3 Transmitter Clock Generation ... 2813
44.3.3.4 Receiver Clock Generation ... 2813
44.3.3.5 Port Control Clock Generation ... 2813
44.3.3.6 Low Power Mode Clock Control ... 2813
44.3.4 SIM Transmitter ... 2814
44.3.4.1 Transmit State Machine ... 2814
44.3.4.2 Transmit Shift Register ... 2816
44.3.4.3 Transmit FIFO ... 2816
44.3.4.4 Transmit Guard Time Generator ... 2816
44.3.4.5 Transmit NACK Generator ... 2817
44.3.4.6 Transmit Data Convention Logic ... 2818
44.3.5 SIM Receiver ... 2819
44.3.5.1 Receive State Machine ... 2819
44.3.5.2 Data Sampling / Voting ... 2821
44.3.5.3 Start Bit Detection ... 2821
44.3.5.4 Parity Error Detection ... 2822
44.3.5.5 Framing Error Detection ... 2822
44.3.5.6 NACK Detection ... 2823
44.3.5.7 Initial Character Detection ... 2823
44.3.5.8 Receive FIFO ... 2824
44.3.5.9 Overrun Detection ... 2825
44.3.5.10 Character Wait Time Counter ... 2825
44.3.6 SIM Port Control ... 2826
44.3.6.1 SIM Card Interface ... 2826

44.3.6.2 SIM Card Presence Detect ... 2826
44.3.6.3 SIM Card Automatic Power Down ... 2826
44.3.7 SIM General Purpose Counter ... 2827
44.3.8 SIM LRC Block ... 2828
44.3.9 SIM CRC Block ... 2829
44.3.10 Module Interrupts ... 2831
44.4 Initialization/Application Information ... 2831
44.4.1 Configuring SIM for Operation ... 2832
44.4.1.1 Configuring SIM Receiver ... 2832
44.4.1.2 Configuring SIM Transmitter ... 2833
44.4.1.3 Configuring SIM General Purpose Counter ... 2834
44.4.1.4 Configuring SIM to Measure WWT (Work Wait Time) for Type=0 Smartcards ... 2834
44.4.1.5 Configuring SIM to measure CWT, BWT, BGT for type=1 SmartCards ... 2835
44.4.1.6 Configuring SIM Linear Redundancy Check (LRC) Block ... 2836
44.4.1.7 Configuring SIM Cyclic Redundancy Check (CRC) Block ... 2836
44.4.2 Using the SIM Receiver ... 2837
44.4.2.1 Receive Parity Errors and Parity NACK Generation ... 2837
44.4.2.2 Receive Frame Errors ... 2838
44.4.2.3 Receive Overrun Errors and Overrun NACK Generation ... 2839
44.4.2.4 Using Initial Character Mode and Resulting Receive Data Formats ... 2839
44.4.2.5 Initial Character Mode Programming Notes ... 2840
44.4.2.6 Automatic Receiver Mode ... 2841
44.4.2.7 Using the SIM Receiver with "T=1" SIM Cards ... 2841
44.4.3 Using the SIM Transmitter ... 2842
44.4.3.1 Transmit Data Formats ... 2843
44.4.3.2 Transmit NACK ... 2843
44.4.3.3 Transmit Guard Time ... 2844
44.4.3.4 Using SIM Transmit with "T=1" SIM Cards ... 2844
44.4.4 Suggested "T=1" Compliant Programming Model ... 2845

44.4.4.1 Answer To Reset (ATR) Detection....2845
44.4.4.2 Programming Considerations for Geldkarte Cards....2847
44.4.4.3 Programming Considerations for T=0 SIM Cards....2848
44.4.4.4 Programming Considerations for T=1 SIM Cards....2849
44.5 SIM Memory Map/Register Definition....2851
44.5.1 SIM Port1 Control Register (SIM_PORT1_CNTL)....2853
44.5.2 SIM Setup Register (SIM_SETUP)....2855
44.5.3 SIM Port 1 Detect Register (SIM_PORT1_DETECT)....2855
44.5.4 SIM Transmit Buffer Register (SIM_XMT_BUF)....2857
44.5.5 SIM Receive Buffer Register (SIM_RCV_BUF)....2858
44.5.6 SIM Port0 Control Register (SIM_PORT0_CNTL)....2859
44.5.7 SIM Control Register (SIM_CNTL)....2861
44.5.8 SIM Clock Prescaler Register (SIM_CLK_PRESCALER)....2863
44.5.9 SIM Receive Threshold Register (SIM_RCV_THRESHOLD)....2864
44.5.10 SIM Enable Register (SIM_ENABLE)....2865
44.5.11 SIM Transmit Status Register (SIM_XMT_STATUS)....2867
44.5.12 SIM Receive Status Register (SIM_RCV_STATUS)....2869
44.5.13 SIM Interrupt Mask Register (SIM_INT_MASK)....2871
44.5.14 SIM Port0 Detect Register (SIM_PORT0_DETECT)....2873
44.5.15 SIM Data Format Register (SIM_DATA_FORMAT)....2875
44.5.16 SIM Transmit Threshold Register (SIM_XMT_THRESHOLD)....2875
44.5.17 SIM Transmit Guard Control Register (SIM_GUARD_CNTL)....2876
44.5.18 SIM Open Drain Configuration Control Register (SIM_OD_CONFIG)....2877
44.5.19 SIM Reset Control Register (SIM_RESET_CNTL)....2878
44.5.20 SIM Character Wait Time Register (SIM_CHAR_WAIT)....2879
44.5.21 SIM General Purpose Counter Register (SIM_GPCNT)....2880
44.5.22 SIM Divisor Register (SIM_DIVISOR)....2880
44.5.23 SIM Block Wait Time Register (SIM_BWT)....2881
44.5.24 SIM Block Guard Time Register (SIM_BGT)....2881

44.5.25 SIM Block Wait Time Register HIGH (SIM_BWT_H)..... 2882
44.5.26 SIM Transmit FIFO Status Register (SIM_XMT_FIFO_STAT)..... 2883
44.5.27 SIM Receive FIFO Counter Register (SIM_RCV_FIFO_CNT)..... 2883
44.5.28 SIM Receive FIFO Write Pointer Register (SIM_RCV_FIFO_WPTR)..... 2884
44.5.29 SIM Receive FIFO Read Pointer Register (SIM_RCV_FIFO_RPTR)..... 2884

Chapter 45
System JTAG Controller (SJC)

45.1 Overview..... 2885
45.1.1 Features..... 2886
45.1.2 Modes of Operation..... 2887
45.2 External Signals..... 2889
45.2.1 External Signal Overview..... 2890
45.2.2 TAP Controller..... 2891
45.2.3 Accessing ExtraDebug Registers..... 2893
45.3 TAP Selection Block (TSB)..... 2895
45.3.1 Select Mode Using Software..... 2895
45.4 Boundary Scan Register (BSR)..... 2896
45.5 SoC JTAG Instruction Register (SJIR)..... 2897
45.5.1 ID_CODE Instruction (IDCODE)..... 2897
45.5.2 SAMPLE/PRELOAD Instruction..... 2899
45.5.3 EXTEST Instruction..... 2899
45.5.4 HIGHZ Instruction..... 2900
45.5.5 BYPASS Instruction..... 2900
45.5.6 ENABLE_ExtraDebug Instruction..... 2900
45.5.7 ENTER_DEBUG instruction..... 2901
45.5.8 TAP Select Instruction..... 2901
45.5.9 EXTEST_PULSE instruction..... 2902
45.5.10 EXTEST_TRAIN instruction..... 2902
45.6 Security..... 2902

45.6.1 JTAG Security Modes 2903
45.6.1.1 Mode 1: No Debug - Maximum Security 2903
45.6.1.2 Mode 2: Secure JTAG - High Security 2904
45.6.1.2.1 Challenge/Response Mechanism in System JTAG Mode 2904
45.6.1.3 Mode 3: JTAG Enabled - Low Security 2905
45.6.2 Software Enabled JTAG 2905
45.6.3 Kill Trace 2906
45.6.4 SJC Disable Fuse 2907
45.7 Functional Description 2908
45.7.1 Static Core Debug 2908
45.7.2 Reset Mechanism 2908
45.8 Initialization/Application Information 2909
45.9 SJC Memory Map/Register Definition 2910
45.9.1 General Purpose Unsecured Status Register 1 (SJC_GPUSR1) 2911
45.9.2 General Purpose Unsecured Status Register 2 (SJC_GPUSR2) 2913
45.9.3 General Purpose Unsecured Status Register 3 (SJC_GPUSR3) 2913
45.9.4 General Purpose Secured Status Register (SJC_GPSSR) 2914
45.9.5 Debug Control Register (SJC_DCR) 2915
45.9.6 Security Status Register (SJC_SSR) 2917
45.9.7 General Purpose Clocks Control Register (SJC_GPCCR) 2920

Chapter 46
Secure Non-Volatile Storage (SNVS)

46.1 SNVS overview 2921
46.1.1 SNVS features 2921
46.1.2 Modes of operation 2921
46.2 SNVS structure 2922
46.2.1 SNVS_HP (high power domain) 2923
46.2.2 Non-secure real time counter 2924
46.2.2.1 Calibrating the time counter 2924

46.2.2.2 Time counter alarm ... 2925
46.2.2.3 Periodic interrupt ... 2925
46.3 SNVS_LP (low power domain) ... 2925
46.3.1 Behavior during system power down ... 2926
46.3.2 Monotonic counter (MC) ... 2926
46.4 SNVS reset and system power up ... 2927
46.4.1 PMIC Interface ... 2927
46.5 SNVS interrupts and alarms ... 2928
46.6 Programming Guidelines ... 2929
46.6.1 RTC control bits setting ... 2929
46.6.2 RTC value read ... 2930
46.6.3 General initialization guidelines ... 2930
46.7 SNVS Register Descriptions ... 2931
46.7.1 SNVS Memory Map ... 2932
46.7.2 SNVS_HP Lock (HPLR) ... 2933
46.7.2.1 Address ... 2933
46.7.2.2 Function ... 2933
46.7.2.3 Diagram ... 2933
46.7.2.4 Fields ... 2934
46.7.3 SNVS_HP Command (HPCOMR) ... 2936
46.7.3.1 Address ... 2936
46.7.3.2 Function ... 2937
46.7.3.3 Diagram ... 2937
46.7.3.4 Fields ... 2937
46.7.4 SNVS_HP Control (HPCR) ... 2940
46.7.4.1 Address ... 2940
46.7.4.2 Function ... 2940
46.7.4.3 Diagram ... 2940
46.7.4.4 Fields ... 2940

46.7.5 SNVS_HP Security Interrupt Control (HPSICR) 2942
46.7.5.1 Address 2942
46.7.5.2 Function 2943
46.7.5.3 Diagram 2943
46.7.5.4 Fields 2943
46.7.6 SNVS_HP Security Violation Control (HPSVCR) 2944
46.7.6.1 Address 2944
46.7.6.2 Function 2944
46.7.6.3 Diagram 2945
46.7.6.4 Fields 2945
46.7.7 SNVS_HP Status (HPSR) 2946
46.7.7.1 Address 2946
46.7.7.2 Function 2946
46.7.7.3 Diagram 2946
46.7.7.4 Fields 2947
46.7.8 SNVS_HP Security Violation Status (HPSVSR) 2949
46.7.8.1 Address 2949
46.7.8.2 Function 2949
46.7.8.3 Diagram 2949
46.7.8.4 Fields 2949
46.7.9 SNVS_HP High Assurance Counter IV (HPHACIVR) 2951
46.7.9.1 Address 2951
46.7.9.2 Function 2951
46.7.9.3 Diagram 2951
46.7.9.4 Fields 2951
46.7.10 SNVS_HP High Assurance Counter (HPHACR) 2952
46.7.10.1 Address 2952
46.7.10.2 Function 2952
46.7.10.3 Diagram 2952

46.7.10.4 Fields....2952
46.7.11 SNVS_HP Real Time Counter MSB (HPRTCMR)....2953
46.7.11.1 Address....2953
46.7.11.2 Function....2953
46.7.11.3 Diagram....2953
46.7.11.4 Fields....2953
46.7.12 SNVS_HP Real Time Counter LSB (HPRTCLR)....2954
46.7.12.1 Address....2954
46.7.12.2 Function....2954
46.7.12.3 Diagram....2954
46.7.12.4 Fields....2954
46.7.13 SNVS_HP Time Alarm MSB (HPTAMR)....2955
46.7.13.1 Address....2955
46.7.13.2 Function....2955
46.7.13.3 Diagram....2955
46.7.13.4 Fields....2955
46.7.14 SNVS_HP Time Alarm LSB (HPTALR)....2956
46.7.14.1 Address....2956
46.7.14.2 Function....2956
46.7.14.3 Diagram....2956
46.7.14.4 Fields....2956
46.7.15 SNVS_LP Lock (LPLR)....2957
46.7.15.1 Address....2957
46.7.15.2 Function....2957
46.7.15.3 Diagram....2957
46.7.15.4 Fields....2957
46.7.16 SNVS_LP Control (LPCR)....2960
46.7.16.1 Address....2960
46.7.16.2 Function....2960

46.7.16.3 Diagram....2960
46.7.16.4 Fields....2960
46.7.17 SNVS_LP Master Key Control (LPMKCR)....2963
46.7.17.1 Address....2963
46.7.17.2 Function....2963
46.7.17.3 Diagram....2963
46.7.17.4 Fields....2964
46.7.18 SNVS_LP Security Violation Control (LPSVCR)....2965
46.7.18.1 Address....2965
46.7.18.2 Function....2965
46.7.18.3 Diagram....2965
46.7.18.4 Fields....2965
46.7.19 SNVS_LP Tamper Glitch Filters Configuration (LPTGFCR)....2966
46.7.19.1 Address....2966
46.7.19.2 Function....2967
46.7.19.3 Diagram....2967
46.7.19.4 Fields....2967
46.7.20 SNVS_LP Tamper Detectors Configuration (LPTDCR)....2968
46.7.20.1 Address....2968
46.7.20.2 Function....2968
46.7.20.3 Diagram....2968
46.7.20.4 Fields....2969
46.7.21 SNVS_LP Status (LPSR)....2971
46.7.21.1 Address....2971
46.7.21.2 Function....2971
46.7.21.3 Diagram....2971
46.7.21.4 Fields....2972
46.7.22 SNVS_LP Secure Real Time Counter MSB (LPSRTCMR)....2974
46.7.22.1 Address....2974

46.7.22.2 Function ... 2974
46.7.22.3 Diagram ... 2974
46.7.22.4 Fields ... 2974
46.7.23 SNVS_LP Secure Real Time Counter LSB (LPSRTCLR) ... 2975
46.7.23.1 Address ... 2975
46.7.23.2 Function ... 2975
46.7.23.3 Diagram ... 2975
46.7.23.4 Fields ... 2975
46.7.24 SNVS_LP Time Alarm (LPTAR) ... 2976
46.7.24.1 Address ... 2976
46.7.24.2 Function ... 2976
46.7.24.3 Diagram ... 2976
46.7.24.4 Fields ... 2976
46.7.25 SNVS_LP Secure Monotonic Counter MSB (LPSMCMR) ... 2977
46.7.25.1 Address ... 2977
46.7.25.2 Function ... 2977
46.7.25.3 Diagram ... 2977
46.7.25.4 Fields ... 2977
46.7.26 SNVS_LP Secure Monotonic Counter LSB (LPSMCLR) ... 2978
46.7.26.1 Address ... 2978
46.7.26.2 Function ... 2978
46.7.26.3 Diagram ... 2978
46.7.26.4 Fields ... 2979
46.7.27 SNVS_LP Power Glitch Detector (LPPGDR) ... 2979
46.7.27.1 Address ... 2979
46.7.27.2 Function ... 2979
46.7.27.3 Diagram ... 2979
46.7.27.4 Fields ... 2980
46.7.28 SNVS_LP General Purpose 0 (alias) (LPGPR0_alias) ... 2980

46.7.28.1 Address....2980
46.7.28.2 Function....2980
46.7.28.3 Diagram....2980
46.7.28.4 Fields....2981
46.7.29 SNVS_LP Zeroizable Master Key (LPZMKRa)....2981
46.7.29.1 Address....2981
46.7.29.2 Function....2981
46.7.29.3 Diagram....2981
46.7.29.4 Fields....2982
46.7.30 SNVS_LP General Purposes 0 .. 3 (LPGPR0_3a)....2982
46.7.30.1 Address....2982
46.7.30.2 Function....2982
46.7.30.3 Diagram....2983
46.7.30.4 Fields....2983
46.7.31 SNVS_LP Tamper Detectors Config 2 (LPTDC2R)....2983
46.7.31.1 Address....2983
46.7.31.2 Function....2983
46.7.31.3 Diagram....2983
46.7.31.4 Fields....2984
46.7.32 SNVS_LP Tamper Detectors Status (LPTDSR)....2986
46.7.32.1 Address....2986
46.7.32.2 Function....2986
46.7.32.3 Diagram....2986
46.7.32.4 Fields....2986
46.7.33 SNVS_LP Tamper Glitch Filter 1 Configuration (LPTGF1CR)....2987
46.7.33.1 Address....2987
46.7.33.2 Function....2988
46.7.33.3 Diagram....2988
46.7.33.4 Fields....2988

46.7.34 SNVS_LP Tamper Glitch Filter 2 Configuration (LPTGF2CR)..... 2989
46.7.34.1 Address..... 2989
46.7.34.2 Function..... 2989
46.7.34.3 Diagram..... 2990
46.7.34.4 Fields..... 2990
46.7.35 SNVS_LP Active Tamper 1 Configuration (LPAT1CR)..... 2991
46.7.35.1 Address..... 2991
46.7.35.2 Function..... 2991
46.7.35.3 Diagram..... 2992
46.7.35.4 Fields..... 2992
46.7.36 SNVS_LP Active Tamper 2 Configuration (LPAT2CR)..... 2992
46.7.36.1 Address..... 2992
46.7.36.2 Function..... 2992
46.7.36.3 Diagram..... 2993
46.7.36.4 Fields..... 2993
46.7.37 SNVS_LP Active Tamper 3 Configuration (LPAT3CR)..... 2993
46.7.37.1 Address..... 2993
46.7.37.2 Function..... 2993
46.7.37.3 Diagram..... 2994
46.7.37.4 Fields..... 2994
46.7.38 SNVS_LP Active Tamper 4 Configuration (LPAT4CR)..... 2994
46.7.38.1 Address..... 2994
46.7.38.2 Function..... 2994
46.7.38.3 Diagram..... 2995
46.7.38.4 Fields..... 2995
46.7.39 SNVS_LP Active Tamper 5 Configuration (LPAT5CR)..... 2995
46.7.39.1 Address..... 2995
46.7.39.2 Function..... 2995
46.7.39.3 Diagram..... 2996

46.7.39.4 Fields....2996
46.7.40 SNVS_LP Active Tamper Control (LPATCTLR)....2996
46.7.40.1 Address....2996
46.7.40.2 Function....2996
46.7.40.3 Diagram....2997
46.7.40.4 Fields....2997
46.7.41 SNVS_LP Active Tamper Clock Control (LPATCLKR)....2998
46.7.41.1 Address....2998
46.7.41.2 Function....2999
46.7.41.3 Diagram....2999
46.7.41.4 Fields....2999
46.7.42 SNVS_LP Active Tamper Routing Control 1 (LPATRC1R)....3000
46.7.42.1 Address....3000
46.7.42.2 Function....3000
46.7.42.3 Diagram....3001
46.7.42.4 Fields....3001
46.7.43 SNVS_LP Active Tamper Routing Control 2 (LPATRC2R)....3003
46.7.43.1 Address....3003
46.7.43.2 Function....3003
46.7.43.3 Diagram....3004
46.7.43.4 Fields....3004
46.7.44 SNVS_HP Version ID 1 (HPVIDR1)....3005
46.7.44.1 Address....3005
46.7.44.2 Function....3005
46.7.44.3 Diagram....3005
46.7.44.4 Fields....3005
46.7.45 SNVS_HP Version ID 2 (HPVIDR2)....3006
46.7.45.1 Address....3006
46.7.45.2 Function....3006

46.7.45.3 Diagram....3006
46.7.45.4 Fields....3006

Chapter 47
Shared Peripheral Bus Arbiter (SPBA)

47.1 Overview....3009
47.1.1 Features....3010
47.1.2 Modes of operation....3011
47.2 Clocks....3011
47.3 Functional description....3012
47.3.1 Masters arbitration....3012
47.4 Resource ownership control....3015
47.4.1 Access control....3015
47.4.1.1 Peripheral access....3015
47.4.1.2 Peripheral Right Register access....3016
47.4.2 Owner election....3017
47.4.3 Ending ownership....3017
47.4.3.1 Software Controlled Ownership Ending....3017
47.4.4 The Un-owned State....3018
47.5 SPBA Memory Map/Register Definition....3018
47.5.1 Peripheral Rights Register (SPBA_PRR*n*)....3020

Chapter 48
Sony/Philips Digital Interface (SPDIF)

48.1 Introduction....3023
48.1.1 Overview....3025
48.2 External Signals....3025
48.3 Clocks....3026
48.4 Functional Description....3026
48.4.1 SPDIF Receiver....3026
48.4.1.1 Audio Data Reception....3027

48.4.1.1.1 Application Note....3028
48.4.1.2 Channel Status Reception....3029
48.4.1.2.1 Channel Status Interrupt....3030
48.4.1.3 User Bit Reception....3030
48.4.1.4 Validity Flag Reception....3032
48.4.1.5 SPDIF Receiver Interrupt Exception Definition....3032
48.4.1.6 Standards Compliance....3033
48.4.1.7 SPDIF PLOCK Detection and Rxclk Output....3034
48.4.1.8 Measuring Frequency of SPDIF_RxClk....3034
48.4.2 SPDIF Transmitter....3034
48.4.2.1 Audio Data Transmission....3035
48.4.2.2 Channel Status Transmission....3036
48.4.2.3 Validity Flag Transmission....3036
48.5 SPDIF Memory Map/Register Definition....3036
48.5.1 SPDIF Configuration Register (SPDIF_SCR)....3038
48.5.2 CDText Control Register (SPDIF_SRCD)....3040
48.5.3 PhaseConfig Register (SPDIF_SRPC)....3041
48.5.4 InterruptEn Register (SPDIF_SIE)....3042
48.5.5 InterruptStat Register (SPDIF_SIS)....3044
48.5.6 InterruptClear Register (SPDIF_SIC)....3046
48.5.7 SPDIFRxLeft Register (SPDIF_SRL)....3047
48.5.8 SPDIFRxRight Register (SPDIF_SRR)....3048
48.5.9 SPDIFRxCChannel_h Register (SPDIF_SRCSH)....3048
48.5.10 SPDIFRxCChannel_l Register (SPDIF_SRCSL)....3049
48.5.11 UchannelRx Register (SPDIF_SRU)....3049
48.5.12 QchannelRx Register (SPDIF_SRQ)....3050
48.5.13 SPDIFTxLeft Register (SPDIF_STL)....3050
48.5.14 SPDIFTxRight Register (SPDIF_STR)....3051
48.5.15 SPDIFTxCChannelCons_h Register (SPDIF_STCSCH)....3051

48.5.16 SPDIFTxCChannelCons_l Register (SPDIF_STCSCL)..... 3052
48.5.17 FreqMeas Register (SPDIF_SRFM)..... 3052
48.5.18 SPDIFTxClk Register (SPDIF_STC)..... 3053

Chapter 49
System Reset Controller (SRC)

49.1 SRC Overview..... 3055
49.1.1 Features..... 3055
49.2 External Signals..... 3055
49.3 Clocks..... 3056
49.4 Top-level resets, power-up sequence and external supply integration..... 3057
49.4.1 Reset and Power-up Flow..... 3057
49.4.2 Finite-State Machine (FSM)..... 3060
49.4.3 Power mode transitions..... 3061
49.5 Power-On Reset and power sequencing..... 3062
49.5.1 External POR using SRC_POR_B..... 3062
49.5.2 Internal POR..... 3063
49.6 Functional Description..... 3063
49.6.1 Reset Control..... 3063
49.6.1.1 Reset inputs and outputs..... 3063
49.6.1.2 Reset Handling..... 3065
49.6.1.2.1 Reset Qualification..... 3065
49.6.1.2.2 Reset Sequence and De-Assertion..... 3066
49.6.1.2.3 POR (SRC_POR_B)..... 3066
49.6.1.2.4 COLD RESET..... 3067
49.6.1.2.5 WARM RESET..... 3068
49.6.2 Parallel Reset Requests..... 3069
49.6.3 Boot Mode Control..... 3070
49.6.3.1 BOOT_MODE Pin Latching..... 3070
49.7 SRC Memory Map/Register Definition..... 3071

49.7.1 SRC Control Register (SRC_SCR)....3072
49.7.2 SRC Boot Mode Register 1 (SRC_SBMR1)....3075
49.7.3 SRC Reset Status Register (SRC_SRSR)....3075
49.7.4 SRC Interrupt Status Register (SRC_SISR)....3078
49.7.5 SRC Boot Mode Register 2 (SRC_SBMR2)....3080
49.7.6 SRC General Purpose Register 1 (SRC_GPR1)....3081
49.7.7 SRC General Purpose Register 2 (SRC_GPR2)....3082
49.7.8 SRC General Purpose Register 3 (SRC_GPR3)....3082
49.7.9 SRC General Purpose Register 4 (SRC_GPR4)....3083
49.7.10 SRC General Purpose Register 5 (SRC_GPR5)....3083
49.7.11 SRC General Purpose Register 6 (SRC_GPR6)....3084
49.7.12 SRC General Purpose Register 7 (SRC_GPR7)....3084
49.7.13 SRC General Purpose Register 8 (SRC_GPR8)....3085
49.7.14 SRC General Purpose Register 9 (SRC_GPR9)....3085
49.7.15 SRC General Purpose Register 10 (SRC_GPR10)....3086

Chapter 50
Temperature Monitor (TEMPMON)

50.1 Overview....3087
50.2 Software Usage Guidelines....3088
50.3 TEMPMON Memory Map/Register Definition....3090
50.3.1 Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0*n*)....3091
50.3.2 Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1*n*)....3093
50.3.3 Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2*n*)....3094

Chapter 51
Touch Screen Controller (TSC)

51.1 Overview....3095
51.1.1 Features....3096
51.2 Functional Description....3097
51.2.1 Operating modes....3097

51.2.1.1 Idle ... 3097
51.2.1.2 Pre-charge ... 3097
51.2.1.3 Detection ... 3097
51.2.1.4 Measurement ... 3098
51.2.1.5 Data valid check ... 3098
51.2.1.6 Interrupt ... 3098
51.2.1.7 Reset ... 3098
51.2.1.8 Debug mode ... 3098
51.2.2 Configuration ... 3099
51.2.2.1 TSC configurations ... 3099
51.2.2.2 TSC-ADC-TSC analogue configuration ... 3099
51.2.2.3 TSC, TSC analogue and ADC connection ... 3100
51.2.2.4 TSC and GPIO ... 3100
51.2.2.5 ADC-TSC co-working ... 3101
51.3 TSC Memory Map/Register Definition ... 3102
51.3.1 PS Input Buffer Address (TSC_BASIC_SETTING) ... 3103
51.3.2 PS Input Buffer Address (TSC_PS_INPUT_BUFFER_ADDR) ... 3104
51.3.3 Flow Control (TSC_FLOW_CONTROL) ... 3105
51.3.4 Measure Value (TSC_MEASEURE_VALUE) ... 3106
51.3.5 Interrupt Enable (TSC_INT_EN) ... 3107
51.3.6 Interrupt Signal Enable (TSC_INT_SIG_EN) ... 3108
51.3.7 Intterrupt Status (TSC_INT_STATUS) ... 3109
51.3.8 TSC_DEBUG_MODE ... 3111
51.3.9 TSC_DEBUG_MODE2 ... 3113

Chapter 52
TrustZone Address Space Controller (TZASC)

52.1 Overview ... 3117
52.2 Clocks ... 3118
52.3 Address Mapping in various memory mapping modes ... 3118

Chapter 53
Universal Asynchronous Receiver/Transmitter (UART)

53.1 Overview....3119

53.1.1 Features....3120

53.1.2 Modes of operation....3121

53.2 External Signals....3121

53.2.1 Detailed Signal Descriptions....3126

53.2.1.1 Interrupt Signals....3126

53.2.1.1.1 interrupt_uart - UART Interrupt....3126

53.2.1.2 DMA Request Signals....3126

53.2.1.2.1 dma_req_rx - Receiver DMA Request....3126

53.2.1.2.2 dma_req_tx - Transmitter DMA Request....3126

53.2.1.3 Special Signals....3126

53.2.1.3.1 stop_req - Stop Mode....3126

53.2.1.3.2 doze_req - Doze Mode....3127

53.2.1.3.3 debug_req - Debug Mode....3127

53.3 Clocks....3127

53.4 Functional Description....3127

53.4.1 Interrupts and DMA Requests....3127

53.4.2 Clocks....3128

53.4.2.1 Clock requirements....3129

53.4.2.2 Maximum Baud Rate....3129

53.4.2.3 Clocking in Low-Power Modes....3129

53.4.3 General UART Definitions....3130

53.4.3.1 RTS_B - UART Request To Send....3131

53.4.3.2 RTS Edge Triggered Interrupt....3131

53.4.3.3 CTS_B - Clear To Send....3132

53.4.3.4 Programmable CTS_B Deassertion....3132

53.4.3.5 TX_DATA - UART Transmit....3133

53.4.3.6 RX_DATA - UART Receive....3133
53.4.4 Transmitter....3135
53.4.4.1 Transmitter FIFO Empty Interrupt Suppression....3135
53.4.4.2 Transmitting a Break Condition....3137
53.4.5 Receiver....3137
53.4.5.1 Idle Line Detect....3138
53.4.5.2 Aging Character Detect....3139
53.4.5.3 Receiver Wake....3140
53.4.5.4 Receiving a BREAK Condition....3141
53.4.5.5 Vote Logic....3141
53.4.5.6 Baud Rate Automatic Detection Logic....3143
53.4.5.6.1 Baud Rate Automatic Detection Protocol....3144
53.4.5.6.2 New Baud Rate Determination....3144
53.4.5.6.2.1 New Autobaud Counter Stopped bit and Interrupt....3145
53.4.6 Escape Sequence Detection....3145
53.5 Binary Rate Multiplier (BRM)....3147
53.6 Infrared Interface....3149
53.6.1 Generalities-Infrared....3149
53.6.2 Inverted Transmission and Reception bits (INVT & INVR)....3150
53.6.3 InfraRed Special Case (IRSC) Bit....3150
53.6.4 IrDA interrupt....3151
53.6.5 Conclusion about IrDA....3152
53.6.6 Programming IrDA Interface....3153
53.6.6.1 High Speed....3153
53.6.6.2 Low Speed....3153
53.7 9-bit RS-485 Mode....3154
53.7.1 Generalities....3154
53.7.2 Transmit 9-bit RS-485 frames....3155
53.7.3 Receive 9-bit RS-485 frames....3155

53.7.3.1 RS-485 Slave Address Normal Detect Mode ... 3155
53.7.3.2 RS-485 Slave Address Automatic Detect Mode ... 3156
53.8 Low Power Modes ... 3156
53.8.1 UART Operation in System Doze Mode ... 3157
53.8.2 UART Operation in System Stop Mode ... 3157
53.8.3 Power Saving Method in UART ... 3158
53.9 UART Operation in System Debug State ... 3158
53.10 Reset ... 3159
53.10.1 Hardware reset ... 3159
53.10.2 Software reset ... 3159
53.11 Transfer Error ... 3159
53.12 Functional Timing ... 3160
53.12.1 IrDA Mode ... 3160
53.13 Initialization ... 3160
53.13.1 Programming the UART in RS-232 mode ... 3160
53.13.2 Programming the UART in 9-bit RS-485 mode ... 3162
53.14 References ... 3163
53.15 UART Memory Map/Register Definition ... 3164
53.15.1 UART Receiver Register (UART*x*_URXD) ... 3171
53.15.2 UART Transmitter Register (UART*x*_UTXD) ... 3173
53.15.3 UART Control Register 1 (UART*x*_UCR1) ... 3174
53.15.4 UART Control Register 2 (UART*x*_UCR2) ... 3176
53.15.5 UART Control Register 3 (UART*x*_UCR3) ... 3179
53.15.6 UART Control Register 4 (UART*x*_UCR4) ... 3181
53.15.7 UART FIFO Control Register (UART*x*_UFCR) ... 3183
53.15.8 UART Status Register 1 (UART*x*_USR1) ... 3185
53.15.9 UART Status Register 2 (UART*x*_USR2) ... 3188
53.15.10 UART Escape Character Register (UART*x*_UESC) ... 3190
53.15.11 UART Escape Timer Register (UART*x*_UTIM) ... 3190

53.15.12 UART BRM Incremental Register (UART*x*_UBIR)....3191
53.15.13 UART BRM Modulator Register (UART*x*_UBMR)....3191
53.15.14 UART Baud Rate Count Register (UART*x*_UBRC)....3192
53.15.15 UART One Millisecond Register (UART*x*_ONEMS)....3193
53.15.16 UART Test Register (UART*x*_UTS)....3194
53.15.17 UART RS-485 Mode Control Register (UART*x*_UMCR)....3195

Chapter 54
Universal Serial Bus Controller (USB)

54.1 Overview....3197
54.1.1 Features....3198
54.1.2 Modes of Operation....3199
54.1.2.1 Normal Mode....3200
54.1.2.2 Low-Power Mode....3200
54.2 External Signals....3200
54.3 Functional Description....3201
54.3.1 USB 2.0 Controller Core 0....3201
54.3.1.1 Host Mode....3201
54.3.1.2 Peripheral (Device) Mode....3202
54.3.2 USB 2.0 Controller Core 1....3202
54.3.3 USB Power Control....3202
54.3.3.1 Entering Low Power Suspend Mode....3202
54.3.3.2 Wake-Up Events....3203
54.3.3.2.1 Host Mode Events....3203
54.3.4 Interrupts....3204
54.3.4.1 USB Core Interrupts....3204
54.3.4.2 USB Wake-Up Interrupts....3204
54.4 USB Operation Model....3204
54.4.1 Register Interface....3204
54.4.1.1 Configuration, Control and Status Register Set....3205

54.4.1.2 Identification Registers ... 3207
54.4.1.3 OTG Operations ... 3207
54.4.1.3.1 Register Bits ... 3207
54.4.2 Host Data Structures ... 3209
54.4.2.1 Periodic Frame List ... 3209
54.4.2.2 Asynchronous List Queue Head Pointer ... 3211
54.4.2.3 Isochronous (High-Speed) Transfer Descriptor (iTD) ... 3212
54.4.2.3.1 Next Link Pointer ... 3213
54.4.2.3.2 iTD Transaction Status and Control List ... 3214
54.4.2.3.3 iTD Buffer Page Pointer List (Plus) ... 3215
54.4.2.4 Split Transaction Isochronous Transfer Descriptor (siTD) ... 3217
54.4.2.4.1 Next Link Pointer ... 3217
54.4.2.4.2 siTD Endpoint Capabilities/Characteristics ... 3218
54.4.2.4.3 siTD Transfer State ... 3219
54.4.2.4.4 siTD Buffer Pointer List (plus) ... 3220
54.4.2.4.5 siTD Back Link Pointer ... 3221
54.4.2.5 Queue element transfer descriptor (qTD) ... 3221
54.4.2.5.1 Next qTD Pointer ... 3222
54.4.2.5.2 Alternate Next qTD Pointer ... 3223
54.4.2.5.3 qTD Token ... 3223
54.4.2.5.4 qTD Buffer Page Pointer List ... 3226
54.4.2.6 Queue Head ... 3227
54.4.2.6.1 Queue Head Horizontal Link Pointer ... 3227
54.4.2.6.2 Queue Head Endpoint Capabilities/Characteristics ... 3228
54.4.2.6.3 Transfer Overlay-Queue Head ... 3230
54.4.2.7 Periodic Frame Span Traversal Node (FSTN) ... 3231
54.4.2.7.1 FSTN Normal Path Pointer ... 3232
54.4.2.7.2 FSTN Back Path Link Pointer ... 3232
54.4.3 Host Operational Model ... 3233

54.4.3.1 Host Controller Initialization 3233
54.4.3.2 Port Routing and Control 3235
54.4.3.2.1 Port Routing Control through EHCI Configured (CF) Bit 3237
54.4.3.2.2 Port Routing Control through PortOwner and Disconnect Event 3238
54.4.3.2.3 Example Port Routing State Machine 3240
54.4.3.2.3.1 EHCI HC Owner 3240
54.4.3.2.3.2 Companion HC Owner 3241
54.4.3.2.4 Port Power 3241
54.4.3.2.5 Port Reporting Over-Current 3242
54.4.3.3 Suspend/Resume-Host Operational Model 3243
54.4.3.3.1 Port Suspend/Resume 3243
54.4.3.4 Schedule Traversal Rules 3246
54.4.3.4.1 Example - Preserving Micro-Frame Integrity 3248
54.4.3.4.1.1 Transaction Fit - A Best-Fit Approximation Algorithm 3248
54.4.3.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries 3250
54.4.3.6 Periodic Schedule 3253
54.4.3.7 Managing Isochronous Transfers Using iTDs 3255
54.4.3.7.1 Host Controller Operational Model for iTDs 3255
54.4.3.7.2 Software Operational Model for iTDs 3257
54.4.3.7.2.1 Periodic scheduling threshold 3259
54.4.3.8 Asynchronous Schedule 3260
54.4.3.8.1 Adding Queue Heads to Asynchronous Schedule 3261
54.4.3.8.2 Removing Queue Heads from Asynchronous Schedule 3262
54.4.3.8.3 Empty Asynchronous Schedule Detection 3264
54.4.3.8.4 Restarting Asynchronous Schedule Before EOF 3265
54.4.3.8.4.1 Example Method for Restarting Asynchronous Schedule Traversal 3266
54.4.3.8.4.2 Async Sched Not Active 3267
54.4.3.8.4.3 Async Sched Active 3267

54.4.3.8.4.4 Async Sched Sleeping 3268
54.4.3.8.4.5 Example Derivation for AsyncSchedSleepTime..... 3268
54.4.3.8.5 Asynchronous schedule traversal: Start Event..... 3268
54.4.3.8.6 Reclamation Status Bit (USBSTS Register) 3269
54.4.3.9 Operational Model for Nak Counter..... 3269
54.4.3.9.1 Nak Count Reload Control 3271
54.4.3.9.1.1 Wait for List Head 3272
54.4.3.9.1.2 Do Reload 3272
54.4.3.9.1.3 Wait for Start Event 3272
54.4.3.10 Managing Control/Bulk/Interrupt Transfers through Queue Heads..... 3273
54.4.3.10.1 Fetch Queue Head 3275
54.4.3.10.2 Advance Queue 3275
54.4.3.10.3 Execute Transaction 3276
54.4.3.10.3.1 Interrupt Transfer Pre-condition Criteria 3277
54.4.3.10.3.2 Asynchronous Transfer Pre-operations and Pre-condition Criteria 3277
54.4.3.10.3.3 Transfer Type Independent Pre-operations..... 3277
54.4.3.10.3.4 Halting a Queue Head 3280
54.4.3.10.3.5 Asynchronous Schedule Park Mode 3281
54.4.3.10.4 Write Back qTD 3283
54.4.3.10.5 Follow Queue Head Horizontal Pointer 3283
54.4.3.10.6 Buffer Pointer List Use for Data Streaming with qTDs 3284
54.4.3.10.7 Adding Interrupt Queue Heads to the Periodic Schedule 3286
54.4.3.10.8 Managing Transfer Complete Interrupts from Queue Heads 3286
54.4.3.11 Ping Control..... 3287
54.4.3.12 Split Transactions 3288
54.4.3.12.1 Split Transactions for Asynchronous Transfers 3289
54.4.3.12.1.1 Asynchronous - Do Start Split..... 3290
54.4.3.12.1.2 Asynchronous - Do Complete Split 3290

54.4.3.12.2 Split Transaction Interrupt 3291
54.4.3.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt 3292
54.4.3.12.2.2 Host Controller Operational Model for FSTNs.....3295
54.4.3.12.2.3 Software Operational Model for FSTNs..... 3298
54.4.3.12.2.4 Tracking Split Transaction Progress for Interrupt Transfers .3299
54.4.3.12.2.5 Split Transaction Execution State Machine for Interrupt 3300
54.4.3.12.2.6 Rebalancing the periodic schedule 3306
54.4.3.12.3 Split Transaction Isochronous 3307
54.4.3.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous . 3307
54.4.3.12.3.2 Tracking Split Transaction Progress for Isochronous Transfers.....3312
54.4.3.12.3.3 Split Transaction Execution State Machine for Isochronous 3314
54.4.3.12.3.4 Periodic Isochronous - Do Start Split 3315
54.4.3.12.3.5 Periodic Isochronous - Do Complete Split 3317
54.4.3.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b 3320
54.4.3.12.3.7 Split Transaction for Isochronous - Processing Examples 3322
54.4.3.13 Host Controller Pause..... 3324
54.4.3.14 Port Test Modes -Host Operational Model..... 3325
54.4.3.15 Interrupts-Host Operational Model..... 3325
54.4.3.15.1 Transfer/Transaction Based Interrupts 3327
54.4.3.15.1.1 Transaction Error 3327
54.4.3.15.1.2 Serial Bus Babble..... 3327
54.4.3.15.1.3 Data Buffer Error 3328
54.4.3.15.1.4 USB Interrupt (Interrupt on Completion (IOC)) 3329
54.4.3.15.1.5 Short Packet.....3329
54.4.3.15.2 Host Controller Event Interrupts 3329
54.4.3.15.2.1 Port Change Events 3330
54.4.3.15.2.2 Frame List Rollover 3330
54.4.3.15.2.3 Interrupt on Async Advance 3330

54.4.3.15.2.4 Host System Error 3330
54.4.4 EHCI Deviation.....3331
54.4.4.1 Embedded Transaction Translator Function.....3332
54.4.4.1.1 Capability Registers.....3332
54.4.4.1.2 Operational Registers.....3333
54.4.4.1.3 Discovery-EHCI Deviation.....3333
54.4.4.1.4 Data Structures.....3333
54.4.4.1.5 Operational Model.....3334
54.4.4.1.5.1 Micro- frame Pipeline.....3334
54.4.4.1.5.2 Split State Machines.....3335
54.4.4.1.5.3 Asynchronous Transaction Scheduling and Buffer Management.....3336
54.4.4.1.5.4 Periodic Transaction Scheduling and Buffer Management....3336
54.4.4.1.5.5 Multiple Transaction Translators.....3336
54.4.4.2 Device Operation.....3337
54.4.4.2.1 USB_USBMODE Register.....3337
54.4.4.2.2 Non-Zero Fields the Register File.....3337
54.4.4.2.3 SOF Interrupt.....3337
54.4.4.3 Embedded Design Interface.....3337
54.4.4.3.1 Frame Adjust Register.....3338
54.4.4.4 Miscellaneous variations from EHCI.....3338
54.4.4.4.1 Programmable Physical Interface Behaviour.....3338
54.4.4.4.2 Discovery.....3338
54.4.4.4.2.1 Port Reset.....3338
54.4.4.4.2.2 Port Speed Detection.....3339
54.4.4.4.3 Port Test Mode.....3339
54.4.5 Device Data Structures.....3339
54.4.5.1 Endpoint Queue Head (dQH).....3340
54.4.5.1.1 Endpoint Capabilities/Characteristics.....3341

54.4.5.1.2 Transfer Overlay-Endpoint Queue Head..... 3342
54.4.5.1.3 Current dTD Pointer..... 3342
54.4.5.1.4 Set-up Buffer.....3343
54.4.5.2 Endpoint Transfer Descriptor (dTD).....3343
54.4.6 Device Operational Model..... 3345
54.4.6.1 Device Controller Initialization..... 3346
54.4.6.2 Port State and Control..... 3347
54.4.6.2.1 Bus Reset..... 3349
54.4.6.2.2 Suspend/Resume..... 3350
54.4.6.2.2.1 Suspend.....3350
54.4.6.2.2.2 Resume..... 3351
54.4.6.3 Managing Endpoints..... 3351
54.4.6.3.1 Endpoint Initialization..... 3352
54.4.6.3.2 Stalling..... 3353
54.4.6.3.3 Data Toggle 3354
54.4.6.3.3.1 Data Toggle Reset..... 3354
54.4.6.3.3.2 Data Toggle Inhibit.....3354
54.4.6.3.3.3 Priming Transmit Endpoints.....3354
54.4.6.3.3.4 Priming Receive Endpoints..... 3355
54.4.6.4 Operational Model For Packet Transfers..... 3355
54.4.6.4.1 Interrupt/Bulk Endpoint Operational Model.....3356
54.4.6.4.1.1 Interrupt/Bulk Endpoint Bus Response Matrix..... 3358
54.4.6.4.2 Control Endpoint Operation Model..... 3358
54.4.6.4.2.1 Setup Phase.....3358
54.4.6.4.2.2 Data Phase..... 3359
54.4.6.4.2.3 Status Phase..... 3360
54.4.6.4.2.4 Control Endpoint Bus Response Matrix.....3360
54.4.6.4.3 Isochronous Endpoint Operational Model..... 3361
54.4.6.4.3.1 Isochronous Pipe Synchronization..... 3363

54.4.6.4.3.2 Isochronous Endpoint Bus Response Matrix..... 3363
54.4.6.5 Managing Queue Heads..... 3363
54.4.6.5.1 Queue Head Initialization..... 3364
54.4.6.5.2 Operational Model For Setup Transfers..... 3365
54.4.6.6 Managing Transfers with Transfer Descriptors..... 3366
54.4.6.6.1 Software Link Pointers..... 3366
54.4.6.6.2 Building a Transfer Descriptor..... 3366
54.4.6.6.3 Executing A Transfer Descriptor..... 3367
54.4.6.6.4 Transfer Completion..... 3368
54.4.6.6.5 Flushing/De-priming an Endpoint..... 3368
54.4.6.6.6 Device Error Matrix..... 3369
54.4.6.7 Servicing Interrupts..... 3370
54.4.6.7.1 High-Frequency Interrupts..... 3370
54.4.6.7.2 Low-Frequency Interrupts..... 3370
54.4.6.7.3 Error Interrupts..... 3370
54.5 USB Non-Core Memory Map/Register Definition..... 3371
54.5.1 USB OTG1 Control Register (USBNC_USB_OTG1_CTRL)..... 3373
54.5.2 USB OTG2 Control Register (USBNC_USB_OTG2_CTRL)..... 3375
54.5.3 OTG1 UTMI PHY Control 0 Register (USBNC_USB_OTG1_PHY_CTRL_0)..... 3377
54.5.4 OTG2 UTMI PHY Control 0 Register (USBNC_USB_OTG2_PHY_CTRL_0)..... 3378
54.6 USB Core Memory Map/Register Definition..... 3378
54.6.1 Identification register (USB_*n*ID)..... 3383
54.6.2 Hardware General (USB_*n*HWGENERAL)..... 3383
54.6.3 Host Hardware Parameters (USB_*n*HWHOST)..... 3385
54.6.4 Device Hardware Parameters (USB_*n*HWDEVICE)..... 3386
54.6.5 TX Buffer Hardware Parameters (USB_*n*HWTXBUF)..... 3386
54.6.6 RX Buffer Hardware Parameters (USB_*n*HWRXBUF)..... 3387
54.6.7 General Purpose Timer #0 Load (USB_*n*GPTIMER0LD)..... 3388
54.6.8 General Purpose Timer #0 Controller (USB_*n*GPTIMER0CTRL)..... 3388

54.6.9 General Purpose Timer #1 Load (USB_*n*GPTIMER1LD)....3390
54.6.10 General Purpose Timer #1 Controller (USB_*n*GPTIMER1CTRL)....3390
54.6.11 System Bus Config (USB_*n*SBUSCFG)....3391
54.6.12 Capability Registers Length (USB_*n*CAPLENGTH)....3392
54.6.13 Host Controller Interface Version (USB_*n*HCIVERSION)....3393
54.6.14 Host Controller Structural Parameters (USB_*n*HCSPARAMS)....3393
54.6.15 Host Controller Capability Parameters (USB_*n*HCCPARAMS)....3395
54.6.16 Device Controller Interface Version (USB_*n*DCIVERSION)....3397
54.6.17 Device Controller Capability Parameters (USB_*n*DCCPARAMS)....3398
54.6.18 USB Command Register (USB_*n*USBCMD)....3399
54.6.19 USB Status Register (USB_*n*USBSTS)....3403
54.6.20 Interrupt Enable Register (USB_*n*USBINTR)....3407
54.6.21 USB Frame Index (USB_*n*FRINDEX)....3409
54.6.22 Frame List Base Address (USB_*n*PERIODICLISTBASE)....3410
54.6.23 Device Address (USB_*n*DEVICEADDR)....3410
54.6.24 Next Asynch. Address (USB_*n*ASYNCLISTADDR)....3411
54.6.25 Endpoint List Address (USB_*n*ENDPTLISTADDR)....3412
54.6.26 Programmable Burst Size (USB_*n*BURSTSIZE)....3412
54.6.27 TX FIFO Fill Tuning (USB_*n*TXFILLTUNING)....3413
54.6.28 Endpoint NAK (USB_*n*ENDPTNAK)....3415
54.6.29 Endpoint NAK Enable (USB_*n*ENDPTNAKEN)....3415
54.6.30 Configure Flag Register (USB_*n*CONFIGFLAG)....3416
54.6.31 Port Status & Control (USB_*n*PORTSC1)....3416
54.6.32 On-The-Go Status & control (USB_*n*OTGSC)....3423
54.6.33 USB Device Mode (USB_*n*USBMODE)....3427
54.6.34 Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT)....3428
54.6.35 Endpoint Prime (USB_*n*ENDPTPRIME)....3429
54.6.36 Endpoint Flush (USB_*n*ENDPTFLUSH)....3430
54.6.37 Endpoint Status (USB_*n*ENDPTSTAT)....3430

54.6.38 Endpoint Complete (USB_*n*ENDPTCOMPLETE)..... 3431
54.6.39 Endpoint Control0 (USB_*n*ENDPTCTRL0)..... 3432
54.6.40 Endpoint Control 1 (USB_*n*ENDPTCTRL1)..... 3434
54.6.41 Endpoint Control 2 (USB_*n*ENDPTCTRL2)..... 3437
54.6.42 Endpoint Control 3 (USB_*n*ENDPTCTRL3)..... 3439
54.6.43 Endpoint Control 4 (USB_*n*ENDPTCTRL4)..... 3442
54.6.44 Endpoint Control 5 (USB_*n*ENDPTCTRL5)..... 3445
54.6.45 Endpoint Control 6 (USB_*n*ENDPTCTRL6)..... 3448
54.6.46 Endpoint Control 7 (USB_*n*ENDPTCTRL7)..... 3451

Chapter 55
Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

55.1 USB PHY Overview..... 3455
55.2 Operation..... 3455
55.2.1 UTMI..... 3455
55.2.2 Digital Transmitter..... 3456
55.2.3 Digital Receiver..... 3456
55.2.4 Analog Receiver..... 3456
55.2.4.1 HS Differential Receiver..... 3457
55.2.4.2 Squelch Detector..... 3458
55.2.4.3 LS/FS Differential Receiver..... 3458
55.2.4.4 HS Disconnect Detector..... 3458
55.2.4.5 USB Plugged-In Detector..... 3458
55.2.4.6 Single-Ended USB_DP Receiver..... 3459
55.2.4.7 Single-Ended USB_DN Receiver..... 3459
55.2.4.8 9X Oversample Module..... 3459
55.2.5 Analog Transmitter..... 3459
55.2.5.1 Switchable High-Speed 45Ω Termination Resistors..... 3459
55.2.5.2 Low-Speed/Full-Speed Differential Driver..... 3460
55.2.5.3 High-Speed Differential Driver..... 3460

55.2.5.4 Switchable 1.5KΩ USB_DP Pullup Resistor....3460
55.2.5.5 Switchable 15KΩ USB_DP Pulldown Resistor....3460
55.2.6 Recommended Register Configuration for USB Certification....3462
55.2.7 Charger detection....3462
55.2.7.1 Charger detect control table....3462
55.2.7.2 Data pin contact detector....3462
55.2.7.3 Charger detector....3463
55.2.7.4 Charger detection software flow....3463
55.2.7.5 Dead Battery Protect....3465
55.3 USB PHY Memory Map/Register Definition....3465
55.3.1 USB PHY Power-Down Register (USBPHY*x*_PWD*n*)....3468
55.3.2 USB PHY Transmitter Control Register (USBPHY*x*_TX*n*)....3470
55.3.3 USB PHY Receiver Control Register (USBPHY*x*_RX*n*)....3471
55.3.4 USB PHY General Control Register (USBPHY*x*_CTRL*n*)....3473
55.3.5 USB PHY Status Register (USBPHY*x*_STATUS)....3476
55.3.6 USB PHY Debug Register (USBPHY*x*_DEBUG*n*)....3478
55.3.7 UTMI Debug Status Register 0 (USBPHY*x*_DEBUG0_STATUS)....3480
55.3.8 UTMI Debug Status Register 1 (USBPHY*x*_DEBUG1*n*)....3481
55.3.9 UTMI RTL Version (USBPHY*x*_VERSION)....3482
55.4 USB Analog Memory Map/Register Definition....3482
55.4.1 USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT*n*)....3484
55.4.2 USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT*n*)....3485
55.4.3 USB VBUS Detect Status Register (USB_ANALOG_USB1_VBUS_DETECT_STAT)....3487
55.4.4 USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)....3489
55.4.5 USB Misc Register (USB_ANALOG_USB1_MISC*n*)....3490
55.4.6 USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT*n*)....3491
55.4.7 USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT*n*)....3493
55.4.8 USB VBUS Detect Status Register (USB_ANALOG_USB2_VBUS_DETECT_STAT)....3495
55.4.9 USB Charger Detect Status Register (USB_ANALOG_USB2_CHRG_DETECT_STAT)....3497

55.4.10 USB Misc Register (USB_ANALOG_USB2_MISC*n*)....3498
55.4.11 Chip Silicon Version (USB_ANALOG_DIGPROG)....3499

Chapter 56
Ultra Secured Digital Host Controller (uSDHC)

56.1 Overview....3501
56.1.1 Features....3504
56.1.2 Modes and Operations....3505
56.1.2.1 Data transfer Modes....3505
56.2 External Signals....3505
56.2.1 Signals Overview....3507
56.3 Clocks....3508
56.4 Functional Description....3508
56.4.1 Data Buffer....3508
56.4.1.1 Write Operation Sequence....3511
56.4.1.2 Read Operation Sequence....3511
56.4.1.3 Data Buffer and Block Size....3512
56.4.1.4 Dividing Large Data Transfer....3513
56.4.1.5 External DMA Request....3514
56.4.2 DMA AHB Interface....3515
56.4.2.1 Internal DMA Request....3516
56.4.2.2 DMA Burst Length....3517
56.4.2.3 AHB Master Interface....3517
56.4.2.4 ADMA Engine....3517
56.4.2.4.1 ADMA Concept and Descriptor Format....3518
56.4.2.4.2 ADMA Interrupt....3522
56.4.2.4.3 ADMA Error....3523
56.4.3 Register Bank with IP Bus Interface....3523
56.4.3.1 SD Protocol Unit....3524
56.4.3.2 SD control misc....3525

56.4.3.3 SD Clock control....3525
56.4.3.4 Command control....3525
56.4.3.5 Data control....3526
56.4.4 Clock & Reset Manager....3526
56.4.5 Clock Generator....3527
56.4.6 SDIO Card Interrupt....3527
56.4.6.1 Interrupts in 1-bit Mode....3527
56.4.6.2 Interrupt in 4-bit Mode....3528
56.4.6.3 Card Interrupt Handling....3528
56.4.7 Card Insertion and Removal Detection....3529
56.4.8 Power Management and Wake Up Events....3530
56.4.8.1 Setting Wake Up Events....3531
56.4.9 MMC fast boot....3531
56.4.9.1 Boot operation....3531
56.4.9.2 Alternative boot operation....3532
56.5 Initialization/Application of uSDHC....3533
56.5.1 Command Send & Response Receive Basic Operation....3533
56.5.2 Card Identification Mode....3534
56.5.2.1 Card Detect....3534
56.5.2.2 Reset....3535
56.5.2.3 Voltage Validation....3536
56.5.2.4 Card Registry....3538
56.5.3 Card Access....3539
56.5.3.1 Block Write....3539
56.5.3.1.1 Normal Write....3539
56.5.3.1.2 DDR Write....3541
56.5.3.1.3 Write with Pause....3541
56.5.3.2 Block Read....3543
56.5.3.2.1 Normal Read....3543

56.5.3.2.2 DDR Read....3544
56.5.3.2.3 Read with Pause....3544
56.5.3.2.4 DLL (Delay Line) in Read Path....3545
56.5.3.3 Suspend Resume....3547
56.5.3.3.1 Suspend....3547
56.5.3.3.2 Resume....3548
56.5.3.4 ADMA Usage....3548
56.5.3.5 Transfer Error....3549
56.5.3.5.1 CRC Error....3549
56.5.3.5.2 Internal DMA Error....3549
56.5.3.5.3 Transfer ADMA Error....3550
56.5.3.5.4 Auto CMD12 Error....3550
56.5.3.6 Card Interrupt....3551
56.5.4 Switch Function....3551
56.5.4.1 Query, Enable and Disable SDIO High Speed Mode....3552
56.5.4.2 Query, Enable and Disable SD High Speed Mode/SDR50/SDR104/DDR50....3552
56.5.4.3 Query, Enable and Disable MMC High Speed Mode....3553
56.5.4.4 Set MMC Bus Width....3553
56.5.5 ADMA Operation....3553
56.5.5.1 ADMA1 Operation....3554
56.5.5.2 ADMA2 Operation....3554
56.5.6 Fast Boot Operation....3555
56.5.6.1 Normal fast boot flow....3555
56.5.6.2 Alternative fast boot flow....3556
56.5.6.3 Fast boot application case (in DMA mode)....3556
56.6 Commands for MMC/SD/SDIO....3558
56.7 Software Restrictions....3563
56.7.1 Initialization Active....3563
56.7.2 Software Polling Procedure....3564

56.7.3 Suspend Operation ... 3564
56.7.4 Data Length Setting ... 3564
56.7.5 (A)DMA Address Setting ... 3564
56.7.6 Data Port Access ... 3565
56.7.7 Change Clock Frequency ... 3565
56.7.8 Multi-block Read ... 3565
56.8 uSDHC Memory Map/Register Definition ... 3565
56.8.1 DMA System Address (uSDHC*x*_DS_ADDR) ... 3568
56.8.2 Block Attributes (uSDHC*x*_BLK_ATT) ... 3569
56.8.3 Command Argument (uSDHC*x*_CMD_ARG) ... 3570
56.8.4 Command Transfer Type (uSDHC*x*_CMD_XFR_TYP) ... 3571
56.8.5 Command Response0 (uSDHC*x*_CMD_RSP0) ... 3574
56.8.6 Command Response1 (uSDHC*x*_CMD_RSP1) ... 3575
56.8.7 Command Response2 (uSDHC*x*_CMD_RSP2) ... 3575
56.8.8 Command Response3 (uSDHC*x*_CMD_RSP3) ... 3576
56.8.9 Data Buffer Access Port (uSDHC*x*_DATA_BUFF_ACC_PORT) ... 3577
56.8.10 Present State (uSDHC*x*_PRES_STATE) ... 3578
56.8.11 Protocol Control (uSDHC*x*_PROT_CTRL) ... 3583
56.8.12 System Control (uSDHC*x*_SYS_CTRL) ... 3588
56.8.13 Interrupt Status (uSDHC*x*_INT_STATUS) ... 3591
56.8.14 Interrupt Status Enable (uSDHC*x*_INT_STATUS_EN) ... 3597
56.8.15 Interrupt Signal Enable (uSDHC*x*_INT_SIGNAL_EN) ... 3600
56.8.16 Auto CMD12 Error Status (uSDHC*x*_AUTOCMD12_ERR_STATUS) ... 3603
56.8.17 Host Controller Capabilities (uSDHC*x*_HOST_CTRL_CAP) ... 3606
56.8.18 Watermark Level (uSDHC*x*_WTMK_LVL) ... 3609
56.8.19 Mixer Control (uSDHC*x*_MIX_CTRL) ... 3610
56.8.20 Force Event (uSDHC*x*_FORCE_EVENT) ... 3612
56.8.21 ADMA Error Status Register (uSDHC*x*_ADMA_ERR_STATUS) ... 3615
56.8.22 ADMA System Address (uSDHC*x*_ADMA_SYS_ADDR) ... 3617

56.8.23 DLL (Delay Line) Control (uSDHCx_DLL_CTRL)..... 3618
56.8.24 DLL Status (uSDHCx_DLL_STATUS)..... 3620
56.8.25 CLK Tuning Control and Status (uSDHCx_CLK_TUNE_CTRL_STATUS)..... 3621
56.8.26 Vendor Specific Register (uSDHCx_VEND_SPEC)..... 3623
56.8.27 MMC Boot Register (uSDHCx_MMC_BOOT)..... 3626
56.8.28 Vendor Specific 2 Register (uSDHCx_VEND_SPEC2)..... 3627
56.8.29 Tuning Control Register (uSDHCx_TUNING_CTRL)..... 3629

Chapter 57
Watchdog Timer (WDOG)

57.1 Overview..... 3631
57.1.1 Features..... 3632
57.2 External signals..... 3633
57.3 Clocks..... 3633
57.4 Watchdog mechanism and system integration..... 3634
57.5 Functional description..... 3634
57.5.1 Timeout event..... 3634
57.5.1.1 Servicing WDOG to reload the counter..... 3635
57.5.2 Interrupt event..... 3635
57.5.3 Power-down counter event..... 3636
57.5.4 Low power modes..... 3636
57.5.4.1 STOP and DOZE mode..... 3636
57.5.4.2 WAIT mode..... 3636
57.5.5 Debug mode..... 3637
57.5.6 Operations..... 3637
57.5.6.1 Watchdog reset generation..... 3637
57.5.6.2 WDOG_B generation..... 3638
57.5.7 Reset..... 3640
57.5.8 Interrupt..... 3640
57.5.9 Flow Diagrams..... 3640

57.6 Initialization....3642
57.7 WDOG Memory Map/Register Definition....3643
57.7.1 Watchdog Control Register (WDOG*x*_WCR)....3644
57.7.2 Watchdog Service Register (WDOG*x*_WSR)....3645
57.7.3 Watchdog Reset Status Register (WDOG*x*_WRSR)....3646
57.7.4 Watchdog Interrupt Control Register (WDOG*x*_WICR)....3647
57.7.5 Watchdog Miscellaneous Control Register (WDOG*x*_WMCR)....3648

Chapter 58
Crystal Oscillator (XTALOSC)

58.1 Overview....3649
58.2 External Signals....3649
58.3 Crystal Oscillator 24 MHz....3650
58.3.1 Oscillator Configuration (24 MHz)....3650
58.3.2 RC Oscillator (24 MHz)....3651
58.3.3 Crystal Frequency Detection(24 MHz)....3652
58.4 Crystal Oscillator 32 kHz....3652
58.4.1 Oscillator Configuration (32 kHz)....3652
58.4.2 Bypass Configuration (32 kHz)....3653
58.5 XTALOSC 24MHz Memory Map/Register Definition....3654
58.5.1 Miscellaneous Register 0 (XTALOSC24M_MISC0)....3656
58.5.2 XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL*n*)....3659
58.5.3 XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0*n*)....3662
58.5.4 XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1*n*)....3663
58.5.5 XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2*n*)....3664

Chapter 1
Introduction

1.1 About This Document

The i.MX 6UltraLite application processors are Freescale Semiconductor's latest additions to a growing family of multimedia-focused products offering high performance processing optimized for lowest power consumption.

The i.MX 6UltraLite processors feature Freescale's advanced implementation of the ARM®Cortex®-A7 core, which can be interfaced with DDR3, DDR3L and LPDDR2 (single channel) DRAM memory devices.

These products are suitable for applications such as:

- Point-of-Sale Device
- General embedded devices
- Industrial devices

The i.MX is a system on a chip (SoC) which integrates on one integrated circuit a complete microcomputer platform based on the ARM Cortex-A9 including an integrated NEON MPE coprocessor, separate 32 KByte instruction and data L1 caches and a unified 1 MByte L2 cache. Supporting the ARM Platform are a large array of system-level components such as system RAM, processor code ROM, Smart DMA controller (SDMA) and system control devices such as a power-on reset controller, voltage regulators, power management, and clock generation circuitry. Coprocessors including two Graphics Processing Units (GPUs), an Image Processing Unit (IPU), Video Processing Unit (VPU) and Asynchronous Sample Rate Converter (ASRC) accelerate computationally intensive tasks offloading the ARM Platform.

1.1.1 Audience

This manual is intended to be used by board-level product designers and product software developers. This manual assumes that the reader has a background in computer engineering and/or software engineering and understands concepts of digital system design, microprocessor architecture, Input / Output (I/O) devices, industry standard communication and device interface protocols.

1.1.2 Organization

This document covers the chip at a system level and provides an architectural overview. Also covered are system memory map, system-level interrupt events, external pins and pin multiplexing, external memory, system debug, system boot, multimedia subsystem, power management, and system security.

1.1.3 Suggested Reading

This section lists additional reading materials that provide background for the information in this manual, as well as general information about the architecture.

1.1.3.1 General Information

The following documentation provides useful background information about the ARM Cortex-A7 processor.

For information about the ARM Cortex-A7 processor see:

- **http://infocenter.arm.com**

1.1.3.2 Related Documentation

Freescale documentation is available from the sources listed on the back cover of this manual; the document order numbers are included in parentheses for ease in ordering.

For a current list of documentation, refer to **http://www.freescale.com**.

1.1.4 Conventions

This document uses the following notational conventions:

cleared / set

When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.

mnemonics

Instruction mnemonics are shown in lowercase bold

italics

Italics indicate variable command parameters, for example, **bcctr***x*

Book titles in text are set in italics

15

An integer in decimal

0x

Prefix to denote hexadecimal number

0b

Prefix to denote binary number. Binary 0 and 1 are written without the prefix.

n'H4000CA00

n-bit Hexadecimal number

BLK_REG_NAME

Register names are all uppercase. The block mnemonic is prepended with an underscore delimiter (_).

BLK_REG[FIELD]

Fields within registers appear in brackets. For example, ESR[RLS] refers to the Receive Last Slot field of the ESAI Status Register.

BLK_REG[*n*]

Bit number *n* within register BLK.REG.

BLK_REG[*l:r*]

Register bit ranges. Ranges are indicated by the left-most bit number *l* and the right-most bit number *r* separated by a colon (:). For example, ESR[15:0] refers to the lower half word in the ESAI Status Register.

x, U

In some contexts, such as signal encodings, an unitalicized x indicates a don't care or uninitialized. The binary value could be 1 or 0.

x

An italicized *x* indicates an alphanumeric variable

n, m

Italicized *n or m* represent integer variables

!

Binary logic operator NOT

&&

Binary logic operator AND

||

Binary logic operator OR

^ or <O+>

Binary logic operator XOR. For example, A <O+> B

|

Bit-wise OR. For example, 0b0001 | 0b1000 yields the value 0b1001.

&

Bit-wise AND. For example, 0b0001 & 0b1000 yields the value 0b0000.

{A,B}

Concatenation, where the *n*-bit value A is prepended to the *m*-bit value B to form an (*n* +*m*)-bit value. For example, {0, REG*m* [14:0]} yeilds a 16-bit value with 0 in the most significant bit.

- or grey fill

Indicates a reserved bit field in an register. Although these bits can be written to as ones or zeros, they are always read as zeros.

>>

Shift right logical one position

<<

Shift left logical one posiiton

<=

Assignment

==

Compare equal

!=

Compare not equal

>

Greater than

<

Less than

1.1.5 Register Access

1.1.5.1 Register Diagram Field Access Type Legend

The following figure provides the interpretation of the notation used in register diagrams for a number of common field access types.

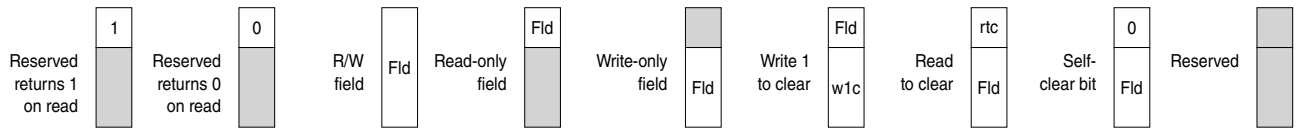


Figure 1-1. Register Field Conventions

NOTE

For reserved register fields, software should mask off the data in the field after read (software can not rely on the contents of data read from a reserved field) and always write all zeros.

1.1.5.2 Register Macro Usage

A common operation is to update one field without disturbing the contents of the remaining fields in the register. Normally, this requires a read-modify-write (RMW) operation, where the CPU reads the register, modifies the target field, then writes the results back to the register. This is an expensive operation in terms of CPU cycles, because of the initial register read.

To address this issue, some hardware registers are implemented as a group, including registers that can be used to either set, clear, or toggle (SCT) individual bits of the primary register. When writing to an SCT register, all bits set to 1 perform the associated operation on the primary register, while all bits set to 0 are not affected. The SCT registers always read back 0, and should be considered write-only. The SCT registers are not implemented if the primary register is read-only.

With this architecture, it is possible to update one or more fields using only register writes. First, all bits of the target fields are cleared by a write to the associated clear register, then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (that is, one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read/modify/write operations. When atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

A set of SCT registers is offered for registers in many modules on this device, as described in this manual. In a module memory map table, the suffix _SET, _CLR, or _TOG is added to the base name of the register. For example, the CCM_ANALOG_PLL_ARM register has three other registers called CCM_ANALOG_PLL_ARM_SET, CCM_ANALOG_PLL_ARM_CLR and CCM_ANALOG_PLL_ARM_TOG.

In the sub-section that describes one of these sets of registers, a short-hand convention is used to denote that a register has an SCT register set. Here there is an italicized *n* appended to the end of the short register name. Using the above example, the name used for this register is CCM_ANALOG_PLL_ARM*n*. When seeing this designation, there is a SCT register set associated with the register, and can verify this by checking it in the memory map table. The address offset for each of these registers is given in the form of the following example:

Address: 20C_8000h base + 0h offset + (4d × i), where i=0d to 3d

In this example, the address for each of the base register and its three SCT registers can be calculated as:

Register	Address
CCM_ANALOG_PLL_ARM	20C_8000h
CCM_ANALOG_PLL_ARM_SET	20C_8004h
CCM_ANALOG_PLL_ARM_CLR	20c_8008h
CCM_ANALOG_PLL_ARM_TOG	20C_800Ch

1.1.6 Signal Conventions

_b, _B

When appended to a signal name, indicates that a signal is active-low

NEG_ACTIVE

Overbar also denotes a negative active signal

UPPERCASE

Package pin names, Block I/O signals

lowercase

Lowercase is used to indicate internal signals

1.1.7 Acronyms and Abbreviations

The table below contains acronyms and abbreviations used in this document.

Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AIPS	ARM IP Bus
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASRC	Asynchronous Sample Rate Converter
AXI	Advanced eXtensible Interface
BEE	Bus Encrypt Engine
BIST	Built-in self test
CA7	ARM Cortex A7
CAAM	Cryptographic Acceleration and Assurance Module
CAN	Controller Area Network
CCM	Clock Controller Module
CPU	Central Processing Unit
CSI	CMOS Sensor Interface
CSU	Central Security Unit
CTI	Cross Trigger Interface
DAP	Debug Access Port
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
ECC	Error correcting codes
ECSPI	Enhanced Configurable SPI
EIM	External Interface Module
ENET	Ethernet
EPIT	Enhanced Periodic Interrupt Timer
EPROM	Erasable programmable read-only memory
ETF	Embedded Trace FIFO
ETM	Embedded Trace Macrocell
FIFO	First-in-first-out
GIC	General Interrupt Controller
GPC	General Power Controller

Table continues on the next page...

Term	Meaning
GPIO	General-purpose I/O
GPR	General-purpose register
GPS	Global Positioning System
GPT	General Purpose Timer
GPU	Graphics Processing Unit
GPV	Global Programmers View
HAB	High Assurance Boot
I2C or I^2C	Inter-integrated circuit
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IOMUX	Input-output multiplexer
IP	Intellectual Property
IrDA	Infrared Data Association
JTAG	Joint Test Action Group (a serial bus protocol usually used for test purposes)
LCD	Liquid Crystal Display
LDO	Low-Dropout
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
LUT	Lookup Table
LVDS	Low Voltage Differential Signaling
MAC	Medium Access Control
MMC	MultiMedia Card
MMDC	Multi Mode DDR Controller
MSB	Most-significant byte
MT/s	Mega transfers per second
OCRAM	On-Chip Random Access Memory
OCOTP	On-Chip One-Time Programmable Controller
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PGC	Power Gating Controller
PIC	Programmable interrupt controller
PMU	Power Management Unit
POR	Power-on reset
PSRAM	Pseudo-Static Random Access Memory
PWM	Pulse Width Modulation
PXP	Pixel Pipeline
QoS	Quality of Service
R2D	Radians to Degrees
RISC	Reduced instruction set computing
ROM	Read-Only Memory

Table continues on the next page...

Term	Meaning
ROMCP	ROM Controller with Patch
RTOS	Real-time operating system
Rx	Receive
SAI	Synchronous Audio Interface
SCU	Snoop Control Unit
SD	Secure Digital
SDIO	Secure Digital Input/Output
SDLC	Synchronous data link control
SDMA	Smart DMA
SIM	Subscriber Identification Module
SNVS	Secure Non-Volatile Storage
SoC	System-On-Chip
SPBA	Shared Peripheral Bus Arbiter
SPDIF	Sony Phillips Digital Interface
SPI	Serial peripheral interface
SRAM	Static random access memory
SRC	System Reset Controller
TFT	Thin Film Transistor
TPIU	Trace Port Interface
TSGEN	Time Stamp Generator
Tx	Transmit
TZASC	TrustZone Address Space Controller
UART	Universal asynchronous receiver/transmitter
USB	Universal serial bus
USDHC	Ultra Secured Digital Host Controller
WDOG	Watchdog
WLAN	Wireless Local Area Network
WXGA	Wide Extended Graphics Array

1.2 Introduction

This chapter introduces the architecture of the i.MX 6UltraLite Multimedia Applications Processor. The i.MX 6UltraLite processor represents Freescale Semiconductor's latest achievement in integrated multimedia applications processors.

It is part of a growing i.MX family of multimedia-focused products, offering high performance processing optimized for the lowest power consumption.

1.3 Target Applications

The architecture's flexibility allows for use in a wide variety of general embedded applications. The i.MX 6UltraLite processor provides all of the interfaces necessary for connecting peripherals such as WLAN, Bluetooth™, GPS, camera sensors, and multiple displays.

1.4 Features

The i.MX 6UltraLite processors are based on ARM®Cortex®-A7 MPCore™ Platform, which has the following features:

- Single ARM Cortex-A7 MPCore (with TrustZone) with:
  - 32 KB L1 Instruction Cache
  - 32 KB L1 Data Cache
  - Cortex-A7 NEON MPE (Media Processing Engine) Co-processor

The ARM Cortex-A7 MPCore complex includes:

- General Interrupt Controller (GIC) with 128 interrupts support
- Snoop Control Unit (SCU)
- 128 KB unified I/D L2 cache
- Single Master AXI (128-bit) bus interface output of L2 cache
- NEON MPE coprocessor
  - SIMD Media Processing Architecture
  - NEON register file with 32x32-bit, 32x64-bit, and 16x128-bit general-purpose registers
  - NEON Integer execute pipeline (ALU, Shift, MAC)
  - NEON dual, single-precision floating point execute pipeline (FADD, FMUL)
  - NEON load/store and permute pipeline
  - Supports single and double-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations as described in the ARM VFPv4 architecture.
  - Provides conversions between 16-bit, 32-bit, and 64-bit floating-point formats and ARM integer word formats

Target frequency of the core (including Neon and L1 cache) is 528 MHz non-overdrive over the specified temperature range:

- 528 MHz non-overdrive over the specified temperature range (Please refer to the *i.MX 6UltraLite Applications Processors for Consumer Products Data Sheet*).

The i.MX 6UltraLite processors support high volume, cost effective handheld DRAM, NOR Flash, and SD memory card standards.

The memory system consists of the following components:
- Level 1 Cache—32 KB Instruction, 32 KB Data cache
- Level 2 Cache—Unified instruction and data (128 KB)
- On-Chip Memory:
  - Boot ROM, including High Assurance Boot (HAB, 96 KB)
  - Internal fast access RAM (OCRAM, 128 KB)
  - Secure / non-Secure RAM (32KB)
- External memory interfaces:
  - 16-bit LP-DDR2, 16-bit DDR3, and LV-DDR3
  - 8-bit NAND-Flash, including support for Raw MLC/TLC, 2 KB ,4KB and 8KB page size, BA-NAND, PBA-NAND, LBA-NAND, OneNAND™ and others
  - BCH ECC up to 40 bits
  - 16-bit NOR Flash
  - 16-bit PSRAM, Cellular RAM
  - One QuadSPI with dual channel capability

The i.MX 6UltraLite processor enables the following interfaces to external devices (some of them are muxed and not available simultaneously):

Displays:
- eLCDIF supporting one parallel 24-bit LCD display with up to WXGA (1366x768) resolution at 60 Hz

Camera sensors:
- One Parallel Camera port (up to 24 bit and up to 148.5 MHz peak)

Expansion cards: Four MMC/SD/SDIO card ports all supporting:
- 1-bit or 4-bit transfer mode specifications for SD and SDIO cards up to UHS-I SDR-104 mode (104 MB/s max)
- 1-bit, 4-bit, or 8-bit transfer mode specifications for MMC cards up to 52 MHz in both SDR and DDR modes (104 MB/s max)
- 4-bit or 8-bit transfer mode specification for eMMC chips up to 200 MHz in HS200 mode (200MB/s max)

USB:
- Two High Speed (HS) USB 2.0 OTG (Up to 480 Mbps), with integrated HS USB Phy

Miscellaneous IPs and interfaces:
- Three I2S/SAI/AC97

- Eight UARTs operating up to 5.0 Mbps each, providing RS232 interface and supporting 9-bit RS485 multidrop mode. Eight UARTs support 4-wire mode.
- Four eCSPI (Enhanced CSPI), three of them can support up to 52 Mbps, one can be of low speed
- Four I2C supporting 400 kHz
- Dual-Megabit Ethernet Controller (IEEE1588 compliant), 10/100 Mbps
- Eight Pulse Width Modulators (PWM)
- System JTAG Controller (SJC)
- GPIO with interrupt capabilities
- 8x8 Key Pad Port (KPP)
- Sony Philips Digital Interface (SPDIF), Rx and Tx
- Two Controller Area Network (FlexCAN), 1 Mbps each
- Three Watchdog timers (WDOG)
- Asynchronous Sample Rate Converter (ASRC)
- Medium Quality Sound (MQS)

The i.MX 6UltraLite processors integrate advanced power management unit and controllers:

- PMU includes multiple LDO supplies for on-chip resources
- Temperature Sensor for monitoring die temperature
- DVFS to support low power modes
- SW State Retention and Power Gating for ARM and MPE
- Multiple power modes
- Flexible clock gating control scheme

The use of HW accelerators is a key factor in obtaining high performance at low power consumption numbers, while having the CPU core relatively free for performing other tasks.

The i.MX 6UltraLite processors incorporate the following hardware accelerators:

- Pixel Processing Pipeline (PXP)

Security functions are enabled and accelerated by the following hardware:

- ARM TrustZone including the TZ architecture (separation of interrupts, memory mapping, etc.)
- SJC—System JTAG Controller. Protecting JTAG from debug port attacks by regulating or blocking the access to the system debug features.
- CAAM—Cryptographic Acceleration and Assurance Module, containing cryptographic and hash engines, 16 KB secure RAM, and True and Pseudo Random Number Generator (NIST certified).
- SNVS—Secure Non-Volatile Storage, including Secure Real Time Clock.

- CSU—Central Security Unit. Enhancement for the IC Identification Module (IIM). Will be configured during boot and by eFUSEs and will determine the security level operation mode as well as the TZ policy.
- A-HAB—Advanced High Assurance Boot—HABv4 with the new embedded enhancements: SHA-256, 2048-bit RSA key, version control mechanism, warm boot, CSU, and TZ initialization.
- DryICE—Blocking a physical attack by manipulating operating temperature, voltage, and frequency.[1]

NOTE

To see supported features, please refer to the *i.MX 6UltraLite Applications Processors for Consumer Products Data Sheet*.

This feature set represents the full-featured derivative of the i.MX 6UltraLite and other derivatives do not have the full set.

1.5 Architectural Overview

This section contains the chip architectural details.

1.5.1 Simplified Block Diagram

A high level block diagram is shown in the figure below. This diagram provides a view of the chip's major sub-systems(processor domains, shared peripherals domain, and memories) and logical connectivity.

1. It is only available on the derivative G3.

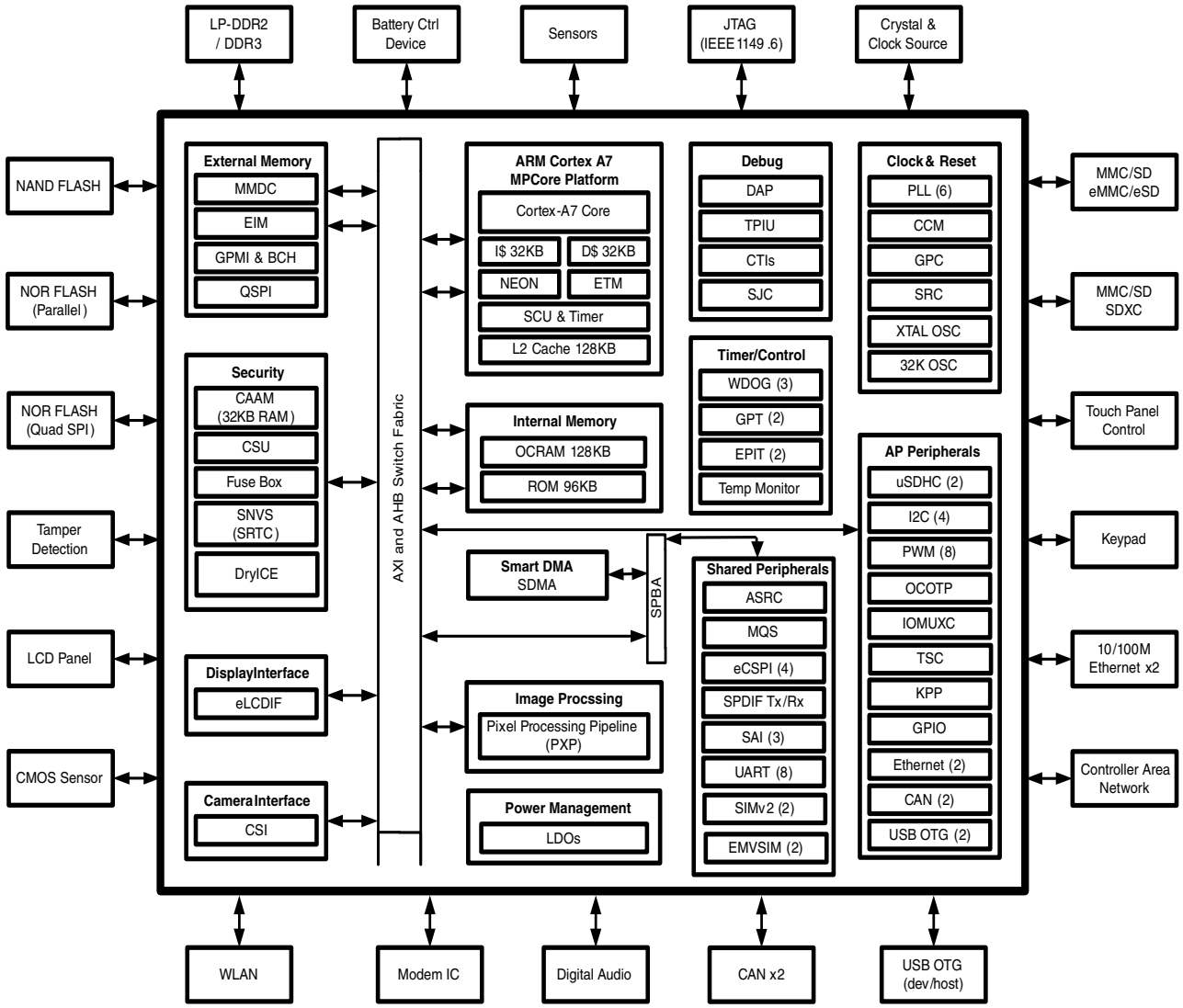


Figure 1-2. Simplified Block Diagram

1.5.2 Architectural Partitioning

Architecture supports processing-intensive tasks in the following ways:

- ARM Cortex A7 MPCore™ Platform provides hardware features for:
  - Operating System
  - User applications (including control over hardware accelerators and non-accelerated functions)
  - TrustZone applications

- Smart DMA enables data transfer between non-mastering peripherals and external or internal memories
- System Control is supported via:
  - Clock Control Module (CCM)
  - XTALOSC- 24 MHz Crystal oscillator source support
  - OSC32KHz - 32.768 Hz Crystal oscillator source support
  - System Reset Controller (SRC)
  - General Power Controller (GPC)
  - Temperature Sensor for monitoring and alarming on high temperature situations.
- Multimedia is supported with:
  - Audio
    - Audio codecs are provided by SW, which runs on ARM core, supporting (but not limited to) MP3, WMA, AAC, HE-AAC and Pro10
    - SPDIF Tx/Rx
    - Audio sample rate conversion accelerator (ASRC)
    - Medium Quality Sound (MQS)
  - Video
    - Display/graphics
    - Pixel Pipeline (PXP)
    - LCD interface (LCDIF)
    - CMOS sensor interface
- Security is supported by:
  - High Assurance Boot (HAB4) System
  - ARM TrustZone (TZ) Trusted Execution environment
  - Peripheral access policy control, using Central Security Unit (CSU)
  - System JTAG controller (SJC)
  - Secure Real Time Clock (SRTC)
  - TrustZone Watchdog (TZ WDOG)
  - DryICE
- Connectivity peripherals, timers and External Memory Interfaces:
  - Embedded DMAs
  - 3.3V IO voltage for seamless integration
  - Two USB 2.0 ports, including two PHYs: 2x HS-USB (OTG)
  - Nand-Flash (MLC up to 40-bit ECC) and NOR Flash memory interface via GPMI Nand-Flash controller
  - Timers: 2xEPIT, GPT and two Watch Dog timers (one of which is used for TZ), in addition to the timers and watchdog timers integrated within the ARM Cortex A7 MPCore™ platform.
  - Miscellaneous connectivity support - FLEXCAN, MMC/SD, I2C, SPI, UART, PWM and Keypad interface

1.5.3 Endianness Support

The chip supports Little Endian mode only.

1.5.4 Memory Interfaces

The chip DDR controller supports the following memory interfaces:

The EIM block provides an interface for SRAM and PSRAM, and 16/8-bit NOR Flash. All EIM pins are muxed on other interfaces.

Chapter 2
Memory Maps

2.1 Memory system overview

This chapter introduces the memory architecture of the chip.

2.2 ARM Platform Memory Map

The chip memory map has been provided in the following tables.

Table 2-1. System memory map

Start address	End address	Size	Description
8000_0000	FFFF_FFFF	2048 MB	MMDC - x16 DDR Controller.
7000_0000	7FFF_FFFF	256 MB	Reserved
6000_0000	6FFF_FFFF	256 MB	QSPI1 Memory
5800_0000	5FFF_FFFF	128 MB	EIM Aliased
5000_0000	57FF_FFFF	128 MB	EIM CS (NOR/SRAM)
1000_0000	4FFF_FFFF	1024 MB	AES Encryption
0E00_0000	0FFF_FFFF	32 MB	Reserved
0C00_0000	0DFF_FFFF	32 MB	QSPI1 Rx Buffer
0900_0000	0BFF_FFFF	48 MB	Reserved
0800_0000	08FF_FFFF	16 MB	Reserved
02C0_0000	07FF_FFFF	84 MB	Reserved
0230_0000	02BF_FFFF	9 MB	Reserved
0220_0000	022F_FFFF	1MB	Reserved
0210_0000	021F_FFFF	1 MB	Table 2-3 AIPS-2. See IP listing on separate map.
0200_0000	020F_FFFF	1 MB	Table 2-2 AIPS-1. See IP listing on separate map.
0181_0000	01FF_FFFF	8128 KB	Reserved
0180_C000	0180_FFFF	16KB	Reserved
0180_4000	0180_BFFF	32KB	APBH DMA

Table continues on the next page...

Table 2-1. System memory map (continued)

Start address	End address	Size	Description
0180_0000	0180_3FFF	16KB	Reserved
0120_0000	017F_FFFF	6 MB	Reserved
0110_0000	011F_FFFF	1 MB	Reserved
0100_0000	010F_FFFF	1 MB	Reserved
00F0_0000	00FF_FFFF	1 MB	Reserved
00E0_0000	00EF_FFFF	1 MB	(per_m) configuration port
00D0_0000	00DF_FFFF	1 MB	(cpu) configuration port
00C0_0000	00CF_FFFF	1 MB	GPV_1 PL301 configuration port
00B0_0000	00BF_FFFF	1 MB	GPV_0 PL301 configuration port
00A0_8000	00AF_FFFF	992 KB	Reserved
00A0_0000	00A0_7FFF	32 KB	ARM Peripherals: GIC400 Only visible to ARM core(s)
009C_0000	009F_FFFF	256 KB	Reserved
0098_0000	009B_FFFF	256 KB	Reserved
0092_0000	0097_FFFF	384 KB	OCRAM aliased
0090_0000	0091_FFFF	128 KB	OCRAM 128 KB
008F_8000	008F_FFFF	32 KB	Reserved
007F_8000	008F_7FFF	1 MB	Reserved
0011_0000	007F_7FFF	7072 KB	Reserved
0010_8000	0010_FFFF	32 KB	CAAM (Reserved)
0010_0000	0010_7FFF	32 KB	CAAM (32K secure RAM)
0001_8000	000F_FFFF	928 KB	Reserved
0001_7000	0001_7FFF	4 KB	Boot ROM - Protected 4 KB area
0000_0000	0001_6FFF	92 KB	Boot ROM (ROMCP)

Table 2-2 shows the ARM IP Bus (AIPS)-1 detailed memory map.

Table 2-2. AIPS-1 memory map

Start Address	End Address	Region	NIC Port	Size
020F_C000	020F_FFFF	AIPS-1	PWM8	16 KB
020F_8000	020F_BFFF		PWM7	16 KB
020F_4000	020F_7FFF		PWM6	16 KB
020F_0000	020F_3FFF		PWM5	16 KB
020E_C000	020E_FFFF		SDMA	16 KB
020E_8000	020E_BFFF		GPT2	16 KB
020E_4000	020E_7FFF		IOMUXC_GPR	16 KB
020E_0000	020E_3FFF		IOMUXC	16 KB
020D_C000	020D_FFFF		GPC	16 KB
020D_8000	020D_BFFF		SRC	16 KB
020D_4000	020D_7FFF		EPIT2	16 KB

Table continues on the next page...

Table 2-2. AIPS-1 memory map (continued)

Start Address	End Address	Region	NIC Port	Size
020D_0000	020D_3FFF	AIPS-1	EPIT1	16 KB
020C_C000	020C_FFFF		SNVS_HP	16 KB
020C_8000	020C_8FFF		ANALOG_DIG	16 KB
020C_4000	020C_7FFF		CCM	16 KB
020C_0000	020C_3FFF		WDOG2	16 KB
020B_C000	020B_FFFF		WDOG1	16 KB
020B_8000	020B_BFFF		KPP	16 KB
020B_4000	020B_7FFF		ENET2	16 KB
020B_0000	020B_3FFF		SNVS_LP	16 KB
020A_C000	020A_FFFF		GPIO5	16 KB
020A_8000	020A_BFFF		GPIO4	16 KB
020A_4000	020A_7FFF		GPIO3	16 KB
020A_0000	020A_3FFF		GPIO2	16 KB
0209_C000	0209_FFFF		GPIO1	16 KB
0209_8000	0209_BFFF		GPT1	16 KB
0209_4000	0209_7FFF		CAN2	16 KB
0209_0000	0209_3FFF		CAN1	16 KB
0208_C000	0208_FFFF		PWM4	16 KB
0208_8000	0208_BFFF		PWM3	16 KB
0208_4000	0208_7FFF		PWM2	16 KB
0208_0000	0208_3FFF		PWM1	16 KB
0207_C000	0207_FFFF		AIPS-1 Configuration	16 KB
0204_8000	0207_BFFF	AIPS-1 Glob. Module enable	Reserved	208KB
0204_4000	0204_7FFF		BEE	16 KB
0204_0000	0204_3FFF		TOUCH_CTRL	16 KB
0203_C000	0203_FFFF	AIPS-1 (via SPBA) Glob,Module ENABLE	SPBA	16 KB
0203_8000	0203_BFFF		Reserved for SDMA internal registers	16 KB
0203_4000	0203_7FFF		ASRC	16 KB
0203_0000	0203_3FFF		SAI3	16 KB
0202_C000	0202_FFFF		SAI2	16 KB
0202_8000	0202_BFFF		SAI1	16 KB
0202_4000	0202_7FFF		UART8	16 KB
0202_0000	0202_3FFF		UART1	16 KB
0201_C000	0201_FFFF		Reserved for SDMA internal registers	16 KB
0201_8000	0201_BFFF		UART7	16 KB
0201_4000	0201_7FFF		eCSPI4	16 KB
0201_0000	0201_3FFF		eCSPI3	16 KB
0200_C000	0200_FFFF		eCSPI2	16KB

Table continues on the next page...

Table 2-2. AIPS-1 memory map (continued)

Start Address	End Address	Region	NIC Port	Size
0200_8000	0200_BFFF		eCSPI1	16 KB
0200_4000	0200_7FFF		SPDIF	16 KB
0200_0000	0200_3FFF		Reserved for SDMA internal registers	16 KB

Table 2-3 shows the AIPS-2 detailed memory map.

Table 2-3. AIPS-2 memory map

Start Address	End Address	Region	Allocation	Size
021F_C000	021F_FFFF		UART6	16 KB
021F_8000	021F_BFFF		I2C4	16 KB
021F_4000	021F_7FFF		UART5	16 KB
021F_0000	021F_3FFF		UART4	16 KB
021E_C000	021E_FFFF		UART3	16 KB
021E_8000	021E_BFFF		UART2	16 KB
021E_4000	021E_7FFF		WDOG3	16 KB
021E_0000	021E_3FFF		QSPI	16 KB
021D_C000	021D_FFFF		System Counter_CTRL	16 KB
021D_8000	021D_BFFF	AIPS-2	System Counter_CMP	16 KB
021D_4000	021D_7FFF		System Counter_RD	16 KB
021D_0000	021D_3FFF		TZASC	16 KB
021C_C000	021C_FFFF		PXP	16 KB
021C_8000	021C_BFFF		LCDIF	16 KB
021C_4000	021C_7FFF		CSI	16 KB
021C_0000	021C_3FFF		CSU	16 KB
021B_C000	021B_FFFF		OCOTP_CTRL	16 KB
021B_8000	021B_BFFF		EIM	16 KB
021B_4000	021B_7FFF		SIM2	16 KB
021B_0000	021B_3FFF		MMDC	16 KB
021A_C000	021A_FFFF		ROMCP	16 KB
021A_8000	021A_BFFF		I2C3	16 KB
021A_4000	021A_7FFF		I2C2	16 KB
021A_0000	021A_3FFF		I2C1	16 KB
0219_C000	0219_FFFF	AIPS-2	ADC2	16 KB
0219_8000	0219_BFFF		ADC1	16 KB
0219_4000	0219_7FFF		uSDHC2	16 KB
0219_0000	0219_3FFF		uSDHC1	16 KB
0218_C000	0218_FFFF		SIM1	16 KB
0218_8000	0218_BFFF		ENET1	16 KB

Table continues on the next page...

Table 2-3. AIPS-2 memory map (continued)

Start Address	End Address	Region	Allocation	Size
0218_4000	0218_7FFF		USBO2 (USB)	16 KB
0218_0000	0218_3FFF		USBO2 (pl301)	16 KB
0217_C000	0217_FFFF		AIPS-2 configuration	16 KB
0214_0000	0217_BFFF		CAAM	240KB
0210_0000	0213_FFFF		CA7/DAPTable 2-4	256KB

Table 2-4 shows the Debug Access Port (DAP) detailed memory map.

Table 2-4. DAP memory map

Start Address	End Address	Region	Allocation	Size
0213_F000	0213_FFFF		Reserved	4 KB
0213_E000	0213_EFFF		Reserved	4 KB
0213_D000	0213_DFFF		Reserved	4 KB
0213_C000	0213_CFFF		ETM0	4 KB
0213_B000	0213_BFFF		Reserved	4 KB
0213_A000	0213_AFFF		Reserved	4 KB
0213_9000	0213_9FFF		Reserved	4 KB
0213_8000	0213_8FFF		CTI0	4 KB
0213_7000	0213_7FFF		Reserved	4 KB
0213_6000	0213_6FFF		Reserved	4 KB
0213_5000	0213_5FFF		Reserved	4 KB
0213_4000	0213_4FFF		Reserved	4 KB
0213_3000	0213_3FFF		Reserved	4 KB
0213_2000	0213_2FFF	CA7_DAP	Reserved	4 KB
0213_1000	0213_1FFF		CPU0 PMU	4 KB
0213_0000	0213_0FFF		CPU0 Debug	4 KB
0212_1000	0212_FFFF		Reserved	60KB
0212_0000	0212_0FFF		CA7 ROM Table	4 KB
0212_8000	0212_FFFF		Reserved	96KB
0210_7000	0210_7FFF		Reserved	4 KB
0210_6000	0210_6FFF		Reserved	4 KB
0210_5000	0210_5FFF		Reserved	4 KB
0210_4000	0210_4FFF		TSGEN	4 KB
0210_3000	0210_3FFF		TPIU	4 KB
0210_2000	0210_2FFF		CTI	4 KB
0210_1000	0210_1FFF		ETF	4 KB
0210_0000	0210_0FFF		DAP ROM Table	4 KB

NOTE

Access to reserved memory regions can cause unpredictable behavior.

2.3 DMA memory map

The Smart DMA memory map can be found in the following table.

Table 2-5. SDMA peripheral memory map

Peripheral	Base address	Size
Reserved for SDMA internal memory	0x0000	4KB
SPDIF	0x1000	4KB
eCSPI1	0x2000	4KB
eCSPI2	0x3000	4KB
eCSPI3	0x4000	4KB
eCSPI4	0x5000	4KB
UART7	0x6000	4KB
Reserved for SDMA internal registers	0x7000	4KB
UART1	0x8000	4KB
UART8	0x9000	4KB
SAI1	0xA000	4KB
SAI2	0xB000	4KB
SAI3	0xC000	4KB
ASRC	0xD000	4KB
Reserved	0xE000	4KB
SPBA Registers	0xF000	4KB

NOTE

Access to reserved memory regions can cause unpredictable behavior.

Chapter 3
Interrupts and DMA Events

3.1 Overview

This chapter discusses the assignments of interrupts from the ARM domain in Cortex A7 interrupts and from DMA events in SDMA event mapping

3.2 Cortex A7 interrupts

The Global Interrupt Controller (GIC) collects up to 128 interrupt requests from all chip sources and provides an interface to the Cortex A7 CPU. The first 32 interrupts are private to the CPUs interface. These interrupts are not included in the table below. All interrupts besides the ones private to the CPU are hooked up to the GPC.

Each interrupt can be configured as a normal or a secure interrupt. Software force registers and software priority masking are also supported. The following table describes the ARM Cortex A7 interrupt sources.

Table 3-1. ARM Cortex A7 domain interrupt summary

IRQ	Interrupt Source	Interrupt Description
32	IOMUXC	General Purpose Register 1 from IOMUXC. Used to notify cores on exception condition while boot.
33	DAP	Debug Access Port interrupt request.
34	SDMA	SDMA interrupt request from all channels.
35	TSC	TSC interrupt.
36	SNVS_LP / SNVS_HP	Logic OR of SNVS_LP and SNVS_HP interrupts.
37	LCDIF	LCDIF sync interrupt.
38	BEE	BEE interrupt.
39	CSI	CMOS Sensor Interface interrupt request.

Table continues on the next page...

Table 3-1. ARM Cortex A7 domain interrupt summary (continued)

IRQ	Interrupt Source	Interrupt Description
40	PXP	PXP interrupt.
41	SCTR	SCTR compare interrupt.
42	SCTR	SCTR compare interrupt.
43	WDOG3	WDOG3 timer reset interrupt request.
44	Reserved	Reserved
45	APBH DMA	Logical OR of APBH DMA channels 0-3 completion and error interrupts.
46	WEIM	WEIM interrupt request.
47	RAWNAND	BCH operation complete interrupt.
48	RAWNAND	GPMI operation timeout error interrupt.
49	UART6	UART6 interrupt request.
50	Reserved	Reserved
51	SNVS	SRTC consolidated interrupt.
52	SNVS	SRTC security interrupt.
53	CSU	CSU interrupt request 1. Indicates to the processor that one or more alarm inputs were asserted.
54	uSDHC1	uSDHC1 (Enhanced SDHC) interrupt request.
55	uSDHC2	uSDHC2 (Enhanced SDHC) interrupt request.
56	SAI3	SAI interrupt
57	SAI3	SAI interrupt.
58	UART1	UART1 interrupt request.
59	UART2	UART2 interrupt request.
60	UART3	UART3 interrupt request.
61	UART4	UART4 interrupt request.
62	UART5	UART5 interrupt request.
63	eCSPI1	eCSPI1 interrupt request.
64	eCSPI2	eCSPI2 interrupt request.
65	eCSPI3	eCSPI3 interrupt request.
66	eCSPI4	eCSPI4 interrupt request.
67	I2C4	I2C4 interrupt request.
68	I2C1	I2C1 interrupt request.
69	I2C2	I2C2 interrupt request.
70	I2C3	I2C3 interrupt request.
71	UART7	UART-7 ORed interrupt.
72	UART8	UART-8 ORed interrupt.
73	Reserved	Reserved
74	USB	USBO2 USB OTG2
75	USB	USBO2 USB OTG1
76	USB_PHY	UTMI0 interrupt request.
77	USB_PHY	UTMI1 interrupt request.

Table continues on the next page...

Table 3-1. ARM Cortex A7 domain interrupt summary (continued)

IRQ	Interrupt Source	Interrupt Description
78	CAAM	CAAM interrupt queue for JQ2.
79	CAAM	CAAM interrupt for recoverable error.
80	CAAM	CAAM interrupt for RTIC.
81	Temperature Monitor	Temperature Sensor (temperature greater than threshold) interrupt request.
82	ASRC	ASRC interrupt request.
83	Reserved	Reserved
84	SPDIF	Logic OR of SPDIF Rx/Tx interrupts.
85	Reserved	Reserved
86	PMU	Brown-out event on either the 1.1, 2.5 or 3.0 regulators.
87	GPT1	Logical OR of GPT1 rollover interrupt line, input capture 1 & 2 lines, output compare 1, 2, and 3 interrupt lines.
88	EPIT1	EPIT1 output compare interrupt.
89	EPIT2	EPIT2 output compare interrupt.
90	GPIO1	INT7 interrupt request.
91	GPIO1	INT6 interrupt request.
92	GPIO1	INT5 interrupt request.
93	GPIO1	INT4 interrupt request.
94	GPIO1	INT3 interrupt request.
95	GPIO1	INT2 interrupt request.
96	GPIO1	INT1 interrupt request.
97	GPIO1	INT0 interrupt request.
98	GPIO1	Combined interrupt indication for GPIO1 signals 0 - 15.
99	GPIO1	Combined interrupt indication for GPIO1 signals 16 - 31.
100	GPIO2	Combined interrupt indication for GPIO2 signals 0 - 15.
101	GPIO2	Combined interrupt indication for GPIO2 signals 16 - 31.
102	GPIO3	Combined interrupt indication for GPIO3 signals 0 - 15.
103	GPIO3	Combined interrupt indication for GPIO3 signals 16 - 31.
104	GPIO4	Combined interrupt indication for GPIO4 signals 0 - 15.
105	GPIO4	Combined interrupt indication for GPIO4 signals 16 - 31.
106	GPIO5	Combined interrupt indication for GPIO5 signals 0 - 15.
107	GPIO5	Combined interrupt indication for GPIO5 signals 16 - 31.
108	Reserved	Reserved
109	Reserved	Reserved
110	Reserved	Reserved
111	Reserved	Reserved
112	WDOG1	WDOG1 timer reset interrupt request.
113	WDOG2	WDOG2 timer reset interrupt request.
114	KPP	Key Pad interrupt request.

Table continues on the next page...

Table 3-1. ARM Cortex A7 domain interrupt summary (continued)

IRQ	Interrupt Source	Interrupt Description
115	PWM1	Cumulative interrupt line for PWM1. Logical OR of rollover, compare, and FIFO waterlevel crossing interrupts.
116	PWM2	Cumulative interrupt line for PWM2. Logical OR of rollover, compare, and FIFO waterlevel crossing interrupts.
117	PWM3	Cumulative interrupt line for PWM3. Logical OR of rollover, compare, and FIFO waterlevel crossing interrupts.
118	PWM4	Cumulative interrupt line for PWM4. Logical OR of rollover, compare, and FIFO waterlevel crossing interrupts.
119	CCM	CCM interrupt request 1.
120	CCM	CCM interrupt request 2.
121	GPC	GPC interrupt request 1.
122	Reserved	Reserved
123	SRC	SRC interrupt request.
124	Reserved	Reserved
125	Reserved	Reserved
126	CPU	Performance Unit interrupt.
127	CPU	CTI trigger outputs interrupt.
128	SRC	Combined CPU wdog interrupts (4x) out of SRC.
129	SAI1	SAI1 interrupt request.
130	SAI2	SAI2 interrupt request.
131	Reserved	Reserved
132	ADC1	ADC1 interrupt request.
133	ADC2	ADC2 interrupt request.
134	Reserved	Reserved
135	Reserved	Reserved
136	SJC	SJC interrupt from General Purpose register.
137	CAAM	CAAM job ring 0 interrupt.
138	CAAM	CAAM job ring 1 interrupt.
139	QSPI	QSPI interrupt request.
140	TZASC	TZASC (PL380) interrupt request.
141	GPT2	Logical OR of GPT2 rollover interrupt line, input capture 1 & 2 lines, output compare 1, 2 & 3 interrupt lines.
142	CAN1	Combined interrupt of ini_int_busoff,ini_int_error,ipi_int_mbor,ipi_int_txwarning and ipi_int_waken
143	CAN2	Combined interrupt of ini_int_busoff,ini_int_error,ipi_int_mbor,ipi_int_txwarning and ipi_int_waken
144	SIM1	SIM interrupt request.
145	SIM2	SIM interrupt request.
146	PWM5	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line

Table continues on the next page...

Table 3-1. ARM Cortex A7 domain interrupt summary (continued)

IRQ	Interrupt Source	Interrupt Description
147	PWM6	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
148	PWM7	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
149	PWM8	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
150	ENET1	ENET1 interrupt
151	ENET1	ENET1 1588 Timer interrupt [synchronous] request.
152	ENET2	ENET2 interrupt
153	ENET2	MAC 0 1588 Timer interrupt [synchronous] request.
154	Reserved	Reserved
155	Reserved	Reserved
156	Reserved	Reserved
157	Reserved	Reserved
158	Reserved	Reserved
159	PMU	Brown-out event on either core, gpu or soc regulators.

3.3 SDMA event mapping

The following table found here shows the DMA request signals for peripherals in the chip.

Table 3-2. SDMA event mapping

Event Number	DMA Source	Description
0	UART6 / SIM2	UART6 Rx FIFO; SIM2 receive DMA request
1	ADC1	ADC1 DMA request
2	EPIT2 / PXP	EPIT2 DMA request; PXP DMA Event
3	eCSPI1	eCSPI1 Rx request
4	eCSPI1	eCSPI1 Tx request
5	eCSPI2	eCSPI2 Rx request
6	eCSPI2	eCSPI2 Tx request
7	eCSPI3 / I2C1	eCSPI3 Rx request; I2C1 DMA event
8	eCSPI3 / I2C2	eCSPI3 Tx request; I2C2 DMA event
9	eCSPI4 / I2C3	eCSPI4 Rx request; I2C3 DMA event
10	eCSPI4 / I2C4	eCSPI4 Tx request; I2C4 DMA event
11	QSPI	QSPI DMA RX request

Table continues on the next page...

Table 3-2. SDMA event mapping (continued)

Event Number	DMA Source	Description
12	QSPI	QSPI DMA TX request
13	ADC2 / TSC	ADC2 DMA request; TSC interrupt
14	Mux_bottom	external DMA pad #1
15	Mux_bottom	external DMA pad #2
16	EPIT1 / CSI	EPIT1 DMA request: CSI DMA Event.
17	ASRC	ASRC DMA1 request (Pair A input Request)
18	ASRC	ASRC DMA2 request (Pair B input Request)
19	ASRC	ASRC DMA3 request (Pair C input Request)
20	ASRC	ASRC DMA4 request (Pair A output Request)
21	ASRC	ASRC DMA5 request (Pair B output Request)
22	ASRC	ASRC DMA6 request (Pair C output Request)
23	GPT1	GPT1 counter event
24	GPT2 / LCDIF	GPT2 counter event; LCDIF DMA Event
25	UART1	UART1 Rx FIFO
26	UART1	UART1 Tx FIFO
27	UART2	UART2 Rx FIFO
28	UART2	UART2 Tx FIFO
29	UART3	UART3 Rx FIFO
30	UART3	UART3 Tx FIFO
31	UART4	UART4 Rx FIFO
32	UART4	UART4 Tx FIFO
33	UART5 / SIM1	UART5 Rx FIFO; SIM1 receive DMA request
34	UART5 / SIM1	UART5 Tx FIFO; SIM1 transmit DMA request
35	SAI1	SAI1 Rx FIFO
36	SAI1	SAI1 Tx FIFO
37	SAI2	SAI2 Rx FIFO
38	SAI2	SAI2 Tx FIFO
39	SAI3	SAI3 Rx FIFO
40	SAI3	SAI3 Rx FIFO
41	SPDIF	SPDIF Rx DMA request
42	SPDIF	SPDIF Tx DMA request
43	UART7 / ENET1	UART7 Rx FIFO; ENET1 1588 Event0 out
44	UART7 / ENET1	UAR7 Tx FIFO; ENET1 1588 Event1 out
45	UART8 / ENET2	UART8 Rx FIFO; ENET2 1588 Event0 out
46	UART8 / ENET2	UART8 Tx FIFO; ENET2 1588 Event1 out
47	UART6 / SIM2	UART6 Tx FIFO; SIM2 transmit DMA request UART6 Tx FIFO; ESAI Tx FIFO DMA request

As shown in the table, some of the events are an output of a mux of two signals or triggers. The select of this mux is controlled by the general purpose registers in IOMUXC.

Chapter 4
External Signals and Pin Multiplexing

4.1 Overview

The chip contains a limited number of pins, most of which have multiple signal options. These signal to pin and pin to signal options are selected by the input-output multiplexer called IOMUX. The IOMUX is also used to configure other pin characteristics, such as voltage level, drive strength, and hysteresis.

The muxing options table lists the external signals grouped by module instance, the muxing options for each signal, and the registers used to route the signal to the chosen pad.

4.1.1 Muxing Options

Instance	Port	Pad	Mode
ADC	ADC_VREFH	ADC_VREFH	-
	ADC_VREFL	GND	-
	ADC1_IN0	GPIO1_IO00	-
	ADC1_IN1	GPIO1_IO01	-
	ADC1_IN2	GPIO1_IO02	-
	ADC1_IN3	GPIO1_IO03	-
	ADC1_IN4	GPIO1_IO04	-
	ADC1_IN5	GPIO1_IO05	-
	ADC1_IN6	GPIO1_IO06	-
	ADC1_IN7	GPIO1_IO07	-
	ADC1_IN8	GPIO1_IO08	-
	ADC1_IN9	GPIO1_IO09	-
	ADC2_IN0	GPIO1_IO00	-
	ADC2_IN1	GPIO1_IO01	-
	ADC2_IN2	GPIO1_IO02	-

Table continues on the next page...

Instance	Port	Pad	Mode
	ADC2_IN3	GPIO1_IO03	-
	ADC2_IN4	GPIO1_IO04	-
	ADC2_IN5	GPIO1_IO05	-
	ADC2_IN6	GPIO1_IO06	-
	ADC2_IN7	GPIO1_IO07	-
	ADC2_IN8	GPIO1_IO08	-
	ADC2_IN9	GPIO1_IO09	-
ARM_PLATFORM	EVENTI	LCD_RESET	ALT2
	EVENTO	LCD_DATA18	ALT2
	TRACE0	LCD_DATA00	ALT2
	TRACE1	LCD_DATA01	ALT2
	TRACE2	LCD_DATA02	ALT2
	TRACE3	LCD_DATA03	ALT2
	TRACE4	LCD_DATA04	ALT2
	TRACE5	LCD_DATA05	ALT2
	TRACE6	LCD_DATA06	ALT2
	TRACE7	LCD_DATA07	ALT2
	TRACE8	LCD_DATA08	ALT2
	TRACE9	LCD_DATA09	ALT2
	TRACE10	LCD_DATA10	ALT2
	TRACE11	LCD_DATA11	ALT2
	TRACE12	LCD_DATA12	ALT2
	TRACE13	LCD_DATA13	ALT2
	TRACE14	LCD_DATA14	ALT2
	TRACE15	LCD_DATA15	ALT2
	TRACE_CLK	LCD_DATA16	ALT2
	TRACE_CTL	LCD_DATA17	ALT2
CCM	CLKO1	JTAG_TMS	ALT3
		SD1_DATA2	ALT6
	CLKO2	JTAG_TDO	ALT3
		SD1_DATA3	ALT6
	PMIC_READY	GPIO1_IO08	ALT6
		JTAG_MOD	ALT4
	PMIC_STBY_REQ	CCM_PMIC_STBY_REQ	No Muxing (ALT0)
	REF_EN_B	GPIO1_IO06	ALT7
CSI	DATA0	LCD_DATA17	ALT3
		UART3_RX_DATA	ALT3
	DATA1	LCD_DATA16	ALT3
		UART3_TX_DATA	ALT3
	DATA2	CSI_DATA00	ALT0
		UART1_TX_DATA	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
	DATA3	CSI_DATA01	ALT0
		UART1_RX_DATA	ALT3
	DATA4	CSI_DATA02	ALT0
		UART1_CTS_B	ALT3
	DATA5	CSI_DATA03	ALT0
		UART1_RTS_B	ALT3
	DATA6	CSI_DATA04	ALT0
		UART2_TX_DATA	ALT3
	DATA7	CSI_DATA05	ALT0
		UART2_RX_DATA	ALT3
	DATA8	CSI_DATA06	ALT0
		UART2_CTS_B	ALT3
	DATA9	CSI_DATA07	ALT0
		UART2_RTS_B	ALT3
	DATA10	LCD_DATA18	ALT3
		UART3_CTS_B	ALT3
	DATA11	LCD_DATA19	ALT3
		UART3_RTS_B	ALT3
	DATA12	LCD_DATA20	ALT3
		UART4_TX_DATA	ALT3
	DATA13	LCD_DATA21	ALT3
		UART4_RX_DATA	ALT3
	DATA14	LCD_DATA22	ALT3
		UART5_TX_DATA	ALT3
	DATA15	LCD_DATA23	ALT3
		UART5_RX_DATA	ALT3
	DATA16	ENET1_RX_DATA0	ALT3
		LCD_DATA08	ALT3
	DATA17	ENET1_RX_DATA1	ALT3
		LCD_DATA09	ALT3
	DATA18	ENET1_RX_EN	ALT3
		LCD_DATA10	ALT3
	DATA19	ENET1_TX_DATA0	ALT3
		LCD_DATA11	ALT3
	DATA20	ENET1_TX_DATA1	ALT3
		LCD_DATA12	ALT3
	DATA21	ENET1_TX_EN	ALT3
		LCD_DATA13	ALT3
	DATA22	ENET1_TX_CLK	ALT3
		LCD_DATA14	ALT3
	DATA23	ENET1_RX_ER	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
		LCD_DATA15	ALT3
	FIELD	GPIO1_IO05	ALT3
		NAND_DQS	ALT1
	HSYNC	CSI_HSYNC	ALT0
		GPIO1_IO09	ALT3
	MCLK	CSI_MCLK	ALT0
		GPIO1_IO06	ALT3
	PIXCLK	CSI_PIXCLK	ALT0
		GPIO1_IO07	ALT3
	VSYNC	CSI_VSYNC	ALT0
		GPIO1_IO08	ALT3
ECSPI1	MISO	CSI_DATA07	ALT3
		LCD_DATA23	ALT2
	MOSI	CSI_DATA06	ALT3
		LCD_DATA22	ALT2
	RDY	LCD_DATA12	ALT8
	SCLK	CSI_DATA04	ALT3
		LCD_DATA20	ALT2
	SS0	CSI_DATA05	ALT3
		LCD_DATA21	ALT2
	SS1	LCD_DATA05	ALT8
	SS2	LCD_DATA06	ALT8
	SS3	LCD_DATA07	ALT8
ECSPI2	MISO	CSI_DATA03	ALT3
		UART5_RX_DATA	ALT8
	MOSI	CSI_DATA02	ALT3
		UART5_TX_DATA	ALT8
	RDY	LCD_ENABLE	ALT8
	SCLK	CSI_DATA00	ALT3
		UART4_TX_DATA	ALT8
	SS0	CSI_DATA01	ALT3
		UART4_RX_DATA	ALT8
	SS1	LCD_HSYNC	ALT8
	SS2	LCD_VSYNC	ALT8
	SS3	LCD_RESET	ALT8
ECSPI3	MISO	NAND_CLE	ALT3
		UART2_RTS_B	ALT8
	MOSI	NAND_CE1_B	ALT3
		UART2_CTS_B	ALT8
	RDY	NAND_WP_B	ALT8
	SCLK	NAND_CE0_B	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
		UART2_RX_DATA	ALT8
	SS0	NAND_READY_B	ALT3
		UART2_TX_DATA	ALT8
	SS1	NAND_ALE	ALT8
	SS2	NAND_RE_B	ALT8
	SS3	NAND_WE_B	ALT8
ECSPI4	MISO	ENET2_TX_CLK	ALT3
		NAND_DATA06	ALT3
	MOSI	ENET2_TX_EN	ALT3
		NAND_DATA05	ALT3
	RDY	NAND_DATA00	ALT8
	SCLK	ENET2_TX_DATA1	ALT3
		NAND_DATA04	ALT3
	SS0	ENET2_RX_ER	ALT3
		NAND_DATA07	ALT3
	SS1	NAND_DATA01	ALT8
	SS2	NAND_DATA02	ALT8
	SS3	NAND_DATA03	ALT8
EIM	ACLK_FREERUN	ENET2_TX_EN	ALT4
	AD0	CSI_DATA00	ALT4
	AD1	CSI_DATA01	ALT4
	AD2	CSI_DATA02	ALT4
	AD3	CSI_DATA03	ALT4
	AD4	CSI_DATA04	ALT4
	AD5	CSI_DATA05	ALT4
	AD6	CSI_DATA06	ALT4
	AD7	CSI_DATA07	ALT4
	AD8	NAND_DATA00	ALT4
	AD9	NAND_DATA01	ALT4
	AD10	NAND_DATA02	ALT4
	AD11	NAND_DATA03	ALT4
	AD12	NAND_DATA04	ALT4
	AD13	NAND_DATA05	ALT4
	AD14	NAND_DATA06	ALT4
	AD15	NAND_DATA07	ALT4
	ADDR16	NAND_CLE	ALT4
	ADDR17	NAND_ALE	ALT4
	ADDR18	NAND_CE1_B	ALT4
	ADDR19	SD1_CMD	ALT4
	ADDR20	SD1_CLK	ALT4
	ADDR21	SD1_DATA0	ALT4

Table continues on the next page...

Instance	Port	Pad	Mode
	ADDR22	SD1_DATA1	ALT4
	ADDR23	SD1_DATA2	ALT4
	ADDR24	SD1_DATA3	ALT4
	ADDR25	ENET2_RX_ER	ALT4
	ADDR26	ENET2_RX_EN	ALT4
	BCLK	NAND_WP_B	ALT4
	CRE	ENET1_RX_ER	ALT4
	CS0_B	CSI_MCLK	ALT4
	CS1_B	NAND_READY_B	ALT4
	CS2_B	LCD_CLK	ALT4
	CS3_B	LCD_ENABLE	ALT4
	DATA0	LCD_DATA08	ALT4
	DATA1	LCD_DATA09	ALT4
	DATA2	LCD_DATA10	ALT4
	DATA3	LCD_DATA11	ALT4
	DATA4	LCD_DATA12	ALT4
	DATA5	LCD_DATA13	ALT4
	DATA6	LCD_DATA14	ALT4
	DATA7	LCD_DATA15	ALT4
	DATA8	LCD_DATA16	ALT4
	DATA9	LCD_DATA17	ALT4
	DATA10	LCD_DATA18	ALT4
	DATA11	LCD_DATA19	ALT4
	DATA12	LCD_DATA20	ALT4
	DATA13	LCD_DATA21	ALT4
	DATA14	LCD_DATA22	ALT4
	DATA15	LCD_DATA23	ALT4
	DTACK_B	NAND_CE0_B	ALT4
	EB0_B	NAND_RE_B	ALT4
	EB1_B	NAND_WE_B	ALT4
	EB2_B	ENET2_TX_DATA0	ALT4
	EB3_B	ENET2_TX_DATA1	ALT4
	LBA_B	CSI_HSYNC	ALT4
	OE	CSI_PIXCLK	ALT4
	RW	CSI_VSYNC	ALT4
	WAIT	NAND_DQS	ALT4
ENET1	1588_EVENT0_IN	GPIO1_IO00	ALT6
	1588_EVENT0_OUT	GPIO1_IO01	ALT6
	1588_EVENT1_IN	UART3_CTS_B	ALT4
	1588_EVENT1_OUT	UART3_RTS_B	ALT4
	1588_EVENT2_IN	LCD_DATA00	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
	1588_EVENT2_OUT	LCD_DATA01	ALT3
	1588_EVENT3_IN	LCD_DATA02	ALT3
	1588_EVENT3_OUT	LCD_DATA03	ALT3
	COL	UART2_RTS_B	ALT1
	CRS	UART2_CTS_B	ALT1
	MDC	ENET2_RX_DATA1	ALT4
		GPIO1_IO07	ALT0
	MDIO	ENET2_RX_DATA0	ALT4
		GPIO1_IO06	ALT0
	RDATA0	ENET1_RX_DATA0	ALT0
	RDATA1	ENET1_RX_DATA1	ALT0
	RDATA2	UART1_TX_DATA	ALT1
	RDATA3	UART1_RX_DATA	ALT1
	RX_CLK	UART1_CTS_B	ALT1
	RX_EN	ENET1_RX_EN	ALT0
	RX_ER	ENET1_RX_ER	ALT0
	TDATA0	ENET1_TX_DATA0	ALT0
	TDATA1	ENET1_TX_DATA1	ALT0
	TDATA2	UART2_TX_DATA	ALT1
	TDATA3	UART2_RX_DATA	ALT1
	TX_CLK	ENET1_TX_CLK	ALT0
	TX_EN	ENET1_TX_EN	ALT0
	TX_ER	UART1_RTS_B	ALT1
ENET2	1588_EVENT0_IN	GPIO1_IO04	ALT6
	1588_EVENT0_OUT	GPIO1_IO05	ALT6
	1588_EVENT1_IN	UART1_CTS_B	ALT4
	1588_EVENT1_OUT	UART1_RTS_B	ALT4
	1588_EVENT2_IN	LCD_DATA04	ALT3
	1588_EVENT2_OUT	LCD_DATA05	ALT3
	1588_EVENT3_IN	LCD_DATA06	ALT3
	1588_EVENT3_OUT	LCD_DATA07	ALT3
	COL	UART5_RX_DATA	ALT1
	CRS	UART5_TX_DATA	ALT1
	MDC	ENET1_TX_EN	ALT4
		GPIO1_IO07	ALT1
	MDIO	ENET1_TX_DATA1	ALT4
		GPIO1_IO06	ALT1
	RDATA0	ENET2_RX_DATA0	ALT0
	RDATA1	ENET2_RX_DATA1	ALT0
	RDATA2	UART3_TX_DATA	ALT1
	RDATA3	UART3_RX_DATA	ALT1

Table continues on the next page...

Instance	Port	Pad	Mode
	RX_CLK	UART3_CTS_B	ALT1
	RX_EN	ENET2_RX_EN	ALT0
	RX_ER	ENET2_RX_ER	ALT0
	TDATA0	ENET2_TX_DATA0	ALT0
	TDATA1	ENET2_TX_DATA1	ALT0
	TDATA2	UART4_TX_DATA	ALT1
	TDATA3	UART4_RX_DATA	ALT1
	TX_CLK	ENET2_TX_CLK	ALT0
	TX_EN	ENET2_TX_EN	ALT0
	TX_ER	UART3_RTS_B	ALT1
EPIT1	OUT	JTAG_TMS	ALT8
		UART3_RX_DATA	ALT8
EPIT2	OUT	JTAG_TDO	ALT8
		UART3_CTS_B	ALT8
FLEXCAN1	RX	ENET1_RX_DATA1	ALT4
		LCD_DATA09	ALT8
		SD1_DATA1	ALT3
		UART3_RTS_B	ALT2
	TX	ENET1_RX_DATA0	ALT4
		LCD_DATA08	ALT8
		SD1_DATA0	ALT3
		UART3_CTS_B	ALT2
FLEXCAN2	RX	ENET1_TX_DATA0	ALT4
		LCD_DATA11	ALT8
		SD1_DATA3	ALT3
		UART2_RTS_B	ALT2
	TX	ENET1_RX_EN	ALT4
		LCD_DATA10	ALT8
		SD1_DATA2	ALT3
		UART2_CTS_B	ALT2
GPIO1	IO0	GPIO1_IO00	ALT5
	IO1	GPIO1_IO01	ALT5
	IO2	GPIO1_IO02	ALT5
	IO3	GPIO1_IO03	ALT5
	IO4	GPIO1_IO04	ALT5
	IO5	GPIO1_IO05	ALT5
	IO6	GPIO1_IO06	ALT5
	IO7	GPIO1_IO07	ALT5
	IO8	GPIO1_IO08	ALT5
	IO9	GPIO1_IO09	ALT5
	IO10	JTAG_MOD	ALT5

Table continues on the next page...

Instance	Port	Pad	Mode
	IO11	JTAG_TMS	ALT5
	IO12	JTAG_TDO	ALT5
	IO13	JTAG_TDI	ALT5
	IO14	JTAG_TCK	ALT5
	IO15	JTAG_TRST_B	ALT5
	IO16	UART1_TX_DATA	ALT5
	IO17	UART1_RX_DATA	ALT5
	IO18	UART1_CTS_B	ALT5
	IO19	UART1_RTS_B	ALT5
	IO20	UART2_TX_DATA	ALT5
	IO21	UART2_RX_DATA	ALT5
	IO22	UART2_CTS_B	ALT5
	IO23	UART2_RTS_B	ALT5
	IO24	UART3_TX_DATA	ALT5
	IO25	UART3_RX_DATA	ALT5
	IO26	UART3_CTS_B	ALT5
	IO27	UART3_RTS_B	ALT5
	IO28	UART4_TX_DATA	ALT5
	IO29	UART4_RX_DATA	ALT5
	IO30	UART5_TX_DATA	ALT5
	IO31	UART5_RX_DATA	ALT5
GPIO2	IO0	ENET1_RX_DATA0	ALT5
	IO1	ENET1_RX_DATA1	ALT5
	IO2	ENET1_RX_EN	ALT5
	IO3	ENET1_TX_DATA0	ALT5
	IO4	ENET1_TX_DATA1	ALT5
	IO5	ENET1_TX_EN	ALT5
	IO6	ENET1_TX_CLK	ALT5
	IO7	ENET1_RX_ER	ALT5
	IO8	ENET2_RX_DATA0	ALT5
	IO9	ENET2_RX_DATA1	ALT5
	IO10	ENET2_RX_EN	ALT5
	IO11	ENET2_TX_DATA0	ALT5
	IO12	ENET2_TX_DATA1	ALT5
	IO13	ENET2_TX_EN	ALT5
	IO14	ENET2_TX_CLK	ALT5
	IO15	ENET2_RX_ER	ALT5
	IO16	SD1_CMD	ALT5
	IO17	SD1_CLK	ALT5
	IO18	SD1_DATA0	ALT5
	IO19	SD1_DATA1	ALT5

Table continues on the next page...

Instance	Port	Pad	Mode
	IO20	SD1_DATA2	ALT5
	IO21	SD1_DATA3	ALT5
GPIO3	IO0	LCD_CLK	ALT5
	IO1	LCD_ENABLE	ALT5
	IO2	LCD_HSYNC	ALT5
	IO3	LCD_VSYNC	ALT5
	IO4	LCD_RESET	ALT5
	IO5	LCD_DATA00	ALT5
	IO6	LCD_DATA01	ALT5
	IO7	LCD_DATA02	ALT5
	IO8	LCD_DATA03	ALT5
	IO9	LCD_DATA04	ALT5
	IO10	LCD_DATA05	ALT5
	IO11	LCD_DATA06	ALT5
	IO12	LCD_DATA07	ALT5
	IO13	LCD_DATA08	ALT5
	IO14	LCD_DATA09	ALT5
	IO15	LCD_DATA10	ALT5
	IO16	LCD_DATA11	ALT5
	IO17	LCD_DATA12	ALT5
	IO18	LCD_DATA13	ALT5
	IO19	LCD_DATA14	ALT5
	IO20	LCD_DATA15	ALT5
	IO21	LCD_DATA16	ALT5
	IO22	LCD_DATA17	ALT5
	IO23	LCD_DATA18	ALT5
	IO24	LCD_DATA19	ALT5
	IO25	LCD_DATA20	ALT5
	IO26	LCD_DATA21	ALT5
	IO27	LCD_DATA22	ALT5
	IO28	LCD_DATA23	ALT5
GPIO4	IO0	NAND_RE_B	ALT5
	IO1	NAND_WE_B	ALT5
	IO2	NAND_DATA00	ALT5
	IO3	NAND_DATA01	ALT5
	IO4	NAND_DATA02	ALT5
	IO5	NAND_DATA03	ALT5
	IO6	NAND_DATA04	ALT5
	IO7	NAND_DATA05	ALT5
	IO8	NAND_DATA06	ALT5
	IO9	NAND_DATA07	ALT5

Table continues on the next page...

Instance	Port	Pad	Mode
	IO10	NAND_ALE	ALT5
	IO11	NAND_WP_B	ALT5
	IO12	NAND_READY_B	ALT5
	IO13	NAND_CE0_B	ALT5
	IO14	NAND_CE1_B	ALT5
	IO15	NAND_CLE	ALT5
	IO16	NAND_DQS	ALT5
	IO17	CSI_MCLK	ALT5
	IO18	CSI_PIXCLK	ALT5
	IO19	CSI_VSYNC	ALT5
	IO20	CSI_HSYNC	ALT5
	IO21	CSI_DATA00	ALT5
	IO22	CSI_DATA01	ALT5
	IO23	CSI_DATA02	ALT5
	IO24	CSI_DATA03	ALT5
	IO25	CSI_DATA04	ALT5
	IO26	CSI_DATA05	ALT5
	IO27	CSI_DATA06	ALT5
	IO28	CSI_DATA07	ALT5
GPIO5	IO0	SNVS_TAMPER0	No Muxing (ALT5)
	IO1	SNVS_TAMPER1	No Muxing (ALT5)
	IO2	SNVS_TAMPER2	No Muxing (ALT5)
	IO3	SNVS_TAMPER3	No Muxing (ALT5)
	IO4	SNVS_TAMPER4	No Muxing (ALT5)
	IO5	SNVS_TAMPER5	No Muxing (ALT5)
	IO6	SNVS_TAMPER6	No Muxing (ALT5)
	IO7	SNVS_TAMPER7	No Muxing (ALT5)
	IO8	SNVS_TAMPER8	No Muxing (ALT5)
	IO9	SNVS_TAMPER9	No Muxing (ALT5)
	IO10	BOOT_MODE0	No Muxing (ALT5)
	IO11	BOOT_MODE1	No Muxing (ALT5)
GPT1	CAPTURE1	GPIO1_IO00	ALT1
		UART2_TX_DATA	ALT4
	CAPTURE2	ENET1_RX_ER	ALT8
		UART2_RX_DATA	ALT4
	CLK	ENET1_TX_CLK	ALT8
		UART1_RX_DATA	ALT4
	COMPARE1	GPIO1_IO01	ALT1
		UART1_TX_DATA	ALT4
	COMPARE2	GPIO1_IO02	ALT1
		UART2_CTS_B	ALT4

Table continues on the next page...

Instance	Port	Pad	Mode
	COMPARE3	GPIO1_IO03	ALT1
		UART2_RTS_B	ALT4
GPT2	CAPTURE1	JTAG_TMS	ALT1
		SD1_DATA2	ALT1
	CAPTURE2	JTAG_TDO	ALT1
		SD1_DATA3	ALT1
	CLK	JTAG_MOD	ALT1
		SD1_DATA1	ALT1
	COMPARE1	JTAG_TDI	ALT1
		SD1_CMD	ALT1
	COMPARE2	JTAG_TCK	ALT1
		SD1_CLK	ALT1
	COMPARE3	JTAG_TRST_B	ALT1
		SD1_DATA0	ALT1
I2C1	SCL	CSI_PIXCLK	ALT3
		GPIO1_IO02	ALT0
		UART4_TX_DATA	ALT2
	SDA	CSI_MCLK	ALT3
		GPIO1_IO03	ALT0
		UART4_RX_DATA	ALT2
I2C2	SCL	CSI_HSYNC	ALT3
		GPIO1_IO00	ALT0
		UART5_TX_DATA	ALT2
	SDA	CSI_VSYNC	ALT3
		GPIO1_IO01	ALT0
		UART5_RX_DATA	ALT2
I2C3	SCL	ENET2_RX_DATA0	ALT3
		LCD_DATA01	ALT4
		UART1_TX_DATA	ALT2
	SDA	ENET2_RX_DATA1	ALT3
		LCD_DATA00	ALT4
		UART1_RX_DATA	ALT2
I2C4	SCL	ENET2_RX_EN	ALT3
		LCD_DATA03	ALT4
		UART2_TX_DATA	ALT2
	SDA	ENET2_TX_DATA0	ALT3
		LCD_DATA02	ALT4
		UART2_RX_DATA	ALT2
KPP	COL0	ENET1_RX_DATA1	ALT6
		NAND_WE_B	ALT3
	COL1	ENET1_TX_DATA0	ALT6

Table continues on the next page...

Instance	Port	Pad	Mode
		NAND_DATA01	ALT3
	COL2	ENET1_TX_EN	ALT6
		NAND_DATA03	ALT3
	COL3	ENET1_RX_ER	ALT6
	COL4	ENET2_RX_DATA1	ALT6
	COL5	ENET2_TX_DATA0	ALT6
	COL6	ENET2_TX_EN	ALT6
	COL7	ENET2_RX_ER	ALT6
	ROW0	ENET1_RX_DATA0	ALT6
		NAND_RE_B	ALT3
	ROW1	ENET1_RX_EN	ALT6
		NAND_DATA00	ALT3
	ROW2	ENET1_TX_DATA1	ALT6
		NAND_DATA02	ALT3
	ROW3	ENET1_TX_CLK	ALT6
	ROW4	ENET2_RX_DATA0	ALT6
	ROW5	ENET2_RX_EN	ALT6
	ROW6	ENET2_TX_DATA1	ALT6
	ROW7	ENET2_TX_CLK	ALT6
LCDIF	BUSY	LCD_VSYNC	ALT1
	CLK	LCD_CLK	ALT0
	CS	LCD_RESET	ALT1
	DATA0	LCD_DATA00	ALT0
	DATA1	LCD_DATA01	ALT0
	DATA2	LCD_DATA02	ALT0
	DATA3	LCD_DATA03	ALT0
	DATA4	LCD_DATA04	ALT0
	DATA5	LCD_DATA05	ALT0
	DATA6	LCD_DATA06	ALT0
	DATA7	LCD_DATA07	ALT0
	DATA8	LCD_DATA08	ALT0
	DATA9	LCD_DATA09	ALT0
	DATA10	LCD_DATA10	ALT0
	DATA11	LCD_DATA11	ALT0
	DATA12	LCD_DATA12	ALT0
	DATA13	LCD_DATA13	ALT0
	DATA14	LCD_DATA14	ALT0
	DATA15	LCD_DATA15	ALT0
	DATA16	LCD_DATA16	ALT0
	DATA17	LCD_DATA17	ALT0
	DATA18	LCD_DATA18	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
	DATA19	LCD_DATA19	ALT0
	DATA20	LCD_DATA20	ALT0
	DATA21	LCD_DATA21	ALT0
	DATA22	LCD_DATA22	ALT0
	DATA23	LCD_DATA23	ALT0
	ENABLE	LCD_ENABLE	ALT0
	HSYNC	LCD_HSYNC	ALT0
	RD_E	LCD_ENABLE	ALT1
	RESET	LCD_RESET	ALT0
	RS	LCD_HSYNC	ALT1
	VSYNC	LCD_VSYNC	ALT0
	WR_RWN	LCD_CLK	ALT1
MMDC	ADDR0	DRAM_ADDR00	No Muxing (ALT0)
	ADDR1	DRAM_ADDR01	No Muxing (ALT0)
	ADDR2	DRAM_ADDR02	No Muxing (ALT0)
	ADDR3	DRAM_ADDR03	No Muxing (ALT0)
	ADDR4	DRAM_ADDR04	No Muxing (ALT0)
	ADDR5	DRAM_ADDR05	No Muxing (ALT0)
	ADDR6	DRAM_ADDR06	No Muxing (ALT0)
	ADDR7	DRAM_ADDR07	No Muxing (ALT0)
	ADDR8	DRAM_ADDR08	No Muxing (ALT0)
	ADDR9	DRAM_ADDR09	No Muxing (ALT0)
	ADDR10	DRAM_ADDR10	No Muxing (ALT0)
	ADDR11	DRAM_ADDR11	No Muxing (ALT0)
	ADDR12	DRAM_ADDR12	No Muxing (ALT0)
	ADDR13	DRAM_ADDR13	No Muxing (ALT0)
	ADDR14	DRAM_ADDR14	No Muxing (ALT0)
	ADDR15	DRAM_ADDR15	No Muxing (ALT0)
	CAS_B	DRAM_CAS_B	No Muxing (ALT0)
	CS0_B	DRAM_CS0_B	No Muxing (ALT0)
	CS1_B	DRAM_CS1_B	No Muxing (ALT0)
	DATA0	DRAM_DATA00	No Muxing (ALT0)
	DATA1	DRAM_DATA01	No Muxing (ALT0)
	DATA2	DRAM_DATA02	No Muxing (ALT0)
	DATA3	DRAM_DATA03	No Muxing (ALT0)
	DATA4	DRAM_DATA04	No Muxing (ALT0)
	DATA5	DRAM_DATA05	No Muxing (ALT0)
	DATA6	DRAM_DATA06	No Muxing (ALT0)
	DATA7	DRAM_DATA07	No Muxing (ALT0)
	DATA8	DRAM_DATA08	No Muxing (ALT0)
	DATA9	DRAM_DATA09	No Muxing (ALT0)

Table continues on the next page...

Instance	Port	Pad	Mode
	DATA10	DRAM_DATA10	No Muxing (ALT0)
	DATA11	DRAM_DATA11	No Muxing (ALT0)
	DATA12	DRAM_DATA12	No Muxing (ALT0)
	DATA13	DRAM_DATA13	No Muxing (ALT0)
	DATA14	DRAM_DATA14	No Muxing (ALT0)
	DATA15	DRAM_DATA15	No Muxing (ALT0)
	DQM0	DRAM_DQM0	No Muxing (ALT0)
	DQM1	DRAM_DQM1	No Muxing (ALT0)
	ODT0	DRAM_ODT0	No Muxing (ALT0)
	ODT1	DRAM_ODT1	No Muxing (ALT0)
	RAS_B	DRAM_RAS_B	No Muxing (ALT0)
	RESET	DRAM_RESET	No Muxing (ALT0)
	SDBA0	DRAM_SDBA0	No Muxing (ALT0)
	SDBA1	DRAM_SDBA1	No Muxing (ALT0)
	SDBA2	DRAM_SDBA2	No Muxing (ALT0)
	SDCKE0	DRAM_SDCKE0	No Muxing (ALT0)
	SDCKE1	DRAM_SDCKE1	No Muxing (ALT0)
	SDCLK0_P	DRAM_SDCLK0_P	No Muxing (ALT0)
	SDCLK0_N	DRAM_SDCLK0_N	No Muxing (ALT0)
	SDQS0_P	DRAM_SDQS0_P	No Muxing (ALT0)
	SDQS0_N	DRAM_SDQS0_N	No Muxing (ALT0)
	SDQS1_P	DRAM_SDQS1_P	No Muxing (ALT0)
	SDQS1_N	DRAM_SDQS1_N	No Muxing (ALT0)
	SDWE_B	DRAM_SDWE_B	No Muxing (ALT0)
MQS	LEFT	GPIO1_IO01	ALT4
		JTAG_TDI	ALT6
		LCD_DATA23	ALT1
	RIGHT	GPIO1_IO00	ALT4
		JTAG_TDO	ALT6
		LCD_DATA22	ALT1
NAND	ALE	NAND_ALE	ALT0
	CE0_B	NAND_CE0_B	ALT0
	CE1_B	NAND_CE1_B	ALT0
	CE2_B	CSI_MCLK	ALT2
	CE3_B	CSI_PIXCLK	ALT2
	CLE	NAND_CLE	ALT0
	DATA00	NAND_DATA00	ALT0
	DATA01	NAND_DATA01	ALT0
	DATA02	NAND_DATA02	ALT0
	DATA03	NAND_DATA03	ALT0
	DATA04	NAND_DATA04	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
	DATA05	NAND_DATA05	ALT0
	DATA06	NAND_DATA06	ALT0
	DATA07	NAND_DATA07	ALT0
	DQS	NAND_DQS	ALT0
	READY_B	NAND_READY_B	ALT0
	RE_B	NAND_RE_B	ALT0
	WE_B	NAND_WE_B	ALT0
	WP_B	NAND_WP_B	ALT0
PWM1	OUT	ENET1_RX_DATA0	ALT2
		GPIO1_IO08	ALT0
		LCD_DATA00	ALT1
PWM2	OUT	ENET1_RX_DATA1	ALT2
		GPIO1_IO09	ALT0
		LCD_DATA01	ALT1
PWM3	OUT	GPIO1_IO04	ALT1
		LCD_DATA02	ALT1
		NAND_ALE	ALT3
PWM4	OUT	GPIO1_IO05	ALT1
		LCD_DATA03	ALT1
		NAND_WP_B	ALT3
PWM5	OUT	ENET1_TX_DATA1	ALT2
		LCD_DATA18	ALT1
		NAND_DQS	ALT3
PWM6	OUT	ENET1_TX_EN	ALT2
		JTAG_TDI	ALT4
		LCD_DATA19	ALT1
PWM7	OUT	CSI_VSYNC	ALT6
		ENET1_TX_CLK	ALT2
		JTAG_TCK	ALT4
PWM8	OUT	CSI_HSYNC	ALT6
		ENET1_RX_ER	ALT2
		JTAG_TRST_B	ALT4
QSPI	A_DATA0	NAND_READY_B	ALT2
	A_DATA1	NAND_CE0_B	ALT2
	A_DATA2	NAND_CE1_B	ALT2
	A_DATA3	NAND_CLE	ALT2
	A_DQS	NAND_ALE	ALT2
	A_SCLK	NAND_WP_B	ALT2
	A_SS0_B	NAND_DQS	ALT2
	A_SS1_B	NAND_DATA07	ALT2
	B_DATA0	NAND_DATA02	ALT2

Table continues on the next page...

Instance	Port	Pad	Mode
	B_DATA1	NAND_DATA03	ALT2
	B_DATA2	NAND_DATA04	ALT2
	B_DATA3	NAND_DATA05	ALT2
	B_DQS	NAND_DATA01	ALT2
	B_SCLK	NAND_RE_B	ALT2
	B_SS0_B	NAND_WE_B	ALT2
	B_SS1_B	NAND_DATA00	ALT2
SAI1	MCLK	CSI_DATA01	ALT6
		LCD_DATA00	ALT8
	RX_BCLK	CSI_DATA03	ALT6
	RX_DATA	CSI_DATA06	ALT6
		LCD_DATA03	ALT8
	RX_SYNC	CSI_DATA02	ALT6
	TX_BCLK	CSI_DATA05	ALT6
		LCD_DATA02	ALT8
	TX_DATA	CSI_DATA07	ALT6
		LCD_DATA04	ALT8
	TX_SYNC	CSI_DATA04	ALT6
		LCD_DATA01	ALT8
SAI2	MCLK	JTAG_TMS	ALT2
		SD1_CLK	ALT2
	RX_BCLK	NAND_DATA06	ALT2
	RX_DATA	JTAG_TCK	ALT2
		SD1_DATA2	ALT2
	RX_SYNC	SD1_CMD	ALT2
	TX_BCLK	JTAG_TDI	ALT2
		SD1_DATA1	ALT2
	TX_DATA	JTAG_TRST_B	ALT2
		SD1_DATA3	ALT2
	TX_SYNC	JTAG_TDO	ALT2
		SD1_DATA0	ALT2
SAI3	MCLK	LCD_CLK	ALT3
		LCD_DATA09	ALT1
	RX_BCLK	LCD_DATA11	ALT1
	RX_DATA	LCD_DATA14	ALT1
	RX_SYNC	LCD_VSYNC	ALT3
		LCD_DATA10	ALT1
	TX_BCLK	LCD_DATA13	ALT1
		LCD_HSYNC	ALT3
	TX_DATA	LCD_DATA15	ALT1
		LCD_RESET	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
	TX_SYNC	LCD_DATA12	ALT1
		LCD_ENABLE	ALT3
SDMA	EXT_EVENT0	GPIO1_IO02	ALT6
		JTAG_MOD	ALT6
		SD1_CMD	ALT6
	EXT_EVENT1	JTAG_TMS	ALT6
		NAND_DQS	ALT6
SIM1	PORT0_CLK	ENET2_RX_DATA1	ALT2
	PORT0_PD	UART3_TX_DATA	ALT2
	PORT0_RST_B	ENET2_RX_EN	ALT2
	PORT0_SVEN	ENET2_TX_DATA0	ALT2
	PORT0_TRXD	ENET2_RX_DATA0	ALT2
	PORT1_CLK	CSI_VSYNC	ALT2
	PORT1_PD	CSI_HSYNC	ALT2
	PORT1_RST_B	CSI_DATA00	ALT2
	PORT1_SVEN	CSI_DATA01	ALT2
	PORT1_TRXD	CSI_DATA02	ALT2
	POWER_FAIL	JTAG_TDI	ALT8
SIM2	PORT0_CLK	ENET2_TX_EN	ALT2
	PORT0_PD	UART3_RX_DATA	ALT2
	PORT0_RST_B	ENET2_TX_CLK	ALT2
	PORT0_SVEN	ENET2_RX_ER	ALT2
	PORT0_TRXD	ENET2_TX_DATA1	ALT2
	PORT1_CLK	CSI_DATA04	ALT2
	PORT1_PD	CSI_DATA03	ALT2
	PORT1_RST_B	CSI_DATA05	ALT2
	PORT1_SVEN	CSI_DATA06	ALT2
	PORT1_TRXD	CSI_DATA07	ALT2
	POWER_FAIL	JTAG_TCK	ALT8
SJC	DE_B	UART2_CTS_B	ALT7
	MOD	JTAG_MOD	ALT0
	TCK	JTAG_TCK	ALT0
	TDI	JTAG_TDI	ALT0
	TDO	JTAG_TDO	ALT0
	TMS	JTAG_TMS	ALT0
	TRSTB	JTAG_TRST_B	ALT0
SNVS	VIO_5	CSI_PIXCLK	ALT6
	VIO_5_CTL	CSI_MCLK	ALT6
	PMIC_ON_REQ	SNVS_PMIC_ON_REQ	No Muxing (ALT0)
SRC	BT_CFG0	LCD_DATA00	ALT6
	BT_CFG1	LCD_DATA01	ALT6

Table continues on the next page...

Instance	Port	Pad	Mode
	BT_CFG2	LCD_DATA02	ALT6
	BT_CFG3	LCD_DATA03	ALT6
	BT_CFG4	LCD_DATA04	ALT6
	BT_CFG5	LCD_DATA05	ALT6
	BT_CFG6	LCD_DATA06	ALT6
	BT_CFG7	LCD_DATA07	ALT6
	BT_CFG8	LCD_DATA08	ALT6
	BT_CFG9	LCD_DATA09	ALT6
	BT_CFG10	LCD_DATA10	ALT6
	BT_CFG11	LCD_DATA11	ALT6
	BT_CFG12	LCD_DATA12	ALT6
	BT_CFG13	LCD_DATA13	ALT6
	BT_CFG14	LCD_DATA14	ALT6
	BT_CFG15	LCD_DATA15	ALT6
	BT_CFG24	LCD_DATA16	ALT6
	BT_CFG25	LCD_DATA17	ALT6
	BT_CFG26	LCD_DATA18	ALT6
	BT_CFG27	LCD_DATA19	ALT6
	BT_CFG28	LCD_DATA20	ALT6
	BT_CFG29	LCD_DATA21	ALT6
	BT_CFG30	LCD_DATA22	ALT6
	BT_CFG31	LCD_DATA23	ALT6
	POR_B	POR_B	No Muxing (ALT0)
	ONOFF	ONOFF	No Muxing (ALT0)
	BOOT_MODE0	BOOT_MODE0	No Muxing
	BOOT_MODE1	BOOT_MODE1	No Muxing
SPDIF	EXT_CLK	LCD_DATA07	ALT4
		NAND_DQS	ALT8
	IN	GPIO1_IO09	ALT2
		LCD_DATA08	ALT1
		SD1_CLK	ALT3
		UART1_RX_DATA	ALT8
	LOCK	LCD_DATA06	ALT4
	OUT	GPIO1_IO08	ALT2
		JTAG_MOD	ALT2
		LCD_DATA05	ALT4
		SD1_CMD	ALT3
		UART1_TX_DATA	ALT8
	SR_CLK	LCD_DATA04	ALT4
UART1	CTS_B	GPIO1_IO06	ALT8
		UART1_CTS_B	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
	RTS_B	GPIO1_IO07	ALT8
		UART1_RTS_B	ALT0
	RX_DATA	GPIO1_IO03	ALT8
		UART1_RX_DATA	ALT0
	TX_DATA	GPIO1_IO02	ALT8
		UART1_TX_DATA	ALT0
UART2	CTS_B	NAND_DATA06	ALT8
		UART2_CTS_B	ALT0
		UART3_TX_DATA	ALT4
	RTS_B	NAND_DATA07	ALT8
		UART2_RTS_B	ALT0
		UART3_RX_DATA	ALT4
	RX_DATA	NAND_DATA05	ALT8
		UART2_RX_DATA	ALT0
	TX_DATA	NAND_DATA04	ALT8
		UART2_TX_DATA	ALT0
UART3	CTS_B	NAND_CE1_B	ALT8
		UART3_CTS_B	ALT0
	RTS_B	NAND_CLE	ALT8
		UART3_RTS_B	ALT0
	RX_DATA	NAND_CE0_B	ALT8
		UART3_RX_DATA	ALT0
	TX_DATA	NAND_READY_B	ALT8
		UART3_TX_DATA	ALT0
UART4	CTS_B	ENET1_RX_DATA1	ALT1
		LCD_HSYNC	ALT2
	RTS_B	ENET1_RX_DATA0	ALT1
		LCD_VSYNC	ALT2
	RX_DATA	LCD_ENABLE	ALT2
		UART4_RX_DATA	ALT0
	TX_DATA	LCD_CLK	ALT2
		UART4_TX_DATA	ALT0
UART5	CTS_B	CSI_DATA03	ALT8
		ENET1_TX_DATA0	ALT1
		GPIO1_IO09	ALT8
	RTS_B	CSI_DATA02	ALT8
		ENET1_RX_EN	ALT1
		GPIO1_IO08	ALT8
	RX_DATA	CSI_DATA01	ALT8
		GPIO1_IO05	ALT8
		UART5_RX_DATA	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
	TX_DATA	CSI_DATA00	ALT8
		GPIO1_IO04	ALT8
		UART5_TX_DATA	ALT0
UART6	CTS_B	CSI_HSYNC	ALT8
		ENET1_TX_DATA1	ALT1
	RTS_B	CSI_VSYNC	ALT8
		ENET1_TX_EN	ALT1
	RX_DATA	CSI_PIXCLK	ALT8
		ENET2_RX_DATA1	ALT1
	TX_DATA	CSI_MCLK	ALT8
		ENET2_RX_DATA0	ALT1
UART7	CTS_B	ENET1_TX_CLK	ALT1
		LCD_DATA06	ALT1
	RTS_B	ENET1_RX_ER	ALT1
		LCD_DATA07	ALT1
	RX_DATA	ENET2_TX_DATA0	ALT1
		LCD_DATA17	ALT1
	TX_DATA	ENET2_RX_EN	ALT1
		LCD_DATA16	ALT1
UART8	CTS_B	ENET2_TX_CLK	ALT1
		LCD_DATA04	ALT1
	RTS_B	ENET2_RX_ER	ALT1
		LCD_DATA05	ALT1
	RX_DATA	ENET2_TX_EN	ALT1
		LCD_DATA21	ALT1
	TX_DATA	ENET2_TX_DATA1	ALT1
		LCD_DATA20	ALT1
USB	OTG1_ID	GPIO1_IO00	ALT2
		SD1_DATA0	ALT8
		UART3_TX_DATA	ALT8
	OTG2_ID	ENET2_TX_CLK	ALT8
		GPIO1_IO05	ALT2
		SD1_DATA3	ALT8
	OTG1_OC	ENET2_RX_DATA1	ALT8
		GPIO1_IO01	ALT2
		SD1_CLK	ALT8
	OTG1_PWR	ENET2_RX_DATA0	ALT8
		GPIO1_IO04	ALT2
		SD1_CMD	ALT8
	OTG2_OC	ENET2_TX_EN	ALT8
		GPIO1_IO03	ALT2

Table continues on the next page...

Instance	Port	Pad	Mode
		SD1_DATA2	ALT8
	OTG2_PWR	ENET2_TX_DATA1	ALT8
		GPIO1_IO02	ALT2
		SD1_DATA1	ALT8
USDHC1	CD_B	CSI_DATA05	ALT8
		GPIO1_IO03	ALT4
		UART1_RTS_B	ALT2
	CLK	SD1_CLK	ALT0
	CMD	SD1_CMD	ALT0
	DATA0	SD1_DATA0	ALT0
	DATA1	SD1_DATA1	ALT0
	DATA2	SD1_DATA2	ALT0
	DATA3	SD1_DATA3	ALT0
	DATA4	NAND_READY_B	ALT1
	DATA5	NAND_CE0_B	ALT1
	DATA6	NAND_CE1_B	ALT1
	DATA7	NAND_CLE	ALT1
	LCTL	ENET1_RX_DATA0	ALT8
	RESET_B	CSI_DATA06	ALT8
		GPIO1_IO04	ALT4
		GPIO1_IO09	ALT6
		NAND_WP_B	ALT1
	VSELECT	CSI_DATA07	ALT8
		ENET1_RX_EN	ALT8
		GPIO1_IO05	ALT4
	WP	CSI_DATA04	ALT8
		GPIO1_IO02	ALT4
		UART1_CTS_B	ALT2
USDHC2	CD_B	CSI_MCLK	ALT1
		GPIO1_IO07	ALT4
		UART1_RTS_B	ALT8
	CLK	CSI_VSYNC	ALT1
		LCD_DATA19	ALT8
		NAND_RE_B	ALT1
	CMD	CSI_HSYNC	ALT1
		LCD_DATA18	ALT8
		NAND_WE_B	ALT1
	DATA0	CSI_DATA00	ALT1
		LCD_DATA20	ALT8
		NAND_DATA00	ALT1
	DATA1	CSI_DATA01	ALT1

Table continues on the next page...

Instance	Port	Pad	Mode
		LCD_DATA21	ALT8
		NAND_DATA01	ALT1
	DATA2	CSI_DATA02	ALT1
		LCD_DATA22	ALT8
		NAND_DATA02	ALT1
	DATA3	CSI_DATA03	ALT1
		LCD_DATA23	ALT8
		NAND_DATA03	ALT1
	DATA4	CSI_DATA04	ALT1
		LCD_DATA14	ALT8
		NAND_DATA04	ALT1
	DATA5	CSI_DATA05	ALT1
		LCD_DATA15	ALT8
		NAND_DATA05	ALT1
	DATA6	CSI_DATA06	ALT1
		LCD_DATA16	ALT8
		NAND_DATA06	ALT1
	DATA7	CSI_DATA07	ALT1
		LCD_DATA17	ALT8
		NAND_DATA07	ALT1
	LCTL	ENET1_RX_DATA1	ALT8
	RESET_B	GPIO1_IO09	ALT4
		LCD_DATA13	ALT8
		NAND_ALE	ALT1
	VSELECT	ENET1_TX_DATA0	ALT8
		GPIO1_IO08	ALT4
	WP	CSI_PIXCLK	ALT1
		GPIO1_IO06	ALT4
		UART1_CTS_B	ALT8
WDOG1	WDOG_ANY	ENET2_RX_ER	ALT8
		GPIO1_IO09	ALT1
		LCD_DATA19	ALT2
		LCD_RESET	ALT4
	WDOG_B	GPIO1_IO01	ALT8
		GPIO1_IO08	ALT1
		UART3_RTS_B	ALT8
	WDOG_RST_B_DEB	ENET1_TX_DATA1	ALT8
		LCD_CLK	ALT8
WDOG2	WDOG_B	LCD_VSYNC	ALT4
	WDOG_RST_B_DEB	ENET1_TX_EN	ALT8
WDOG3	WDOG_B	GPIO1_IO00	ALT8

Table continues on the next page...

Instance	Port	Pad	Mode
	WDOG_RST_B_DEB	LCD_HSYNC	ALT4
XTALOSC	REF_CLK1	ENET1_TX_CLK	ALT4
		GPIO1_IO00	ALT3
		GPIO1_IO04	ALT0
	REF_CLK2	ENET2_TX_CLK	ALT4
		GPIO1_IO01	ALT3
		GPIO1_IO05	ALT0
	ENET_REF_CLK_25M	ENET2_RX_EN	ALT8
		GPIO1_IO02	ALT3
		JTAG_MOD	ALT3
	XTALI	XTALI	no muxing
	XTALO	XTALO	no muxing

Chapter 5
Fusemap

5.1 Boot Fusemap

The following section details the various modes and selection of the required boot devices.

A separate map is given for each and every boot device. The device select is specified by BOOT_CFG1[6:4] fuses listed in Table 5-1.

Table 5-1. Boot Device Select

Boot Device	BOOT_CFG1[7]	BOOT_CFG1[6]	BOOT_CFG1[5]	BOOT_CFG1[4]	BOOT_CFG1[3]
QSPI	0	0	0	1	x
WEIM	0	0	0	0	Memory Type: 0 - NOR Flash 1 - OneNAND
Serial-ROM	0	0	1	1	x
SD/eSD	0	1	0	x	x
MMC/eMMC	0	1	1	x	x
NAND	1	x	x	x	x

NOTE

Fuses marked as “Reserved” are reserved for Freescale internal (and future) use only. Customers should not attempt to burn these, as the IC behavior may be unpredictable. The reserved fuses can be read as either '0' or '1'.

Table 5-2. QSPI Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	0	1	x	DDRSMP: 000 - Default		

Table continues on the next page...

Table 5-2. QSPI Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
0x450[15:8] (BOOT_CFG2)	Reserved	HSPHS: Half Speed Phase Selection 0 - select sampling at non-inverted clock 1 - select sampling at inverted clock	HSDLY: Half Speed Delay selection 0 - one clock delay 1 - two clock delay	FSPHS: Full Speed Phase Selection 0 - select sampling at non-inverted clock 1 - select sampling at inverted clock	FSDLY: Full Speed Delay selection 0 - one clock delay 1 - two clock delay	Boot Frequencies (ARM/DDR) 0 - 500 / 400 MHz 1 - 250 / 200 MHz	Reserved	Reserved
0x450[23:16] (BOOT_CFG3)	Reserved							
0x460[7:0]	Reserved		FORCE_COLD_BOOT	BT_FUSE_SEL	DIR_BT_DIS	Reserved	SEC_CONFIG[1]	Reserved
0x460[15:8]	Reserved							
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_ENABLE 0 - Disabled 1 - Enabled	SJC_DISABLE	Reserved			
0x460[31:24]	Reserved			TZASC_ENABLE	JTAG_HEO	KTE	Reserved	DLL_ENABLE 0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC
0x470[7:0]	Not used				Reserved	L1 I-Cache DISABLE	BT_MMU_DISABLE	Reserved
0x470[15:8]	Reserved	Not used	Reserved	ADD_DS__SET_GPR1_16 0 - Set 1 - Don't set	Reserved			
0x470[23:16]	Reserved	LPB_BOOT (Core / DDR-Bus) 00 - LPB Disable 01 - 1 GPIO (def freq) 10 - Div by2 11 - Div by 4		BT_LPB_POLARITY (GPIO polarity) 0 - Active High 1 -Active Low	Reserved			

Table 5-3. EIM Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	0	0	Memory Type: 0 - NOR Flash 1 - OneNAND	Reserved	Reserved	Reserved
0x450[15:8] (BOOT_CFG2)	Muxing Scheme: 00 - A/D16 (HW Default in external boot) 01 - A+DH 10 - A+DL 11 - Reserved		OneNand Page Size: 00 - 1KB 01 - 2KB 10 - 4KB 11 - Reserved		Reserved	Boot Frequencies (ARM/DDR) 0 - 500 / 400 MHz 1 - 250 / 200 MHz	Reserved	Reserved
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved	EEPROM Recovery Enable 0 - Disabled 1 - Enabled Fuse is Reserved for 'Serial-ROM' Boot mode	eCSPI chip select: 00 - ECSPIx_SS0 (default) 01 - ECSPIx_SS1 10 - ECSPIx_SS2 11 - ECSPIx_SS3		eCSPI Addressing: 0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)	Port Select: 000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4 100 - Reserved 101 - Reserved 110 - Reserved 111 - Reserved		
0x460[7:0]	Reserved		FORCE_COLD_BOOT	BT_FUSE_SEL	DIR_BT_DIS	Reserved	SEC_CONFIG[1]	Reserved
0x460[15:8]	Reserved							
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_ENABLE 0 - Disabled 1 - Enabled	SJC_DISABLE	Reserved			
0x460[31:24]	Reserved			TZASC_ENABLE	JTAG_HEO	KTE	Reserved	DLL_ENABLE 0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC
0x470[7:0]	DLL Override:	SD1_RST_POLARITY_SELECT	SD2 VOLTAGE SELECTION	Reserved	Disable SDMMC Manufacture mode	L1 I-Cache DISABLE	BT_MMU_DISABLE	Override SD Pad Settings

Table continues on the next page...

Table 5-3. EIM Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
	0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	0 - Reset active low 1 - Reset active high	0 - 3.3V 1 - 1.8V		0 - Enable 1 - Disable			(using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT 0 - Reset acitve low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP 0 - 47K pullup 1 - 22K pullup	ADD_DS__SET_GPR1_16 0 - Set 1 - Do not set	USDHC_IOMUX_SION_BIT_ENABLE 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT (Core / DDR-Bus) 00 - LPB Disable 01 - 1 GPIO (def freq) 10 - Div by2 11 - Div by 4		BT_LPB_POLARITY (GPIO polarity) 0 - Active High 1 -Active Low	Reserved			
0x470[31:24]	Override NAND Pad Settings (Use PAD_SETTINGS value)	MMC_DLL_DLY[6:0]						
0x6D0[7:0]	RANDOMIZER_ENABLE	Reserved	PAD_SETTINGS[5:0]					

Table 5-4. Serial-ROM Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	1	1	Reserved	Reserved	Reserved	Reserved
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	Boot Frequencies (ARM/DDR) 0 - 500 / 400 MHz 1 - 250 / 200 MHz	Reserved	Reserved
0x450[23:16]	Reserved							

Table continues on the next page...

Table 5-4. Serial-ROM Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
(BOOT_CFG3)								
0x450[31:24] (BOOT_CFG4)	Reserved	EEPROM Recovery Enable 0 - Disabled 1 - Enabled Fuse is Reserved for 'Serial-ROM' Boot mode	eCSPI chip select: 00 - ECSPIx_SS0 (default) 01 - ECSPIx_SS1 10 - ECSPIx_SS2 11 - ECSPIx_SS3		eCSPI Addressing: 0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)	Port Select: 000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4 100 - Reserved 101 - Reserved 110 - Reserved 111 - Reserved		
0x460[7:0]	Reserved		FORCE_COLD_BOOT	BT_FUSE_SEL	DIR_BT_DIS	Reserved	SEC_CONFIG[1]	Reserved
0x460[15:8]	Reserved							
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_ENABLE 0 - Disabled 1 - Enabled	SJC_DISABLE	Reserved			
0x460[31:24]	Reserved			TZASC_ENABLE	JTAG_HEO	KTE	Reserved	DLL_ENABLE 0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_POLARITY_SELECT 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	Reserved	Disable SDMMC Manufacture mode 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MMU_DISABLE	Override SD Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT 0 - Reset acitve low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP 0 - 47K pullup 1 - 22K pullup	ADD_DS__SET_GPR1_16 0 - Set 1 - Do not set	USDHC_IOMUX_SION_BIT_ENABLE 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT (Core / DDR-Bus)		BT_LPB_POLARITY	Reserved			

Table continues on the next page...

Table 5-4. Serial-ROM Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
		00 - LPB Disable 01 - 1 GPIO (def freq) 10 - Div by2 11 - Div by 4		(GPIO polarity) 0 - Active High 1 -Active Low				
0x470[31:24]	Override NAND Pad Settings (Use PAD_SETTINGS value)	MMC_DLL_DLY[6:0]						
0x6D0[7:0]	RANDOMIZER_ENABLE	Reserved	PAD_SETTINGS[5:0]					

Table 5-5. SD/eSD Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	1	0	Fast Boot: 0 - Regular 1 - Fast Boot	SD/SDXC Speed 00 - Normal/SDR12 01 - High/SDR25 10 - SDR50 11 - SDR104		SD Power Cycle Enable 0 - No power cycle 1 - Enabled via USDHC_RST pad	SD Loopback Clock Source Sel (for SDR50 and SDR104 only) 0 - through SD pad 1 - direct
0x450[15:8] (BOOT_CFG2)	SD Calibration Step 00 - 1 delay cell		Bus Width: 0 - 1-bit 1 - 4-bit	Port Select: 00 - eSDHC1 01 - eSDHC2 10 - Reserved 11 - Reserved		Boot Frequencies (ARM/DDR) 0 - 500 / 400 MHz 1 - 250 / 200 MHz	SD VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	Reserved
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved	EEPROM Recovery Enable 0 - Disabled 1 - Enabled Fuse is Reserved	eCSPI chip select: 00 - ECSPIx_SS0 (default) 01 - ECSPIx_SS1 10 - ECSPIx_SS2 11 - ECSPIx_SS3	eCSPI Addressing: 0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)	Port Select: 000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4			

Table continues on the next page...

Table 5-5. SD/eSD Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
		for 'Serial-ROM' Boot mode				100 - Reserved 101 - Reserved 110 - Reserved 111 - Reserved		
0x460[7:0]	Reserved		FORCE_COLD_BOOT	BT_FUSE_SEL	DIR_BT_DIS	Reserved	SEC_CONFIG[1]	Reserved
0x460[15:8]	Reserved							
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_ENABLE 0 - Disabled 1 - Enabled	SJC_DISABLE	Reserved			
0x460[31:24]	Reserved			TZASC_ENABLE	JTAG_HEO	KTE	Reserved	DLL Enable: 0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/eMMC 1 - DLL Override Mode for SD/eMMC	SD1 RST_POLARITY_SELECT 0 - Reset active-low 1 - Reset active-high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	Reserved	Disable SDMMC Manufacture mode 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MMU_DISABLE	Override SD Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2 Reset Signal Polarity: 0 - Reset active-low 1 - Reset active-high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP 0 - 47K pullup 1 - 22K pullup	ADD_DS__SET_GPR1_16 0 - Set 1 - Don't set	USDHC_IOMUX_SION_BIT_ENABLE 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT (Core / DDR-Bus) 00 - LPB Disable 01 - 1 GPIO (def freq) 10 - Div by2 11 - Div by 4		BT_LPB_POLARITY (GPIO polarity) 0 - Active High 1 -Active Low	Reserved			

Table continues on the next page...

Table 5-5. SD/eSD Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
0x470[31:24]	Override NAND Pad Settings (Use PAD_SETTINGS value)	MMC_DLL_DLY[6:0]						
0x6D0[7:0]	RANDOMIZER_ENABLE	Reserved	PAD_SETTINGS[5:0]					

Table 5-6. MMC/eMMC Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	1	1	Fast Boot: 0 - Regular 1 - Fast Boot	SD/MMC Speed 0 - High 1- Normal	Fast Boot Acknowledge Disable: 0 - Boot Ack Enabled 1 - Boot Ack Disabled	SD Power Cycle Enable 0 - No power cycle 1 - Enabled via USDHC_RST pad	SD Loopback Clock Source Sel (for SDR50 and SDR104 only) 0 - through SD pad 1 - direct
0x450[15:8] (BOOT_CFG2)	Bus Width: 000 - 1-bit 001 - 4-bit 010 - 8-bit 101 - 4-bit DDR (MMC 4.4) 110 - 8-bit DDR (MMC 4.4) Else - Reserved			Port Select: 00 - eSDHC1 01 - eSDHC2 10 - Reserved 11 - Reserved		Boot Frequencies (ARM/DDR) 0 - 500 / 400 MHz 1 - 250 / 200 MHz	SD VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	Reserved
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved	EEPROM Recovery Enable 0 - Disabled 1 - Enabled Fuse is Reserved	eCSPI chip select: 00 - ECSPIx_SS0 (default) 01 - ECSPIx_SS1 10 - ECSPIx_SS2 11 - ECSPIx_SS3	eCSPI Addressing: 0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)	Port Select: 000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4 100 - Reserved 101 - Reserved			

Table continues on the next page...

Table 5-6. MMC/eMMC Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
		for 'Serial-ROM' Boot mode				110 - Reserved 111 - Reserved		
0x460[7:0]	Reserved		FORCE_COLD_BOOT	BT_FUSE_SEL	DIR_BT_DIS	Reserved	SEC_CONFIG[1]	Reserved
0x460[15:8]	Reserved							
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_ENABLE 0 - Disabled 1 - Enabled	SJC_DISABLE	Reserved			
0x460[31:24]	Reserved			TZASC_ENABLE	JTAG_HEO	KTE	Reserved	DLL_ENABLE 0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_POLARITY_SELECT 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	Reserved	Disable SDMMC Manufacture mode 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MMU_DISABLE	Override SD Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2 Reset Signal Polarity: 0 - Reset active-low 1 - Reset active-high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP 0 - 47K pullup 1 - 22K pullup	ADD_DS__SET_GPR1_16 0 - Set 1 - Don't set	USDHC_IOMUX_SION_BIT_ENABLE 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable 0 - Disable 1 - Enable
0x470[23:16]	USDHC_CMD_OE_PRE_EN (SD/eMMC Debug) 0 - COM OE Disabled 1 - COM OE Enabled	LPB_BOOT (Core / DDR-Bus) 00 - LPB Disable 01 - 1 GPIO (def freq) 10 - Div by2 11 - Div by 4		BT_LPB_POLARITY (GPIO polarity) 0 - Active High 1 - Active Low	Reserved			
0x470[31:24]	Override NAND Pad Settings	MMC_DLL_DLY[6:0]						

Table continues on the next page...

Table 5-6. MMC/eMMC Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
	(Use PAD_SETTINGS value)							
0x6D0[7:0]	RANDOMIZER_ENABLE	Reserved	PAD_SETTINGS[5:0]					

Table 5-7. NAND Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	1	BT_TOGGLEMODE	Pages in Block: 00 - 128 01 - 64 10 - 32 11 - 256		Nand Number Of Devices: 00 - 1 01 - 2 10 - 4 11 - Reserved		Nand_Row_address_bytes: 00 - 3 01 - 2 10 - 4 11 - 5	
0x450[15:8] (BOOT_CFG2)	Toggle Mode 33 MHz Preamble Delay, Read Latency: 000 - 16 GPMICLK cycles. 001 - 1 GPMICLK cycles. 010 - 2 GPMICLK cycles. 011 - 3 GPMICLK cycles. 100 - 4 GPMICLK cycles. 101 - 5 GPMICLK cycles. 110 - 6 GPMICLK cycles. 111 - 7 GPMICLK cycles.		Boot Search Count: 00 - 2 01 - 2 10 - 4 11 - 8		Boot Frequencies (ARM/DDR) 0 - 500 / 400 MHz 1 - 250 / 200 MHz		Reset Time 0 - 12 ms 1 - 22 ms (LBA NAND)	Reserved
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved	EEPROM Recovery Enable 0 - Disabled 1 - Enabled Fuse is Reserved for 'Serial-ROM' Boot mode	eCSPI chip select: 00 - ECSPIx_SS0 (default) 01 - ECSPIx_SS1 10 - ECSPIx_SS2 11 - ECSPIx_SS3		eCSPI Addressing: 0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)	Port Select: 000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4 100 - Reserved 101 - Reserved 110 - Reserved 111 - Reserved		
0x460[7:0]	Reserved		FORCE_COLD_BOOT	BT_FUSE_SEL	DIR_BT_DIS	Reserved	SEC_CONFIG[1]	Reserved
0x460[15:8]	Reserved							

Table continues on the next page...

Table 5-7. NAND Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_ENABLE 0 - Disabled 1 - Enabled	SJC_DISABLE	Reserved			
0x460[31:24]	Reserved			TZASC_ENABLE	JTAG_HEO	KTE	Reserved	DLL_ENABLE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ eMMC
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_POLARITY_SELECT 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	Reserved	Disable SDMMC Manufacture mode 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MMU_DISABLE	Override SD Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT 0 - Reset acitve low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP 0 - 47K pullup 1 - 22K pullup	ADD_DS__SET_GPR1_16 0 - Set 1 - Do not set	USDHC_IOMUX_SION_BIT_ENABLE 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT (Core / DDR-Bus) 00 - LPB Disable 01 - 1 GPIO (def freq) 10 - Div by2 11 - Div by 4		BT_LPB_POLARITY (GPIO polarity) 0 - Active High 1 -Active Low	Reserved			
0x470[31:24]	Override NAND Pad Settings (Use PAD_SETTINGS value)	MMC_DLL_DLY[6:0]						
0x6D0[7:0]	RANDOMIZER_ENABLE	Reserved	PAD_SETTINGS[5:0]					

Table continues on the next page...

Table 5-7. NAND Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
0x6D0[23:16]	NAND_READ_CMD_CODE1[7:0]							
0x6D0[31:24]	NAND_READ_CMD_CODE2[7:0]							

5.2 Lock Fusemap

Table 5-8 describes the functions of a set of lock fuses.

Table 5-8. Lock Fuses

Addr	7	6	5	4	3	2	1	0
0x400[7:0]	Reserved	SJC_RESP_LOCK WRP,OP,RDP	MEM_TRIM_LOCK 1x - OP x1 - WP		BOOT_CFG_LOCK 1x - OP x1 - WP		TESTER_LOCK 1x - OP x1 - WP	
00x400[15:8]	GP3_LOCK 1 - WP + OP	SRK_LOCK	GP2_LOCK 1x - OP x1 - WP		GP1_LOCK 1x - OP x1 - WP		MAC_ADDR_LOCK 1x - OP x1 - WP	
0x400[23:16]	GP4_LOCK 1 - WP + OP	MISC_CONF_LOCK 1 - WP + OP	ROM_PATCH_LOCK 1 - WP + OP	OTPMK_CRC_LOCK 1 - WP + RP	ANALOG_LOCK 1x - OP x1 - WP		OTPMK_LOCK 1 - RP, WP, OP	SW_GP_LOCK 1 - WP + OP of SW_GP fuses
0x400[31:24]	GP6_LOCK 1x - OP x1 - WP		GP8_LOCK 1x - WP + Read Protect (CRC locking) x1 - WP + OP (General purpose use)		GP7_LOCK 1x - WP + Read Protect (CRC locking) x1 - WP + OP (General purpose use)		Reserved	GP5_LOCK 1 - WP + OP

5.3 Fusemap Descriptions Table

Table 5-9. Fusemap Descriptions

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x400[1:0]	TESTER_LOCK	2			OCOTP
0x400[3:2]	BOOT_CFG_LOCK	2	Perform lock on BOOT related fuses.	0 - Unlock (The controlled field can be read, sensed,	OCOTP

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
				burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can be read or sensed only)	
0x400[5:4]	MEM_TRIM_LOCK	2	Trimming fuses. Burnt on the tester or by customer before the final product shipment.	0 - Unlock (The controlled field can be read, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can be read only)	OCOTP
0x400[6]	SJC_RESP_LOCK	1			OCOTP
0x400[9:8]	MAC_ADDR_LOCK	2	Lock MAC_ADDR fuses.		OCOTP
0x400[11:10]	GP1_LOCK	2	Lock for General Purpose fuse register #1 (GP1)		OCOTP
0x400[13:12]	GP2_LOCK	2	Lock for General Purpose fuse register #2 (GP2)		OCOTP
0x400[14]	SRK_LOCK	1	Locking SRK_HASH[255:0]		OCOTP
0x400[15]	GP3_LOCK	1	Lock for General Purpose fuse register #3 (GP3)		OCOTP
0x400[19:18]	ANALOG_LOCK	2			OCOTP
0x400[22]	MISC_CONF_LOCK	1			OCOTP
0x400[23]	GP4_LOCK	1	Lock for General Purpose fuse register #4 (GP4)		OCOTP
0x400[24]	GP5_LOCK	1	Lock for General Purpose fuse register #5 (GP5)		OCOTP
0x400[27:26]	GP7_LOCK	2	Lock for GP7[127:0] '1x' – WP + Read Protect (CRC locking) 'x1' – WP + OP (General Purpose use)		OCOTP
0x400[29:28]	GP8_LOCK	2	Lock for GP8[127:0] '1x' – WP + Read Protect (CRC locking) 'x1' – WP + OP (General Purpose use)		OCOTP
0x400[31:30]	GP6_LOCK	2	Lock for General Purpose fuse register #1 (GP6) '1x' – OP 'x1' – WP		OCOTP
0x410[10:0]	LOT_NO_ENC[42:0] (SJC CHALL/ UNIQUE_ID[42:0])	43	FSL-wide unique, encoded LOT ID STD II/SJC CHALLENGE/ Unique ID		SJC, SW
0x420[15:11]	WAFER_NO[4:0] (SJC_CHALL[47:4	5	The wafer number of the wafer on which the device		SJC, SW

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

<table>
<tr><th>Fuse Address</th><th>Fuses Name</th><th>Number of Fuses</th><th>Fuses Function</th><th>Setting</th><th>Used by</th></tr>
<tr><td></td><td>3] / UNIQUE_ID[47:43])</td><td></td><td>was fabricated/SJC CHALLENGE/ Unique ID</td><td></td><td></td></tr>
<tr><td>0x420[23:16]</td><td>DIE-Y-CORDINATE[7:0] (SJC_CHALL[55:48] / UNIQUE_ID[55:48])</td><td>8</td><td>The Y-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID</td><td></td><td>SJC, SW</td></tr>
<tr><td>0x420[31:24]</td><td>DIE-X-CORDINATE[7:0] (SJC_CHALL[63:56] / UNIQUE_ID[63:56])</td><td>8</td><td>The X-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID</td><td></td><td>SJC, SW</td></tr>
<tr><td>0x430[19:16]</td><td>SI_REV[3:0]</td><td>4</td><td>Silicon Revision number. Can be used in conjunction with HW value set in GPR registers (gpr4[15:8]).</td><td></td><td>SW</td></tr>
<tr><td>0x430[21:20]</td><td>TAMPER_PIN_DISABLE[1:0]</td><td>2</td><td>Disable ten tamper pins</td><td>00 - enabled, TAMPER0-9 used as TAMPER detection pins.
01 - disabled, TAMPER2-4 and TAMPER7-9 used as GPIO.
10 - disabled, TAMPER0-1 and TAMPER5-6 used as GPIO.
11 - disabled, TAMPER0-9 used as GPIO.</td><td>SNVS</td></tr>
<tr><td>0x440[17:16]</td><td>SPEED_GRADING[1:0]</td><td>2</td><td>Burned by tester program, for indicating IC core speed (Hot burn may not be used).</td><td><table><tr><th>FHA[3:2]</th><th>FCA[5:4]</th><th>FRAL[0:1]</th><th>MHz</th><th>P/N Code</th></tr><tr><td>xx</td><td>xx</td><td>00</td><td>Reserved</td><td>Reserved</td></tr><tr><td>xx</td><td>xx</td><td>01</td><td>528</td><td>05</td></tr><tr><td>xx</td><td>xx</td><td>10</td><td>700</td><td>07</td></tr><tr><td>xx</td><td>xx</td><td>11</td><td>Reserved</td><td>Reserved</td></tr></table></td><td>PROD / SW</td></tr>
<tr><td>0x450[7:0]</td><td>BOOT_CFG1</td><td>8</td><td>BOOT configuration register #1, Usage varies, depending on selected boot device.</td><td>0x0000XXXX - WEIM (NOR/ OneNAND) boot
0x0011XXXX - Serial ROM (I2C/SPI) boot
0x1XXXXXXX - NAND</td><td>SRC</td></tr>
</table>

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
				FLASH boot 0x010XXXXX - SD/eSD 0x011XXXXX - MMC/eMMC boot Others - Reserved Refer to Fuse Map for details.	
0x450[15:8]	BOOT_CFG2	8	BOOT configuration register #2, Usage varies, depending on selected boot device.	See fuse-map tab for details.	SRC
0x450[23:16]	BOOT_CFG3	8	BOOT configuration register #3	See fuse-map tab for details.	SRC
0x450[31:24]	BOOT_CFG4	8	BOOT configuration register #3	See fuse-map tab for details.	SRC
0x460[1]	SEC_CONFIG[1]	1	Security Configuration (with SEC_CONFIG[0])	00 - FAB (Open) 01 - Open - allows any code to be flashed and executed, even if it has no valid signature. 1x - Closed (Security On)	SW (ROM), SRC, SNVS, TPSMP
0x460[3]	DIR_BT_DIS	1	Direct External Memory Boot Disable	0 - Direct boot from external memory is allowed 1 - Direct boot from external memory is not allowed	SW(ROM), SRC
0x460[4]	BT_FUSE_SEL	1	Determines, whether using fuses for boot configuration, or GPIO /Serial loader.	If boot_mode="00" (Development) 0=Boot mode configuration is taken from GPIOs. 1=Boot mode configuration is taken from fuses. If boot_mode="10" (Production) 0 - Boot using Serila Loader (USB) 1- Boot mode configuration is taken from fuses.	SRC SW(ROM)
0x460[5]	FORCE_COLD_BOOT(SBMR)	1	Force cold boot when A9 core come out of reset. Reflected in SBMR reg of SRC	Fuse Function: 0 – Default behavior equivalent to the rest of the i.MX6 family allowing a fast recovery from low power modes. That is, the ROM is allowed to jump to the address previously programmed in the SRC persistent register. 1 – Fast recovery path in the ROM is not allowed and a cold boot is always performed. Customers wanting a higher level of security should burn this fuse.	SRC
0x460[15:8]	DDR3_CONFIG[7:0]	8	DDR3 config options		SW(ROM)

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x460[16]	FORCE_INTERNAL_BOOT	1	Once been blown, the external BT_MODE[1:0] pins will be ignored, BootROM takes Boot Mode as "Internal Boot"	0—Boot Mode from BT_MODE pins 1—BT_MODE pins be ignored, Boot Mode forced to "Internal Boot"	SW(ROM)
0x460[17]	SDP_DISABLE	1	Disable/Enable serial download support	0—Serial download supported 1—No serial download support	SW(ROM)
0x460[18]	SDP_READ_DISABLE	1	Disable/Enable serial download READ_REGISTER command	0—SDP READ_REGISTER command is enabled 1—SDP READ_REGISTER is disabled	SW(ROM)
0x460[20]	SJC_DISABLE	1	Disable/Enable the Secure JTAG Controller module. This fuse is used to create highest JTAG security level, where JTAG is totally blocked.	0 - Secure JTAG Controller is enabled 1 - Secure JTAG Controller is disabled	Security logic
0x460[21]	WDOG_ENABLE	1	Watchdog Enable	Used to specify whether to enable / not watchdog at boot. '0' - Watch-Dog is disabled. '1' - Watch-Dog is enabled.	SW(ROM)
0x460[23:22]	JTAG_SMODE[1:0]	2	JTAG Security Mode. Controls the security mode of the JTAG debug interface	00 - JTAG enable mode 01 - Secure JTAG mode 11 - No debug mode	Security logic
0x460[24]	DLL_ENABLE	1	Controls the enable/disable of the DLL for SD/eMMC boot.	0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC	SW (ROM)
0x460[26]	KTE	1	Kill Trace Enable. Enables tracing capability on ETM, and other modules.	0 - Bus tracing is allowed 1 - Bus tracing is allowed in case security state as defined by Secure JTAG allows it (for example, JTAG_ENABLE or NO_DEBUG)	SJC
0x460[27]	JTAG_HEO	1	JTAG HAB Enable Override. Disallows HAB JTAG enabling. The HAB may normally enable JTAG debugging by means of the HAB_JDE-bit in the OCOTP SCS register. The JTAG_HEO-bit can override this behavior.	0 - HAB may enable JTAG debug access 1 - HAB JTAG enable is overridden (HAB may not enable JTAG debug access)	SJC

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x460[28]	TZASC_ENABLE	1	TZASC enable fuse.	0 - TZASC module left in disable and bypass state. 1 - TZASC modules and associated clocks and muxing are enabled by Boot ROM code.	SW (ROM)
0x460[29]	PWR_STABLE_CYCLE_SELECTION	1	Select power stable cycle	0 - 5 ms 1 - 2.5 ms	SW (ROM)
0x460[31:30]	SD_PWR_CYCLE_SELECTION	2	Select SD Power Cycle	00 - 20 ms 01 - 10 ms 10 - 5 ms 11 - 2.5 ms	SW (ROM)
0x470[0]	Override SD Pad Settings	1	Overrides ROM default value for SD PAD control register. When set ROM will override SD pad control register with value programmed into PAD_SETTINGS fuse bits.		SW (ROM)
0x470[1]	BT_MMU_DISABLE	1	The fuse bit is used for ROM to not enable MMU		SW (ROM)
0x470[2]	L1 I-Cache DISABLE	1	The fuse bit is used for ROM to not enable L1 I-Cache		SW (ROM)
0x470[3]	Disable SDMMC Manufacture mode	1	The fuse bit is used to disable ROM feature “SD/MMC Manufacturing Mode”.	0 - Enable 1 – Disable	SW (ROM)
0x470[4]	UART Serial Download Disable	1	Disable UART Serial Download	0 - Enable 1 – Disable	SW (ROM)
0x470[5]	SD2 VOLTAGE SELECTION	1	Fuse bit to change voltage selection for SD2 pads. When set ROM will select 1.8V for SD3 pads otherwise 3.3V	0 - 3.3V 1 - 1.8V	SW (ROM)
0x470[6]	SD1_RST_POLARITY_SELECT	1	Select reset polarity for SD1	0 - Reset active low 1 - Reset active high	SW (ROM)
0x470[7]	DLL Override	1	Select the DLL mode for SD/eMMC.	0 - DLL Slave Mode for SD/eMMC 1 - DLL Override Mode for SD/eMMC	SW (ROM)
0x470[8]	USDHC IOMUX SRE Enable	1	The fuse bit is used for ROM to enable SRE bit for SD pads	0 - Disable 1 - Enable	SW (ROM)
0x470[9]	USDHC_IOMUX_SION_BIT_ENABLE	1	The fuse bit is used for ROM to enable SION bit for MUX control register.	0 - Disable 1 - Enable	SW (ROM)

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x470[10]	Boot Failure Indicator Pin Select[4]	1	Indicator boot failure	00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 _ gpio1.IO31	SW (ROM)
0x470[11]	ENABLE_EMMC_22K_PULLUP	1	The fuse bit is used for ROM to enable 22K pullup for SD pads.	0 - 47K pullup 1 - 22K pullup	SW (ROM)
0x470[12]	USDHC_PAD_PULL_DOWN	1	The fuse bit is used for ROM to enable pull down bit for SD pads.	0 - no action 1 - pull down	SW (ROM)
0x470[13]	Override HYS bit for SD/MMC pads	1	Once this fuse be blown, the [HYS] bit of IOMUXC_SW_PAD_CTL_PAD_SDx_CLK(x is the SD port which the ROM boot from), IOMUXC_SW_PAD_CTL_PAD_SDx_CMD, IOMUXC_SW_PAD_CTL_PAD_SDx_DAT0-n(n will be 0, 3, or 7, depends on the bus width selected) will be set.		SW (ROM)
0x470[14]	eMMC 4.4 - RESET TO PRE-IDLE STATE	1	Once this fuse be blown, the CMD0 with argument 0xf0f0f0f0 will be sent to put the eMMC card into pre-IDLE state so that eMMC card's fast boot can work properly. This is useful for the warm boot, such as boot due to the WDOG reset.		SW (ROM)
0x470[15]	SD2_RST_POLARITY_SELECT	1	select the polarity of SD2	0 – Active Low ; 1 – Active high	SW (ROM)
0x470[19:16]	Boot Failure Indicator Pin Select[3:0]	4	Miscellaneous power management configuration bits.	00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31	SW (ROM)
0x470[20]	BT_LPB_POLARITY	1	Define GPIO3 polarity, for determining LPB boot mode.	0' - Active High '1' -Active Low	SW (ROM)
0x470[22:21]	LPB_BOOT	2	Defined the LowPower Boot options	(Core / DDR- Bus) '00' - LPB Disable '01' - 1 GPIO (def freq) '10' - Div by2 '11' - Div by 4	SW (ROM)
0x470[30:24]	MMC_DLL_DLY[6:0]	7	eMMC 4.4 delay line default value (set by boot rom), used in conjunction with "DLL	Connected to LVDS module	SW (ROM)

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
			Override" = 1 (BOOT_CFG3[3])		
0x470[31]	Override NAND Pad Settings	1	Override pad settings for NAND boot.		SW (ROM)
0x4F0[15:0]	USB_VID[31:0]	16	USB VID		SW
0x4F0[31:16]	USB_PID[31:0]	16	USB PID		SW
0x580[31:0]	SRK_HASH[255:0]	256	SRK key, no HW visible lines. NO HW Visible signals available		SW (HAB)
0x600[23:0]	SJC_RESP[55:0]	56	Response reference value for the secure JTAG controller		SJC
0x620[15:0]	MAC1_ADDR[47:0]	48	Ethernet1 MAC Address		SW
0x660[31:0]	GP1[31:0]	32	General Purpose fuse register #1		PROD / SW
0x670[31:0]	GP2[31:0]	32	General Purpose fuse register #2		PROD / SW
0x680[31:0]	SW_GP[159:0]	160	SW general purpose key		PROD / SW
0x6D0[5:0]	PAD_SETTINGS	6	Used with conjunction of MMC/SD/Nand "Override Pad Settings" fuse value, as follow: '0' - Use IO default settings for boot device IO pads. '1' - Use "Override" value, as set by this register.	IO pads settings of selected boot interface, are override with this fuses, as follow: [0] - Slew Rate [3:1] Drive Strength [5:4] - Speed Settings. Refer to IO PAD chapter for "Settings" fields value	SW (ROM)
0x6D0[6]	USB_VBUS_EVENT_HANDLER_EN	1	ROM handle USB VBUS attach or detach event	0 - ROM not handle USB VBUS attach or detach event 1 - ROM handle USB VBUS attach or detach event	SW (ROM)
0x6D0[7]	Enable boot failure indication pin	1	Enable boot failure indicator pin	0 - disable 1 - enable	SW (ROM)
0x6D0[11:8]	READ_RETRY_SEQ_ID[3:0]	4	Choose read retry sequence	0000 - don't use read retry(RR) sequence embedded in ROM 0001 - Micron 20 nm RR sequence 0010 - Toshiba A19 nm RR sequence 0011 - Toshiba 19 nm RR sequence 0100 - SanDisk 19 nm RR sequence	SW (ROM)

Table continues on the next page...

Table 5-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
				0101 - SanDisk 1ynmRR sequence 0110 - use Hynix 20 nm A die read retry sequence 0111 - use Hynix 26 nm read retry sequence 1000 - use Hynix 20 nm B die read retry sequence 1001 - use Hynix 20 nm C die read retry sequence Others - Reserved	
0x6D0[15:13]	WDOG Timeout Select	3	Select WDOG timeout	000 - 64s 001 - 32s 010 - 16s 011 - 8s 100 - 4s Others - Reserved	SW (ROM)
0x6D0[23:16]	NAND_READ_CMD_CODE1[7:0]	8	NAND_READ_CMD_CODE1	First command word to be used for Nand read.	SW (ROM)
0x6D0[31:24]	NAND_READ_CMD_CODE2[7:0]	8	NAND_READ_CMD_CODE2	Second command word to be used for Nand read.	SW (ROM)
0x6E0[0]	FIELD_RETURN	1	Configure device for field return testing. Fuse burning is protected by CSF command, with proper parameter passed. Write/OP protected by FIELD_RETURN_LOCK bit in control register.	0 - Device is in functional / secure mode. 1 - Device is open for field-return testing.	SW (ROM), SNVS_HP, SRC, TPSMP, Security Logic
0x900-0x9F0	GP3[511:0]	512	General Purpose fuse Register #3		PROD/SW
0xA00-0xAF0	GP4[511:0]	512	General Purpose fuse Register #4		PROD/SW
0xB00-0xBF0	GP5[511:0]	512	General Purpose fuse Register #5		PROD/SW
0xC00-0xC70	GP6[255:0]	256	General Purpose fuse Register #6		PROD/SW
0xC80-0xCB0	GP7[127:0]	128	General Purpose fuse Register #7		PROD/SW
0xCC0-0xCF0	GP8[127:0]	128	General Purpose fuse Register #8		PROD/SW

Chapter 6
External Memory Controllers

6.1 Overview

This chip has the following external memory interfaces and controllers:

- Multi-mode DDR controller (MMDC)
- EIM-PSRAM/NOR Flash controller

6.2 Multi-mode DDR controller (MMDC) overview and feature summary

The MMDC module is a DDR controller that can support several types of DDR memories.

Table 6-1. MMDC feature summary

Feature	Description
Supported standards	• LPDDR2 1ch x16, LV-DDR3, DDR3 x16

DDR interface	• Density of 256 Mbytes-4Gbytes
	  • Column size of 8-12 bits
	  • Row size of 11-16 bits
	• Supports burst length of 8 (aligned) for DDR3 and burst lengths of 4 for LPDDR2

DDR performance	• Supports Real-Time priority by means of QoS sideband priority signals from the chip to enable different priority levels in the re-ordering mechanism • Page hit/page miss optimizations • Consecutive read/write access optimizations • Supports deep read and write requests queues to enable bank prediction • Drives back the critical word in a read transaction as soon as it is received by the DDR device (doesn't wait until the whole data phase has been completed) • Can track open memory pages • Supports bank interleaving • Special optimization for non-aligned wrap accesses in burst length 8
AXI interface	• AXI bus compliant with glueless interface to PL301 AXI network interconnect

Table continues on the next page...

Table 6-1. MMDC feature summary (continued)

Feature	Description

DDR calibration and delay-lines	• All calibrations can be done automatically by hardware or manually by software
	• ZQ calibration for external DDR device (in DDR3 through the ZQ calibration command and in LPDDR2 through the MRW command).
	  • Can be handled automatically for ZQ Short (periodically) and ZQ Long (at exit from self-refresh).
	  • Can be handled manually at ZQ INIT.

DDR general	• Configurable timing parameters • Configurable refresh scheme • Supports dynamic voltage, frequency change and low power mode entry through hardware negotiation with the system (req/ack handshake) • Suppors automatic self-refresh and power down entry and exit • Supports fast and slow precharge power down in DDR3 • Supports various ODT control schemes. • Assertion/Deassertion of ODT control per read or write accesses and for active or passive CS • Supports MRW and MRR commands for LPDDR2. • Software control for moving to derated timing parameters and derated refresh rate according to temperature variation. • Supports various debug and profiling modes

6.3 EIM-PSRAM/NOR Flash controller overview

EIM is an external interface module and manages the interface to devices external to the chip, including the generation of chip selects, clocks, and control for external peripherals and memory.

It provides asynchronous and synchronous access to devices with an SRAM-like interface.

6.3.1 EIM features

- Up to four (software configurable) chip selects for external devices
  - Flexible address decoding; each chip select memory space is determined separately according to the GPR bits in IOMUXC.
  - 128 MByte maximum supported density by default (AUS bit is cleared). When the AUS bit is set, maximum supported density is 32MBytes.
- Selectable write protection for each chip select
- Programmable wait-state generator for each chip select, for write and read accesses separately
- Asynchronous accesses with programmable setup and hold times for control signals

- Independent synchronous memory burst write mode support for PSRAM and NOR-Flash like memories (CellularRAM™ from Micron, Infineon, and Cypress, OneNAND™ and utRAM™ from Samsung, and COSMORAM™from Toshiba)
- Supports NAND-Flash devices with NOR-Flash like interface - OneNAND™ (Samsung)
- Independent programmable variable/fix latency support for read and write synchronous (burst) mode
- Support for little endian operation
- ARM AXI slave interface accesses only handled in parallel for single AXI ID transactions
- External interrupt support using the RDY_INT signal function as an external interrupt pin
- Boot from external device support according to boot signals, using the RDY_INT signal
  - RDY signal support assertion after reset
  - INT signal support assertion after reset for OneNAND™ (Samsung) device
  - Supports little endian mode only

6.3.2 EIM boot scenarios

EIM allows booting from NOR Flash devices. To select NOR Flash as the boot source, use either the boot mode and configuration GPIO pins or the internal boot-related fuses.

See Sytem Boot for more information.

6.3.3 EIM boot configuration

The following table shows the EIM boot configuration.

Table 6-2. EIM boot configuration

EIM_BOOT_CFG bus	EIM affected bits	EIM register
12	NUM16_BYP_GRANT	CS0GCR2
11	DSZ[2]	CS0GCR1
10	AUS	CS0GCR1
[9:8]	CSREC[2:1]	CS0GCR1
[7:5]	RWSC[4:2]	CS0GCR1
	WWSC[4:2]	CS0WCR
4	ERRST	WCR
3	RAL	CS0RCR1

Table continues on the next page...

Table 6-2. EIM boot configuration (continued)

EIM_BOOT_CFG bus	EIM affected bits	EIM register
	WAL	CS0WCR
2	MUM	CS0GCR1
	OEA[1]	CS0RCR1
[1:0]	DSZ[1:0]	CS0GCR1

6.3.4 OneNAND requirements

By default Ready/Busy pin is not in use, take the following steps to configure OneNAND:

- Poll the device to see whether it is ready; software performs a read from the device.
- Connect the Ready/Busy signal of the OneNAND device to any GPIO pin and use it as an interrupt that indicates the OneNAND device's ready state.

Chapter 7
System Debug

7.1 Overview

This chapter describes the hardware and software debug and application development features and resources of the chip. It discusses the following:

- Core/platform-specific resources
- Resources associated with complex IP blocks
- Chip-wide resources
- Interface to the external debug and development tools

The debug and trace architecture is designed around the following:

- ARM's CoreSight architecture, adapted to SoC (for core debug), including a cross-trigger subsystem for cross-domain triggering of debug resources
- JTAG port used to interact with cores under debug by means of SJC, the system JTAG controller port
- DAP, the debug access port that supports the interface to the ARM RealView Debugging tools and other third party tools
- TPIU, a trace port interface unit that efficiently accesses program trace information from the system
- Various chip-wide resources, such as debug features built into the IP blocks and critical signal visibility available through alternate pin functions or observability muxes

7.2 Chip and ARM Platform Debug Architecture

ARM Debug architecture is based on CoreSight architecture by ARM Ltd. The CoreSight architecture provides a system wide solution to real-time debug and trace.

The CoreSight architecture is embodied in a set of CoreSight components and compliant processors that form CoreSight systems. Its architecture maintains the traditional requirements of debug and trace:

- To access debug functionality without software interaction;
- To connect to a running system without performing a reset.

Full access to the processor debug capability is available by the ARM debug register map through the Advanced Peripheral Bus (APB) slave port. The core includes a Processor Debug Unit allow which stops program execution, examines and alters the processor and coprocessor state, examines and alters the memory and input/output peripheral state and restarts the processor core.

7.2.1 Debug Features

- CoreSight Embedded Trace Macrocell (ETM): trace generator for ARM processors
- Support for a TrustZone-related 3-level debug scheme:
  - Debug in Non-Secure privileged and user, and Secure user
  - Debug in Non-Secure only
- EmbeddedICE-RT logic
  - Support for both monitor-mode and halt-mode debugging:
  - Core run/halt control, debug status/control
  - Breakpoint/watchpoint control
  - Core- and memory-mapped resource examination/modification
- Data communication channel between ARM core and host debugger via JTAG, and the Debug Access Port (DAP) module.
- PMU: Performance Metrics Unit used for system profiling and debug.
- CP15 register for debugging the MMU, I & D L1 cache, and TLB

The chip includes ARM CoreSight components for multicore debug and trace solutions.

7.2.2 Debug System components

CoreSight components include:

- Embedded Trace Buffer (ETB): RAM array to be used for on-chip capture of trace data output from the PTM
- ATB Replicator to connect the trace data to TPIU (Trace Port Interface) and ETB (Embedded Trace Buffer)
- Cross Triggering logic for event routing, including CTIs and CTMs

Other related IPs and functionality:

- ROMPATCH module to support modification of program/data information in the MCU ROM
- Debug Visibility, which selects critical signals routed to the I/O pads as alternate outputs for external visibility

7.2.2.1 AMBA trace bus (ATB)

ATB transfers trace data though the CoreSight infrastructure in a chip. Trace sources are ATB masters and sinks are ATB slaves. The ARM (via PTM) cores are the data generators. Link components such as the Trace Funnel and Replicator provide both master and slave interfaces.

The ATB protocol supports:

- Stalling of trace sources to enable the CoreSight components to funnel and combine sources into a single trace stream.
- Association of the trace data with the generating source using trace source IDs. A CoreSight system can trace up to 111 different items at any one time.
- Capture and transfer of multiple byte bus widths, currently to 32 bits.
- A flushing mechanism to force historic trace to drain from any sources, links, or sinks up to the point that the request was initiated

7.2.2.2 ATB replicator

The ATB replicator enables two trace sinks to be wired together and to operate from the same incoming trace stream.

There are no programmable registers. This component is invisible to the user on a particular trace path, from source to sink.

- Incoming ATB Interface—The ATB replicator accepts trace data from the trace source, either directly or through a trace funnel.
- Outgoing ATB Interfaces—The ATB replicator sends identical trace data on outgoing master port interfaces.

7.2.2.3 Embedded Cross Triggering

The ECT is a modular component from ARM Limited that supports the interaction and synchronization of multiple triggering events within a chip. The main function of the ECT (CTI and CTM) is to pass debug events from one core to another. For example, the ECT can communicate debug state information from one core to another, so that program execution on both processors can be stopped at the same time if required.

The ECT consists of the following types of modules:

- Cross trigger interfaces (CTI)
- Cross trigger matrix (CTM)

Cross trigger interfaces provide the interface between a component or subsystem and the cross trigger matrix. The system requires a CTI for each subsystem that supports cross triggering. The CTI combines and maps the trigger requests, and broadcasts them to all other interfaces on the ECT as channel events. When the CTI receives a channel event it maps this onto a trigger output. This enables subsystems to cross trigger with each other. The receiving and transmitting of triggers is performed through the trigger interface.

The Cross Trigger Matrix (CTM) combines the trigger request generated from the CTIs and broadcasts them to all CTIs as channel triggers. The CTM controls the distribution of channel events. It provides Channel Interfaces (CIs) to connect to either CTIs or CTMs. This enables multiple CTIs to be linked together. The ECT is composed of three/five CTIs (Cross Trigger Interface) and two CTMs (Cross Trigger Matrix). The ECT is key in the multi-core and multi-IP debug strategy. The outcome is a SWcontrolled debug signal matrix that receives signals from various sources (i.e. cores and peripherals) and propagates/ routes them to the different debug resources of the SoC. Those debug resources can include time stamping capability, profiling capabilities, real-time trace (trace enabled or disabled), triggers, SOC level multiplexing, and debug interrupts.

NOTE

As the ECT should only be used during debug sessions, it is disabled by default.

7.2.2.3.1 Cross-Trigger Matrix (CTM)

The CTM (Cross Trigger Matrix) is provided by ARM. A brief description is provided below. It is advised to refer to ARM documentation for more details.

The CTM is a relatively simple block with no configuration options. There are two CTM instances in in ARM Platform.

One of them is used to route Core's CTI's, while the second one, is used for additional CoreSight CTI. Each CTI has 4 channel lines, which CTI events are mapped to. The exact mapping is configured in the CTI logic.

7.2.2.3.2 Cross-Trigger Interface (CTI)

The Cross-Trigger Interface (CTI) component is provided by ARM. A brief description of the CTI is provided below.

There are CTIs in the chip's ARM platform. of which, dedicated to each Core, while the is used for various other signals routing.

Each of these CTIs has 8 trigger inputs and 8 trigger outputs that connect to logic in the domain to be debugged or profiled. Each CTI also includes a 4 channel interface to the CTM (4 inputs and 4 outputs).

For more information, see ARM platform chapter.

7.2.2.4 Debug Access Port (DAP)

The DAP enables debug access to the chip modules through APB-AP (the APB access port) and APB-Mux (the APB multiplexer).

AHB-AP provides system access. Debug tools can use JTAG to connect to the chip.

DAP has the following features:

- AMBA 3 Peripheral Bus Multiplexor access though AMBA 3 APB Access Port, providing debug peripheral access through the APB interface.
- External JTAG access using the JTAG Debug Port (JTAG-DP).
- Internal chip module access using:
  - AHB Access Port (AHB-AP)
  - APB Access Port (APB-AP)
  - JTAG Access Port (JTAG-AP)

APB-Mux enables system access to CoreSight components connected to the Debug APB.

The ROM table provides a list of memory locations of CoreSight components connected to the Debug APB. This is visible from both tools and system access and one configures it during system implementation.

External read/write access to the internal interface is provided by JTAG-DP. JTAG-DP provides a standard interface for debug access to the chip through DAP. It interfaces to the DAP internal bus.

Internal access to on-chip buses and other interfaces are provided by the access ports (APs). The available APs are:

- AHB-AP which provides an AHB-Lite master for access to a system AHB bus.
- APB-AP which provides an AMBA 3 APB master for access to the Debug APB that configures all CoreSight components.
- JTAG-AP which provides JTAG access to on-chip components and operates as a JTAG master port to drive JTAG chains throughout the chip.

7.2.3 Chip-Specific SJC Features

7.2.3.1 JTAG Disable Mode

In addition to different JTAG security modes that are implemented internally in the System JTAG Controller (SJC), there is an option to disable the SJC functionality by e-fuse configuration.

This creates additional JTAG mode "JTAG Disabled" with highest level of JTAG protection. In this mode all JTAG features are disabled. Specifically, the following debug features are disabled in addition to the features that were already disabled in "No Debug" JTAG mode:

- Non-Secure JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secure JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

7.2.3.2 JTAG ID

Table 7-1. i.MX JTAG ID

Device	Silicon revision	JTAG ID (ID CODE)
i.MX 6UltraLite	Rev 1.0	0891_D01Dh 1891_D01Dh [1]

1. In follow-on silicon revisions, the ID value is subject to change by incrementing the first nibble as follows: 1891_D01Dh for Rev 1.1, 2891_D01Dh for Rev 1.2 , etc.

7.2.4 System JTAG Controller - SJC

The SJC module is the bridge between external development and test instrumentation and the internal JTAG-accessible debug and test resources.

It implements and manages the daisy-chained topology consisting of its own TAP and those of the SDMA, and the ARM Debug Access Port (DAP).

NOTE

Single Wire Debug (SWD) protocol is not supported.

7.2.5 System JTAG controller main features

- IEEE P1149.1, 1149.6 (standard JTAG) interface to off-chip test and development equipment
  - Includes an SJC-only mode for true IEEE P1149.1 compliance, used primarily for board-level implementation of boundary scan.
  - Supports IEEE P1149.6 extensions to the JTAG standard for AC testing of selected I/O signals.
- Debug-related control and status; puting selected cores into reset and/or debug mode and monitoring individual core status signals by means of JTAG
- System status, such as the state of the PLLs (locked or not locked)
- levels of security, ranging from no security to no JTAG accessibility to the chip

7.2.6 SJC TAP Port

The SJC supports the following standard JTAG pins:

- TRSTB
- TDI
- TDO
- TCK
- TMS

7.2.7 SJC main blocks

- Interface to the outside world via the standard JTAG pins
- Interface to the external Debug_Event pin
- A master TAP controller which implements the standard JTAG state machine
- Implementation of the mandatory and optional IEEE P1149.1 (JTAG) instructions

    - Mandatory: "EXTEST", "SAMPLE/PRELOAD", and "BYPASS"
    - Optional: "ID_CODE" (SOC JTAG ID register), "HIGHZ"
- Supports the SDMA's DR-path-only JTAG architecture by implementing the controller portion of its TAP (including "BYPASS" as the default state) within the SJC
- The ExtraDebug registers, which implement a variety of control and status features
    - Three 32-bit insecure general purpose status registers
    - Two 32-bit secure status registers - one predefined, one general purpose.
    - Control and status registers for debug, core, charge pump, and PLL.
- Four levels of fuse-defined security, ranging from no security to no access.

Both predefined and user-defined control and status functions are supported by the SJC.

7.3 Smart DMA (SDMA) core

SDMA is a dedicated, programmable DMA engine. It is an integration of a 32-bit RISC core and DMA-specific hardware. It includes ports for the AP domain and a peripheral domain, along with a burst-capable port for direct external memory access.

NOTE

The SDMA and its integration in the chip is unchanged from previous i.MX chips.

The main SDMA debug features are:

- OnCE - On Chip Emulator, provides the following capabilities:
    - SDMA core control - run/halt/single-step
    - SDMA core register/memory-map access
    - Event detection, watchpoints, and hardware breakpoints
    - Real time buffer and PC trace buffer capability
- Trace buffer
    - Contains information to identify the 32 last changes of flow detected during a program execution
- Context dump
    - Includes information about all the channel dump activity
    - Current contents of SDMA RAM
- ROMPATCH

7.3.1 SDMA On Chip Emulation Module (OnCE) Feature Summary

The SDMA debug features are primarily defined by the OnCE portion of its design.

They are summarized as follows:

- Memory And Register Access - dedicated logic enables user-access to SDMA memory and register locations. These accesses are supported only when the processor is in debug mode.
- Event Detection Unit - watches signals from the data memory bus (DMBus) which is used by the RISC core to access its RAM, ROM, and memory-mapped registers
- Watchpoints - one output signal is available to watch event matching conditions at the chip level. Match conditions are defined by programming memory-mapped registers.
- Hardware Breakpoint - a counter is decremented after an event detection. A debug request is sent to the SDMA core only when the counter reaches the value of zero. It is possible to program the initial value of the counter or to disable the use of the counter if a debug request must be generated after each event detection.
- Real Time Buffer - The Real Time Buffer Register (RTB) is a single 32-bit memory-mapped register which can be accessed as a regular memory location during program execution. It is used to store and retrieve run time information without putting the SDMA in debug mode. Each write to this register causes an event. This register is, in fact, located in the OnCE. Executing through JTAG, a buffer command exports the content of this register through the JTAG port.
- Core Control (Core Status / Single Stepping) - Commands are provided to monitor and control processor activity. The commands can halt the core, rerun the core from another address location, and get processor status.
- Trace Buffer - a 32x32 buffer which records the last 32 changes of flow during program execution. The buffer stores data in a modulo fashion (i.e. the 33rd instruction change replaces the 1st). Captured trace information is retrieved via reads to the Trace Buffer Register.

7.3.1.1 Other SDMA Debug Functionality

- Core Trace - basic core trace capability is available through debug visibility functionality only. PTM trace capability does not exist.
- ROM Patch - can be accomplished by manipulating the CHN0ADDR register through JTAG or via the MCU's ability to write to SDMA OnCE registers. This must be done right after reset and before the SDMA core is enabled to begin processing events.
- Additional debug control/status interaction with the SJC module

- SJC-controlled Debug Request
- SJC-readable Debug Acknowledge (in debug mode)
- Debug clock control - allows SJC to force clocks on for debug purposes
- Debug core state (SDMA RISC Core State) - 4 bits accessible from the SJC via JTAG

7.3.1.2 SDMA ROM Patching

After reset, the SDMA is in its IDLE_AFTER_RESET mode. A debug request also puts the SDMA in its DEBUG_IN_IDLE_AFTER_RESET mode. The new address boot must be stored in CHN0ADDR register (e.g., through the SDMA OnCE via debugger).

The user must then issue the exec_core <instruction> SDMA OnCE instruction to return to the IDLE_AFTER_RESET mode. The very first instructions of the boot code fetches the contents of this register (which is also mapped in the SDMA memory space) and jumps to the given address.

7.4 Miscellaneous

The Miscellaneous function described in this section provide useful general capabilities.

7.4.1 Clock/Reset/Power

CDBGPWRUPREQ and CDBGPWRUPACK are the handshake signals between the DAP and the clock control module to ensure debug power and clocks are turned on. If the debug components are always powered on, the handshake becomes a mechanism to turn debug clocks on. Similarly, there is a register bit in the CCM which allows internal software to turn debug clocks on as well because the CDBGPWRUPREQ is in the TCLK domain and is inaccessible to software.

The Cortex-A7 core can receive resets from the following sources:

- Debug Reset (CDBGRSTREQ bit within the SWJ-DP CTRL/STAT register of the DAP) in the TCLK domain. This allows the debug tools to reset the debug logic.
- System POR reset

7.5 Supported tools

DS-5 ARM Debugger is supported.

The debugger is connected to the chip from the host by the DS-5 ICE protocol converter. Other third party tools can be used via the standard JTAG interface, but may need to be adapted for individual IC. It is important to check with tool vendors for specific tool requirements, especially for on-chip IC.

Chapter 8
System Boot

8.1 Overview

The boot process begins at Power On Reset (POR) where the hardware reset logic forces the ARM core to begin execution starting from the on-chip boot ROM.

Boot ROM code uses the state of the internal register BOOT_MODE[1:0] as well as the state of various eFUSEs and/or GPIO settings to determine the boot flow behavior of the device.

The main features of the ROM include:

- Support for booting from various boot devices
- Serial downloader support (USB OTG and UART)
- Device configuration data (DCD) and plugin
- Digital signature and encryption based High Assurance Boot (HAB)
- Wake-up from low power modes

The boot ROM supports the following boot devices:

- NOR Flash
- NAND Flash
- OneNAND Flash
- SD/MMC
- Serial (SPI) NOR Flash and EEPROM
- QuadSPI (QSPI) Flash

The Boot ROM uses the state of BOOT_MODE and eFUSEs to determine the boot device. For development purposes, eFUSEs used to determine the boot device may be overridden by using GPIO pin inputs.

Boot ROM code also allows the downloading of programs to be run on the device. An example is a provisioning program that can make further use of the serial connection to provision a boot device with a new image. Typically the provisioning program is

downloaded to internal RAM and allows the programming of boot devices, such as an SD/MMC Flash. The ROM Serial Downloader uses high speed USB in a non-stream mode connection and UART in a stream mode connection.

Boot ROM allows waking up from low-power modes. On reset the ROM checks power gating status register. On waking from low power mode, the core will skip loading an image from the boot device and jump to the address saved in PERSISTENT_ENTRY0.

The device configuration data (DCD) feature allows boot ROM code to obtain SOC configuration data from an external Program Image residing on the boot device. As an example, DCD can be used to program the DDR controller for optimal settings improving the boot performance. DCD is restricted to memory areas and peripheral addresses that are considered essential for boot purposes (see Write Data Command).

A key feature of the boot ROM is the ability to perform a secure boot or High Assurance Boot (HAB). This is supported by the HAB security library which is a subcomponent of the ROM code. HAB uses a combination of hardware and software together with a Public Key Infrastructure (PKI) protocol to protect the system from executing unauthorized programs. Before the HAB allows a user's image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as part of the final Program Image. If configured to do so, the ROM verifies the signatures using the public keys included in the Program Image. In addition to supporting digital signature verification to authenticate Program Images, Encrypted boot is also supported. Encrypted boot can be used to prevent cloning of the Program Image directly off the boot device. A secure boot with HAB can be performed on all boot devices supported on the chip in addition to the Serial Downloader. The HAB library in the boot ROM also provides API functions, allowing additional boot chain components (bootloaders) to extend the secure boot chain. The out-of-fab setting for SEC_CONFIG is the Open configuration in which the ROM/HAB performs image authentication, but all authentication errors are ignored and the image is still allowed to execute.

8.2 Boot modes

During reset, the chip checks Power Gating Controller status register.

During boot, the core's behavior is defined by the Boot Mode pins settings as described in Boot mode pin settings. On waking up from low power boot mode, the core skips clock settings. Boot ROM checks that PERSISTENT_ENTRY0 (see Persistent Bits) is a pointer to valid address space (OCRAM, DDR, QSPI, or EIM). If

PERSISTENT_ENTRY0 is a pointer to valid range, it starts execution using entry point from PERSISTENT_ENTRY0 register. If PERSISTENT_ENTRY0 is a pointer to invalid range, the core performs system reset.

8.2.1 Boot mode pin settings

The device has four boot modes (one is reserved for Freescale use). Boot mode is selected based on the binary value stored in the internal BOOT_MODE register.

BOOT_MODE is initialized by sampling the BOOT_MODE0 and BOOT_MODE1 inputs on the rising edge of POR_B. After these inputs are sampled, their subsequent state does not affect the contents of the BOOT_MODE internal register. The state of the internal BOOT_MODE register may be read from the BMOD[1:0] field of the SRC Boot Mode Register (SRC_SBMR2). The available boot modes are: Boot From Fuses, serial boot via USB, and Internal Boot. See the table below for settings.

Table 8-1. Boot MODE Pin Settings

BOOT_MODE[1:0]	Boot Type
00	Boot From Fuses
01	Serial Downloader
10	Internal Boot
11	Reserved

8.2.2 High level boot sequence

The figure found here shows the high-level boot ROM code flow.

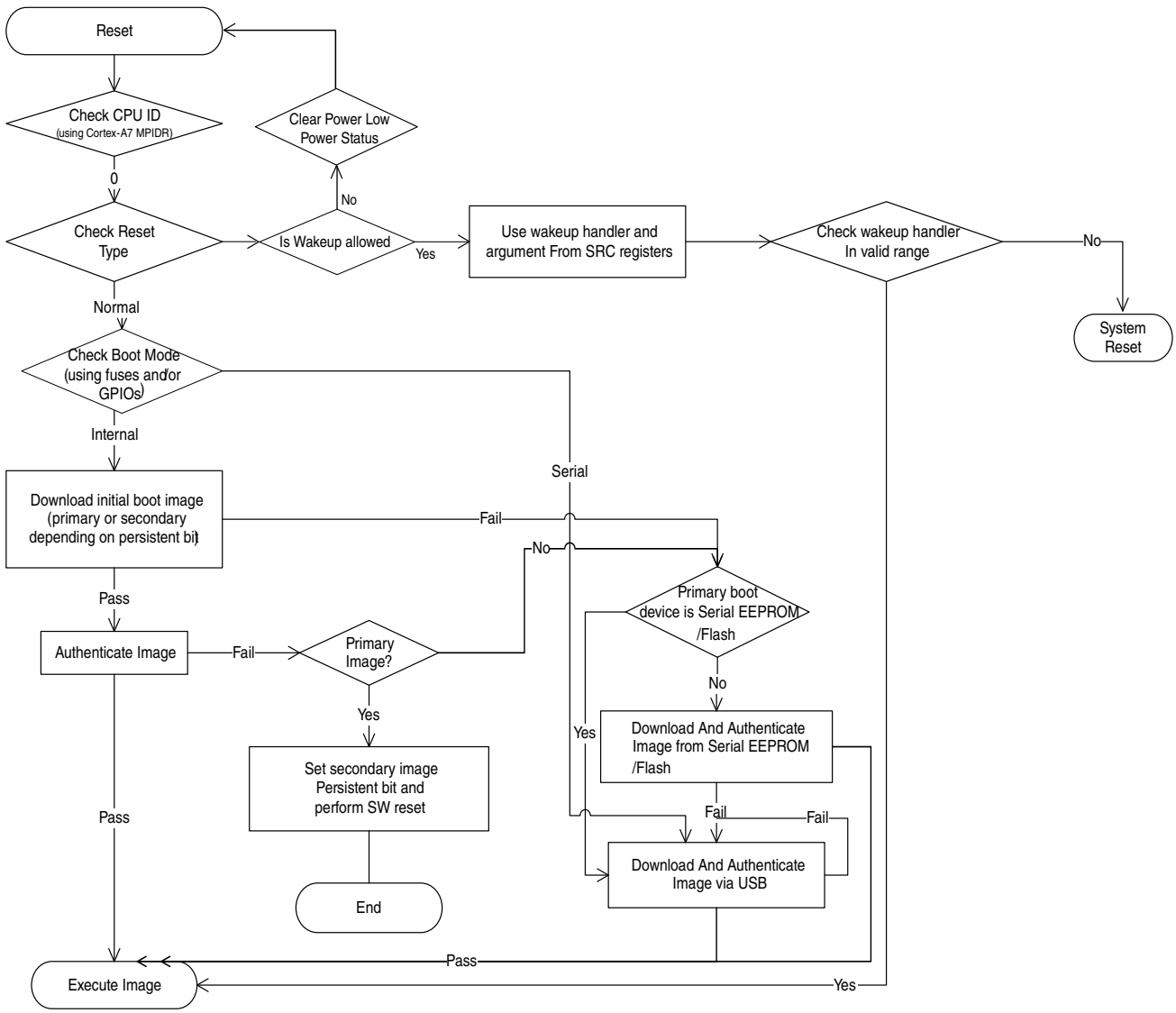


Figure 8-1. Boot Flow

8.2.3 Boot From Fuses Mode (BOOT_MODE[1:0] = 00b)

A value of 00b in the BOOT_MODE[1:0] register selects the Boot From Fuses mode.

This mode is similar to the Internal Boot mode described in Internal Boot Mode (BOOT_MODE[1:0] = 0b10) with one difference. In this mode the GPIO boot override pins are ignored. The boot ROM code uses the boot eFUSE settings only. This mode also supports a secure boot using HAB.

If set to Boot From Fuses, the boot flow is controlled by the BT_FUSE_SEL eFUSE value. If BT_FUSE_SEL = 0, indicating that the boot device (for example, Flash, SD/MMC) has not yet been programmed, the boot flow jumps directly to the Serial Downloader. If BT_FUSE_SEL = 1, the normal boot flow is followed, where the ROM attempts to boot from the selected boot device.

The first time a board is used, the default eFUSEs may be configured incorrectly for the hardware on the platform. In such a case, the Boot ROM code may try to boot from a device that does not exist. This may cause an electrical/logic violation on some pads. Using Boot From Fuses mode addresses this problem.

Setting BT_FUSE_SEL=0 forces the ROM code to jump directly to the Serial Downloader. This allows a bootloader to be downloaded which can then provision the boot device with a Program Image and blow the BT_FUSE_SEL and the other boot configuration eFUSEs. After reset, the Boot ROM code determines that BT_FUSE_SEL is blown (BT_FUSE_SEL = 1) and the ROM code performs internal boot according to the new eFUSE settings. This allows a user to set BOOT_MODE[1:0]=00b on a production device and burn fuses on the same device (by forcing entry to the Serial Downloader), without changing the value of BOOT_MODE[1:0] or pullups/pulldowns on the BOOT_MODE pins.

8.2.4 Serial Downloader

The Serial Downloader provides a means to download a Program Image to the chip over USB and UART serial connection.

In this mode the ROM programs WDOG1 for a time-out specified by fuse WDOG Time-out Select (See fusemap for details) if WDOG_ENABLE eFuse is 1 and continuously polls for USB and UART connection. If no activity is found on USB OTG1, UART 1/2 and the watchdog timer expires, the ARM core is reset.

NOTE

After being loaded, the downloaded image is responsible to manage the watchdog resets properly.

Figure 8-2 shows USB and UART boot flow.

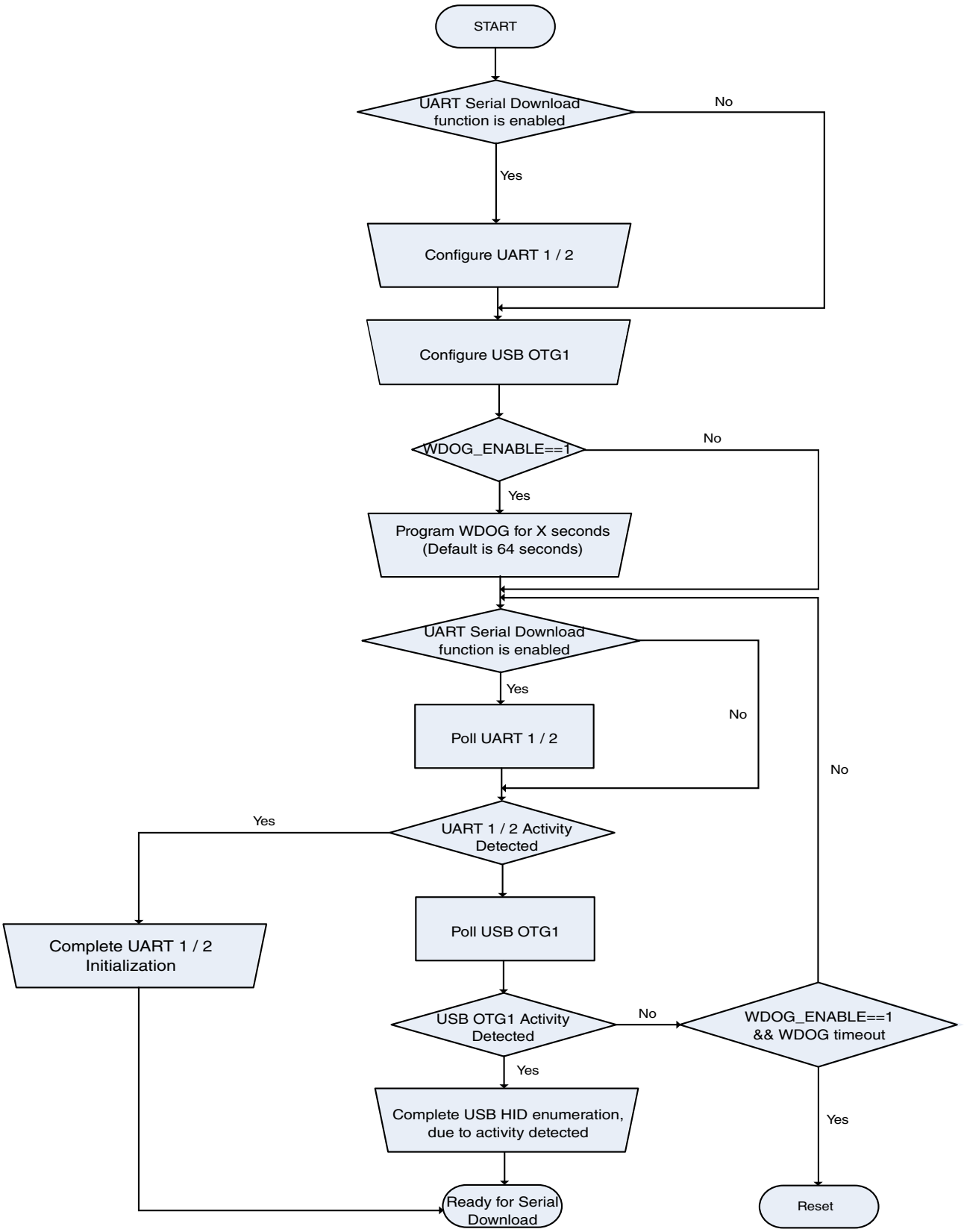


Figure 8-2. Serial Download Boot Flow

8.2.5 Internal Boot Mode (BOOT_MODE[1:0] = 0b10)

A value of 0b10 in the BOOT_MODE[1:0] register selects the internal boot mode. In this mode, the processor continues to execute boot code from the internal boot ROM.

The boot code performs hardware initialization, loads the Program Image from the chosen boot device, performs image validation using the HAB library (see Boot security settings), and then jumps to an address derived from the Program Image. If any error occurs during internal boot, the boot code jumps to the Serial Downloader (see Serial Downloader). A secure boot using the HAB is possible in all the three boot modes.

When set to internal boot, the boot flow may be controlled by a combination of eFUSE settings with an option of overriding the fuse settings using General Purpose I/O (GPIO) pins. The GPIO Boot Select FUSE (BT_FUSE_SEL) determines whether the ROM uses GPIO pins for a select number of configuration parameters or eFUSEs in this mode.

- If BT_FUSE_SEL = 1, all boot options are controlled by the eFUSEs described in Table 8-2.
- If BT_FUSE_SEL = 0, specific boot configuration parameters may be set using GPIO pins rather than eFUSEs. The fuses that can be overridden when in this mode are indicated in the GPIO column of Table 8-2. Table 8-3 provides the details on the GPIO pins.

The use of GPIO overrides is intended for development since these pads are used for other purposes in deployed products. Freescale recommends controlling the boot configuration by eFUSEs in deployed products and reserving the use of the GPIO mode for development and testing purposes only.

8.2.6 Boot security settings

Internal boot modes use one of three security configurations:

- Closed: This level is intended for use with shipping secure products. All HAB functions are executed and security hardware is initialized (the Security Controller, or SNVS, enters Secure state), DCD is processed if present, and the program image is authenticated by HAB prior to its execution. All detected errors will be logged, and the boot flow aborted with control passing to the serial downloader. At this level, execution does not leave the internal ROM unless the target executable image has been authenticated.
- Open: This level is intended for use in non-secure products or during the development phases of a secure product. All HAB functions are executed as for a

closed device. Security hardware is initialized (except the SNVS is left in Non-Secure state), DCD is processed if present, and the program image is authenticated by HAB prior to its execution. All detected errors will be logged, but have no influence on the boot flow, which continues as if the errors did not occur. This configuration is useful for secure product development, since the Program Image will run even if the authentication data is missing or incorrect, and the error log can be examined to determine the cause of authentication failure.

- Field Return: This level is intended for parts returned from shipped products.

NOTE

If the DIR_BT_DIS eFuse is not blown, authentication may be bypassed. In this case the system is not secure.

8.3 Device Configuration

This section describes the external inputs that control the behavior of the Boot ROM code.

This includes boot device selection (SPI, EIM, NOR, SD, MMC, QSPI, etc.), boot device configuration (SD bus width, speed, etc), and so on. In general, the source for this configuration comes from eFUSEs embedded inside the chip. However, certain configuration parameters can be sourced from GPIO pins allowing further flexibility during the development process.

8.3.1 Boot eFUSE Descriptions

The table below is a comprehensive list of the configuration parameters that the ROM uses.

Table 8-2. Boot eFUSE Descriptions

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
DIR_BT_DIS	OEM	Disables Freescale reserved modes. Must be set for secure boot.	NA	0	0 Reserved Freescale modes enabled 1 Reserved Freescale modes disabled
BT_FUSE_SEL	OEM	In internal Boot mode BOOT_MODE[1:0] = 10, the BT_FUSE_SEL fuse determines whether the boot settings indicated by a Yes in the GPIO column are	NA	0	If BOOT_MODE[1:0] = 0b10 0 Bits of SBMR are overridden by GPIO pins. 1 Specific bits of SBMR are controlled by eFUSE settings.

Table continues on the next page...

Table 8-2. Boot eFUSE Descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
		controlled by GPIO pins or eFUSE settings in the On-Chip OTP Controller (OCOTP). In Boot From Fuse mode BOOT_MODE[1:0] = 00, BT_FUSE_SEL fuse indicates whether bit configuration eFuses have been programmed.			If BOOT_MODE[1:0] = 0b00 0 BOOT configuration eFuses are not yet programmed. Boot flow jumps to serial downloader. 1 BOOT configuration eFuses have been programmed. Regular boot flow is performed.
SEC_CONFIG[1:0]	SEC_CONFIG[0] - Freescale SEC_CONFIG[1] - OEM	Security Configuration as defined in Boot security settings	NA	01	00 Reserved 01 Open (allows any program image, even if authentication fails) 1x Closed (Program image executes only if authenticated)
FIELD_RETURN	OEM	Enables Freescale reserved modes			0 - Freescale reserved modes are enabled/ disabled based on DIR_BT_DIS value 1 - Freescale reserved modes are enabled
SRK_HASH[255:0]	OEM	256-bit hash value of super root key (SRK_HASH)	NA	0	Settings vary - used by HAB
DIE-X-CORDINATE[7:0] DIE-Y-CORDINATE[7:0] WAFER_NO[4:0] LOT_NO_ENC[42:40] LOT_NO_ENC[39:32] LOT_NO_ENC[31:24] LOT_NO_ENC[23:16] LOT_NO_ENC[15:8] LOT_NO_ENC[7:0]	Freescale	Device Unique ID, 64-bit UID.	NA	Unique ID	Settings vary - used by HAB
BT_MMU_DISABLE	OEM	MMU/L1 D Cache/PL310 disable bit used by boot ROM for fast HAB processing	No	0	0 - MMU/L1 D Cache/PL310 is enabled by ROM during the boot 1 - MMU/L1 D Cache/PL310 is disabled by ROM during the boot
L1 I-Cache DISABLE	OEM	L1 I Cache disable bit used by boot during entire execution	No	0	0 - L1 I Cache is enabled by ROM during the boot 1 - L1 I Cache is disabled by ROM during the boot

Table continues on the next page...

Table 8-2. Boot eFUSE Descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
BT_FREQ	OEM	Boot Frequency Selection	Yes	0	0 - ARM - 396 MHz, DDR - 396 MHz, AXI - 198 MHz 1 - ARM - 198 MHz, DDR - 307 MHz, AXI - 153 MHz
BOOT_CFG1[7:0]	OEM	Boot Configuration1	Yes	0	Specific to selected boot mode
BOOT_CFG2[7:0]	OEM	Boot Configuration2	Yes	0	Specific to selected boot mode
BOOT_CFG4[6:0]	OEM	Boot Configuration4		0	Specific to selected boot mode
BOOT_CFG4[7]	OEM	Infinite Loop Enable at start of boot ROM. Used for debugging purposes. Ignored if DIR_BT_DIS is 1 and FIELD_RETURN is 0.	Yes	0	0 - Disabled 1 - Enabled
LPB_BOOT	OEM	USB Low Power Boot	No	0	00 - 396 MHz 01 - 396 MHz 10 - 198 MHz 11 - 99 MHz
BT_LPB_POLARITY	OEM	USB Low Power Boot GPIO polarity	No	0	0 - low on GPIO pad indicates low power condition 1 - high on GPIO pad indicates low power condition
WDOG_ENABLE	OEM	Watchdog reset counter enable	No	0	0 - watchdog reset counter is disabled during serial downloader 1 - watchdog reset counter is enabled during serial downloader
SRK_REVOKE[2:0]	OEM	SRK revocation mask	No	0	SRK revocation mask
DISABLE_SDMMC_MFG	OEM	Disable SDMMC manufacture mode	Yes	0	0: enable SD/MMC MFG mode 1: disable SD/MMC MFG mode
PAD_SETTINGS	OEM	Override values for SD/MMC and NAND boot modes	No	0	Override the following IO PAD settings: PAD_SETTINGS[0] - Slew Rate PAD_SETTINGS[3:1] Drive Strength PAD_SETTINGS[5:4] - Speed Settings .

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See GPIO Boot Overrides for corresponding GPIO pin.
2. 0 = intact fuse and 1= blown fuse

8.3.2 GPIO Boot Overrides

The table below provides a list of GPIO boot overrides.

Table 8-3. GPIO Override Contact Assignments

Package Pin	Direction on reset	eFuse
BOOT_MODE1	Input	Boot Mode Selection
BOOT_MODE0	Input	
LCD1_DATA00	Input	BOOT_CFG1[0]
LCD1_DATA01	Input	BOOT_CFG1[1]
LCD1_DATA02	Input	BOOT_CFG1[2]
LCD1_DATA03	Input	BOOT_CFG1[3]
LCD1_DATA04	Input	BOOT_CFG1[4]
LCD1_DATA05	Input	BOOT_CFG1[5]
LCD1_DATA06	Input	BOOT_CFG1[6]
LCD1_DATA07	Input	BOOT_CFG1[7]
LCD1_DATA08	Input	BOOT_CFG2[0]
LCD1_DATA09	Input	BOOT_CFG2[1]
LCD1_DATA10	Input	BOOT_CFG2[2]
LCD1_DATA11	Input	BOOT_CFG2[3]
LCD1_DATA12	Input	BOOT_CFG2[4]
LCD1_DATA13	Input	BOOT_CFG2[5]
LCD1_DATA14	Input	BOOT_CFG2[6]
LCD1_DATA15	Input	BOOT_CFG2[7]
LCD1_DATA16	Input	BOOT_CFG4[0]
LCD1_DATA17	Input	BOOT_CFG4[1]
LCD1_DATA18	Input	BOOT_CFG4[2]
LCD1_DATA19	Input	BOOT_CFG4[3]
LCD1_DATA20	Input	BOOT_CFG4[4]
LCD1_DATA21	Input	BOOT_CFG4[5]
LCD1_DATA22	Input	BOOT_CFG4[6]
LCD1_DATA23	Input	BOOT_CFG4[7]

The input pins provided are sampled at boot, and can be used to override corresponding eFUSE values, depending on the setting of the BT_FUSE_SEL fuse.

8.3.3 Device Configuration Data (DCD)

DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various on-chip peripherals. See Device Configuration Data (DCD) for more details on Device Configuration Data.

8.4 Device Initialization

This section describes the details on the ROM and provides initialization details.

This includes details on:

- The ROM Memory Map
- The RAM Memory Map
- On-chip blocks that the ROM should make use of or change POR register default values
- Clock initialization
- Enabling the MMU/L2 cache
- Exception handling and interrupt handling

8.4.1 Internal ROM /RAM memory map

The following figure shows the iROM memory map.

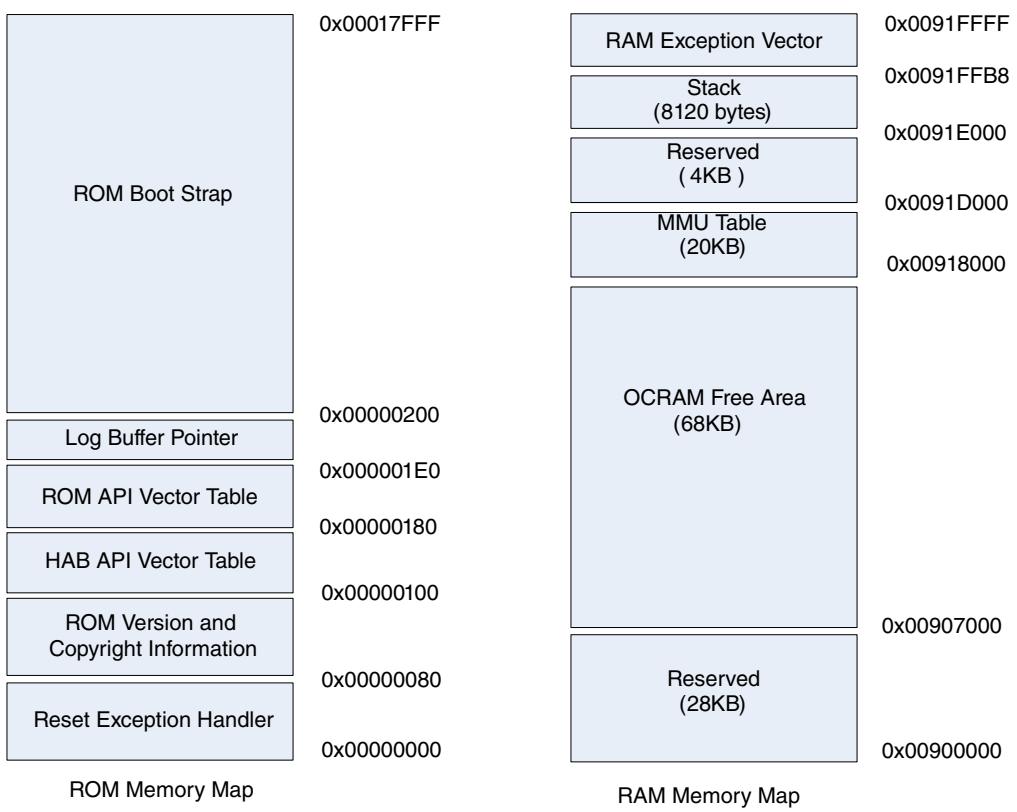


Figure 8-3. Internal ROM and RAM memory map

NOTE

The entire OCRAM region can be used freely post boot.

8.4.2 Boot Block Activation

The boot ROM affects a number of different hardware blocks which are activated and play a vital role in the boot flow.

The ROM configures and uses the following blocks (listed in alphabetical order) during the boot process. Note that the blocks actually used depend on the boot mode and boot device selection:

- APBH - DMA engine to drive the GPMI module
- BCH - 40-bit error correction hardware engine with AXI bus master and private connection to GPMI
- CCM - Clock Control Module
- ECSPI - Enhanced Configurable Serial Peripheral Interface
- EIM - External Interface Module. Used for NOR and OneNAND devices
- GPMI - NAND controller pin interface
- OCOTP_CTRL - On-Chip OTP Controller. The OCOTP contains the eFUSEs.
- IOMUXC - I/O Multiplexer Control allows GPIO use to override eFUSE boot settings
- IOMUXC GPR - I/O Multiplexer Control General Purpose registers
- CAAM - Cryptographic Acceleration and Assurance Module
- QSPI - QuadSPI Flash
- SNVS - Secure Non-Volatile Storage
- SRC - System Reset Controller
- UART - Universal Asynchronous Receiver/Transmitter Controller
- USB - Used for serial download of a boot device provisioning program
- USDHC - Ultra Secure Digital Host Controller
- WDOG-1 - Watchdog Timer

8.4.3 Clocks at Boot Time

The table below shows the various clocks and their sources used by ROM.

Table 8-4. Normal Frequency Clocks Configuration

Clock	CCM signal	Source	Frequency(MHz) BT_FREQ=0	Frequency(MHz) BT_FREQ=1
ARM PLL	pll1_sw_clk		396	396

Table continues on the next page...

Table 8-4. Normal Frequency Clocks Configuration (continued)

Clock	CCM signal	Source	Frequency(MHz) BT_FREQ=0	Frequency(MHz) BT_FREQ=1
System PLL	pll2_sw_clk		528	528
USB PLL	pll3_sw_clk		480	480
AHB	ahb_clk_root	528 MHz PLL/PFD352	132	88
IPG	ipg_clk_root	528 MHz PLL/PFD352	66	44

Following reset, each ARM core has access to all peripherals. The ROM code will disable the clocks listed in the following table, except for the boot devices listed in the second column.

Table 8-5. List Of Disabled Clocks

Clock Name	Enabled For Boot Device
CCGR0_APBHDMA_CLK_SEL	NAND
CCGR0_ASRC_CLK_SEL	
CCGR0_CAN1_CLK_SEL	
CCGR0_CAN1_SERIAL_CLK_SEL	
CCGR0_CAN2_CLK_SEL	
CCGR0_CAN2_SERIAL_CLK_SEL	
CCGR0_UART2_CLK_SEL	UART2
CCGR1_ECSPI1_CLK_SEL	ECSPI1
CCGR1_ECSPI2_CLK_SEL	ECSPI2
CCGR1_ECSPI3_CLK_SEL	ECSPI3
CCGR1_ECSPI4_CLK_SEL	ECSPI4
CCGR1_EPIT1_CLK_SEL	
CCGR1_EPIT2_CLK_SEL	
CCGR1_ADC1_CLK_SEL	
CCGR1_ADC2_CLK_SEL	
CCGR1_UART3_CLK_SEL	
CCGR1_UART4_CLK_SEL	
CCGR2_CSI_CLK_SEL	
CCGR3_CSI_CORE_CLK_SEL	
CCGR2_I2C1_CLK_SEL	
CCGR2_I2C2_CLK_SEL	
CCGR2_I2C3_CLK_SEL	
CCGR2_LCD_CLK_SEL	
CCGR2_PXP_CLK_SEL	
CCGR3_ENET_CLK_SEL	
CCGR3_LCDIF_PIX_CLK_SEL	
CCGR3_QSPI1_CLK_SEL	QSPI1

Table continues on the next page...

Table 8-5. List Of Disabled Clocks (continued)

Clock Name	Enabled For Boot Device
CCGR3_UART5_CLK_SEL	
CCGR3_UART6_CLK_SEL	
CCGR4_PWM1_CLK_SEL	
CCGR4_PWM2_CLK_SEL	
CCGR4_PWM3_CLK_SEL	
CCGR4_PWM4_CLK_SEL	
CCGR4_RAWNAND_U_BCH_INPUT_A_CLK_SEL	NAND
CCGR4_RAWNAND_U_GPMI_BCH_INPUT_BCG_CLK_SEL	NAND
CCGR4_RAWNAND_U_GPMI_BCH_INPUT_GPMI_CLK_SEL	NAND
CCGR4_RAWNAND_U_GPMI_INPUT_APB_CLK_SEL	NAND
CCGR4_TSC_CLK_SEL	
CCGR5_SDMA_CLK_SEL	
CCGR5_SPDIF_CLK_SEL	
CCGR5_SPBA_CLK_SEL	
CCGR5_UART1_CLK_SEL	
CCGR5_UART7_CLK_SEL	
CCGR5_KPP_CLK_SEL	
CCGR5_SAI1_CLK_SEL	
CCGR5_SAI2_CLK_SEL	
CCGR5_SAI3_CLK_SEL	
CCGR6_USBOH3_CLK_SEL	USB
CCGR6_USDHC1_CLK_SEL	USDHC1
CCGR6_USDHC2_CLK_SEL	USDHC2
CCGR6_EMI_SLOW_CLK_SEL	NOR, OneNAND
CCGR6_PWM8_CLK_SEL	
CCGR6_SIM1_CLK_SEL	
CCGR6_SIM2_CLK_SEL	
CCGR6_I2C4_SERIAL_CLK_SEL	
CCGR6_PWM5_CLK_SEL	
CCGR6_PWM6_CLK_SEL	
CCGR6_PWM7_CLK_SEL	
CCGR6_UART8_CLK_SEL	

8.4.4 Enabling MMU and Caches

The boot ROM includes a feature of enabling the Memory Management Unit (MMU) and caches to improve boot speed.

L1 instruction cache is enabled at the start of image download. L1 data cache, L2 cache and MMU are enabled during image authentication. Once HAB authentication completes the ROM disables the L1 data cache, L2 cache and MMU.

L1 Instruction cache, L1 data cahce, L2 cache and MMU is controlled by eFuse. By default these features are enabled.

Enabling the MMU when booting non-securely with SEC_CONFIG=Open, and setting the CSF pointer in the Image Vector Table to NULL, has no impact on the boot performance. With this configuration it is recommended to blow BT_MMU_DISABLE fuse.

8.4.5 Exception Handling

The exception vectors located at the start of ROM are used to map all the ARM exceptions (except the reset exception) to a duplicate exception vector table in internal RAM.

During the boot phase of CPU0, the RAM vectors point to the serial downloader in ROM.

After boot the program image can overwrite the vectors as required. The code shown below is used to map the ROM exception vector table to the duplicate one in RAM.

Mapping ROM Exception Vector Table

```
;; Define linker area for ROM exception vector table
AREA IROM_VECTORS, CODE, READONLY
LDR     PC, Reset_Addr
LDR     PC, Undefined_Addr
LDR     PC, SWI_Addr
LDR     PC, Prefetch_Addr
LDR     PC, Abort_Addr
NOP                          ; Reserved vector
LDR     PC, IRQ_Addr
LDR     PC, FIQ_Addr

;; Define exception vector table
Reset_Addr      DCD     start_address
Undefined_Addr  DCD     iRAM_Undefined_Handler
SWI_Addr        DCD     iRAM_SWI_Handler
Prefetch_Addr   DCD     iRAM_Prefetch_Handler
Abort_Addr      DCD     iRAM_Abort_Handler
                DCD     0               ; Reserved vector
IRQ_Addr        DCD     iRAM_IRQ_Handler
FIQ_Addr        DCD     iRAM_FIQ_Handler

start_address DCD start ;reset handler vector
```

8.4.6 Interrupt Handling During Boot

No special interrupt handling routines are required during the boot process. Interrupts are disabled during boot ROM execution and may be enabled in a later boot stage.

8.4.7 Persistent Bits

Some modes of boot ROM require registers that keep their values after warm reset. SRC General Purpose registers are used for this purpose.

See the table below for persistent bits list and description.

Table 8-6. Persistent Bits

Bit Name	Bit Location	Description
PERSIST_SECONDARY_BOOT	SRC_GPR10[30]	This bit identifies which image must be used - primary and secondary. Used only for boot modes that support redundant boot.
PERSIST_BLOCK_REWRITE	SRC_GPR10[29]	This bit is used as warning. It identifies that there are errors in NAND blocks that hold the application image. See NAND Flash for more details.
PERSISTENT_ENTRY0[31:0]	SRC_GPR1[31:0]	Holds entry function for CPU0 for waking-up from low power mode.
PERSISTENT_ARG0[31:0]	SRC_GPR2[31:0]	Holds argument of entry function for CPU0 for waking-up from low power mode.
PERSIST_OVERRIDE_SBMR1	SRC_GPR10[28]	This bit instructs the ROM code to use the SRC_GPR9 register as if it is SBMR1. This allows software to override the fuse bits and boot from an alternative boot source. This feature is vaild for a WDOG time-out reset only, and can be disabled if fuse DIR_BT_DIS had been blown.
PERSIST_SBMR1_VALUE	SRC_GPR9	The value to override SBMR1.

8.5 Boot Devices (Internal Boot)

The Chip supports the following boot Flash devices:

- NOR Flash with External Interface Module (EIM), located on CS0, 16-bit bus width
- OneNAND Flash with EIM interface, located on CS0, 16-bits bus width
- Raw NAND (MLC and SLC), and Toggle-mode NAND flash through GPMI-2 interface. Page sizes of 2 Kbyte, 4 Kbyte and 8 Kbyte. Bus widths of 8-bit with 2 through 40-bit BCH Hardware ECC (Error Correction) are supported.
- Quad SPI Flash

- SD/MMC/eSD/SDXC/eMMC4.4 via USDHC interface, supporting high capacity cards
- EEPROM boot via SPI (serial flash)

The selection of external boot device type is controlled by BOOT_CFG1[7:4] eFUSEs. See the table below for more details.

Table 8-7. Boot Device Selection

BOOT_CFG1[7:4]	Boot Device
0000	NOR/OneNAND (EIM)
0001	QSPI
0011	Serial ROM (SPI)
010x	SD/eSD/SDXC
011x	MMC/eMMC
1xxx	Raw NAND

8.5.1 NOR Flash/OneNAND using EIM Interface

The External Interface Module (EIM) works in the asynchronous mode, and supports either muxed, Address/Data, or non-muxed schemes based on fuse settings.

Table 8-8. EIM Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG1[7:4]	OEM	Boot Device Selection	Yes	0000	0000 - Boot from EIM Interface
BOOT_CFG1[3]	OEM	NOR/OneNAND Selection	Yes	0	0 - NOR 1 - OneNAND
BOOT_CFG2[7:6]	OEM	Muxing Scheme	Yes	00	00 - Muxed, 16-bit data (low half) interface 01 - Reserved 10 - Not muxed, 16-bit data (low half) interface 11 - Reserved
BOOT_CFG2[5:4]	OEM	OneNAND Page Size	Yes	00	00 - 1K 01 - 2K 10 - 4K 11 - Reserved

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See GPIO Boot Overrides for corresponding GPIO pin.

8.5.1.1 NOR Flash Boot Operation

Booting from the NOR Flash is supported via EIM interface. The ROM reads Image Vector Table and Boot Data structures to determine if the image can be executed directly from EIM address space or should be copied to other memory.

The start field of Boot Data Structure specifies the final location of the image (see Image Vector Table and Boot Data).

8.5.1.2 OneNAND Flash Boot Operation

At system power-up, the OneNAND device automatically copies an Initial Load Region of 1 Kbyte from the start of the flash array (sector 0 and sector 1, page 0, block 0) to its Boot RAM (OneNAND's internal RAM).

NOTE

The OneNAND boot RAM memory containing the Initial 1K Load Region must contain the IVT, DCD and the Boot Data structures.

Next, the ROM processes the DCD and then proceeds to copy the Program Image contents to the application destination pointer (located in the start entry of Boot Data (see Image Vector Table and Boot Data). The ROM determines the size of the Program Image by the length specified by size entry in Boot Data structure (see Image Vector Table and Boot Data). A failure loading data from the OneNAND device for any reason forces the Chip to enter the Serial Downloader, otherwise the booting from the OneNAND device continues.

The figure below illustrates the layout of the Program Image on a OneNAND boot device.

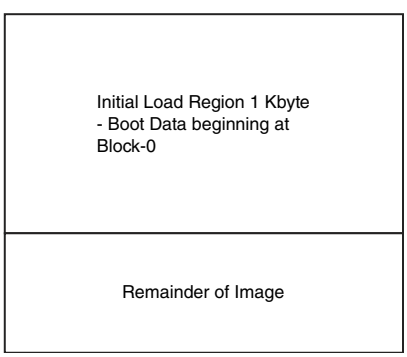


Figure 8-4. Program Image Layout on a OneNAND Flash Device

Prior to accessing the OneNAND device, the Chip waits approximately 500 µs after Power On Reset. This delay is required for the OneNAND device to become ready. After this initial 500 µs delay it can take an addition 70 µs for the OneNAND device to load the Initial Load Region of 1 Kbyte into its boot RAM. The Chip polls the OneNAND device Interrupt Status Register to confirm that the first 1 Kbytes has been loaded to the OneNAND boot RAM before continuing with the boot flow.

8.5.1.3 IOMUX Configuration for EIM Devices

The EIM interface uses dedicated contacts on the IC.

The contacts assigned to the data signals used by EIM are shown in the table below.

Table 8-9. EIM IOMUX Pin Configuration

Signal	A/D16 (Muxed, 16-bit data interface)	A+D (Not muxed, 16-bit data interface)
DATA0	CSI_DATA00.alt4	LCD_DATA08.alt4
DATA1	CSI_DATA01.alt4	LCD_DATA09.alt4
DATA2	CSI_DATA02.alt4	LCD_DATA10.alt4
DATA3	CSI_DATA03.alt4	LCD_DATA11.alt4
DATA4	CSI_DATA04.alt4	LCD_DATA12.alt4
DATA5	CSI_DATA05.alt4	LCD_DATA13.alt4
DATA6	CSI_DATA06.alt4	LCD_DATA14.alt4
DATA7	CSI_DATA07.alt4	LCD_DATA15.alt4
DATA8	NAND_DATA00.alt4	LCD_DATA16.alt4
DATA9	NAND_DATA01.alt4	LCD_DATA17.alt4
DATA10	NAND_DATA02.alt4	LCD_DATA18.alt4
DATA11	NAND_DATA03.alt4	LCD_DATA19.alt4
DATA12	NAND_DATA04.alt4	LCD_DATA20.alt4
DATA13	NAND_DATA05.alt4	LCD_DATA21.alt4
DATA14	NAND_DATA06.alt4	LCD_DATA22.alt4
DATA15	NAND_DATA07.alt4	LCD_DATA23.alt4
ADDR0	CSI_DATA00.alt4	
ADDR1	CSI_DATA01.alt4	
ADDR2	CSI_DATA02.alt4	
ADDR3	CSI_DATA03.alt4	
ADDR4	CSI_DATA04.alt4	
ADDR5	CSI_DATA05.alt4	
ADDR6	CSI_DATA06.alt4	
ADDR7	CSI_DATA07.alt4	
ADDR8	NAND_DATA00.alt4	
ADDR9	NAND_DATA01.alt4	

Table continues on the next page...

Table 8-9. EIM IOMUX Pin Configuration (continued)

Signal	A/D16 (Muxed, 16-bit data interface)	A+D (Not muxed, 16-bit data interface)
ADDR10	NAND_DATA02.alt4	
ADDR11	NAND_DATA03.alt4	
ADDR12	NAND_DATA04.alt4	
ADDR13	NAND_DATA05.alt4	
ADDR14	NAND_DATA06.alt4	
ADDR15	NAND_DATA07.alt4	
ADDR16	NAND_CLE.alt4	
ADDR17	NAND_ALE.alt4	
ADDR18	NAND_CE1_B.alt4	
ADDR19	SD1_CMD.alt4	
ADDR20	SD1_CLK.alt4	
ADDR21	SD1_DATA0.alt4	
ADDR22	SD1_DATA1.alt4	
ADDR23	SD1_DATA2.alt4	
ADDR24	SD1_DATA3.alt4	
ADDR25	ENET2_RXER.alt4	
ADDR26	ENET2_CRS_DV.alt4	

8.5.2 NAND Flash

The boot ROM supports a number of MLC/SLC NAND Flash devices from different vendors and LBA NAND Flash devices. The Error Correction and Control (ECC) subblock (BCH) is used to detect the errors.

8.5.2.1 NAND eFUSE Configuration

The boot ROM determines the configuration of external NAND flash by parameters, either provided by eFUSE, or sampled on GPIO pins, during boot. See Table 8-10 for parameters details.

NOTE

BOOT_CFGx sampled on GPIO pins depends on BT_FUSE_SEL setting. See Boot Fusemap for details.

NOTE

For BOOT_CFG[3:2], although ROM always boots from CS0_B, for multiple chip selects, such as some NAND chips which consist of multi CS. This fuse must burned correctly. The number of devices means the number of chip selects.

Table 8-10. NAND Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG1[7]	OEM	Boot Device Selection	Yes	0	1 - Boot from NAND Interface
BOOT_CFG1[6]	OEM	BT_TOGGLEMODE	Yes	0	0 - raw NAND 1 - toggle mode NAND
BOOT_CFG1[5:4]	OEM	Pages In Block	Yes	0	00 - 128 01- 64 10- 32 11- 256
BOOT_CFG1[3:2]	OEM	Number of devices	Yes	00	00 - 1 device 01- 2 device 10- 4 device 11- Reserved
BOOT_CFG1[1:0]	OEM	Row Address Cycles	Yes	00	00 - 3 01 - 2 10 - 4 11 - 5
BOOT_CFG2[7:5]	OEM	Toggle Mode 33 MHz Preamble Delay, Read Latency	Yes	000	000 - 16 GPMICLK cycles 001 - 1 GPMICLK cycles 010 - 2 GPMICLK cycles 011 - 3 GPMICLK cycles 100 - 4 GPMICLK cycles 101 - 5 GPMICLK cycles 110 - 6 GPMICLK cycles 111 - 7 GPMICLK cycles
BOOT_CFG2[4:3]	OEM	Boot Search Count	Yes	00	00 - 2 01 - 2 10 - 4 11 - 8
BOOT_CFG2[2]	OEM	Boot Frequencies (ARM/DDR)	Yes	0	0 - 500/400 MHz 1 - 250/200 MHz
BOOT_CFG2[1]	OEM	Reset time	Yes	0	0 - 12ms 1 - 22ms (LBA NAND)
0x470[0]	OEM	Override Pad Settings	Yes	0	Override NAND Pad Settings

Table continues on the next page...

Table 8-10. NAND Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
					0 - Use default values 1 - Use PAD_SETTINGS value
0x6D0[5:0]	OEM	PAD_SETTINGS[5:0]	Yes	0	NAND Pad Settings Value
0x6D0[11:8]	OEM	READ_RETRY_SEQ_ID[3:0]	Yes	0000	0000 - Don't use ROM embedded read-retry sequence 0001 - use Micron 20 nm read-retry sequence 0010 - use Toshiba A19 nm read-retry sequence 0011 - use Toshiba 19 nm read-retry sequence 0100 - use SanDisk 19 nm read-retry sequence 0101 - use SanDisk 1ynm read-retry sequence 0110 - use Hynix 20 nm A die read retry sequence 0111 - use Hynix 26 nm read retry sequence 1000 - use Hynix 20 nm B die read retry sequence 1001 - use Hynix 20 nm C die read retry sequence 1010 to 1111 - Reserved

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 3 for corresponding GPIO pin.

8.5.2.2 NAND Flash Boot Flow and Boot Control Blocks (BCB)

There are two BCB data structures:

- FCB
- DBBT

As part of the NAND media initialization, the ROM driver uses safe NAND timings to search for a Firmware Configuration Block (FCB) that contains the optimum NAND timings, page address of Discovered Bad Block Table (DBBT) Search Area and start page address of primary and secondary firmware.

The hardware ECC level to use is embedded inside FCB block. The FCB data structure is also protected using ECC. Driver reads raw 2112 bytes of first sector and runs through software ECC engine that determines whether FCB data is valid or not.

If the FCB is found, the optimum NAND timings are loaded for further reads. If the ECC fails, or the fingerprints do not match, the Block Search state machine increments page number to Search Stride number of pages to read for the next BCB until SearchCount pages have been read.

If search fails to find a valid FCB, the NAND driver responds with an error and the boot ROM enters into serial download mode.

The FCB contains the page address of DBBT Search Area, and the page address for primary and secondary boot images. DBBT is searched in DBBT Search Area just like how FCB is searched. After the FCB is read, the DBBT is loaded, and the primary or secondary boot image is loaded using starting page address from FCB.

Figure 8-5 shows the state diagram of FCB search.

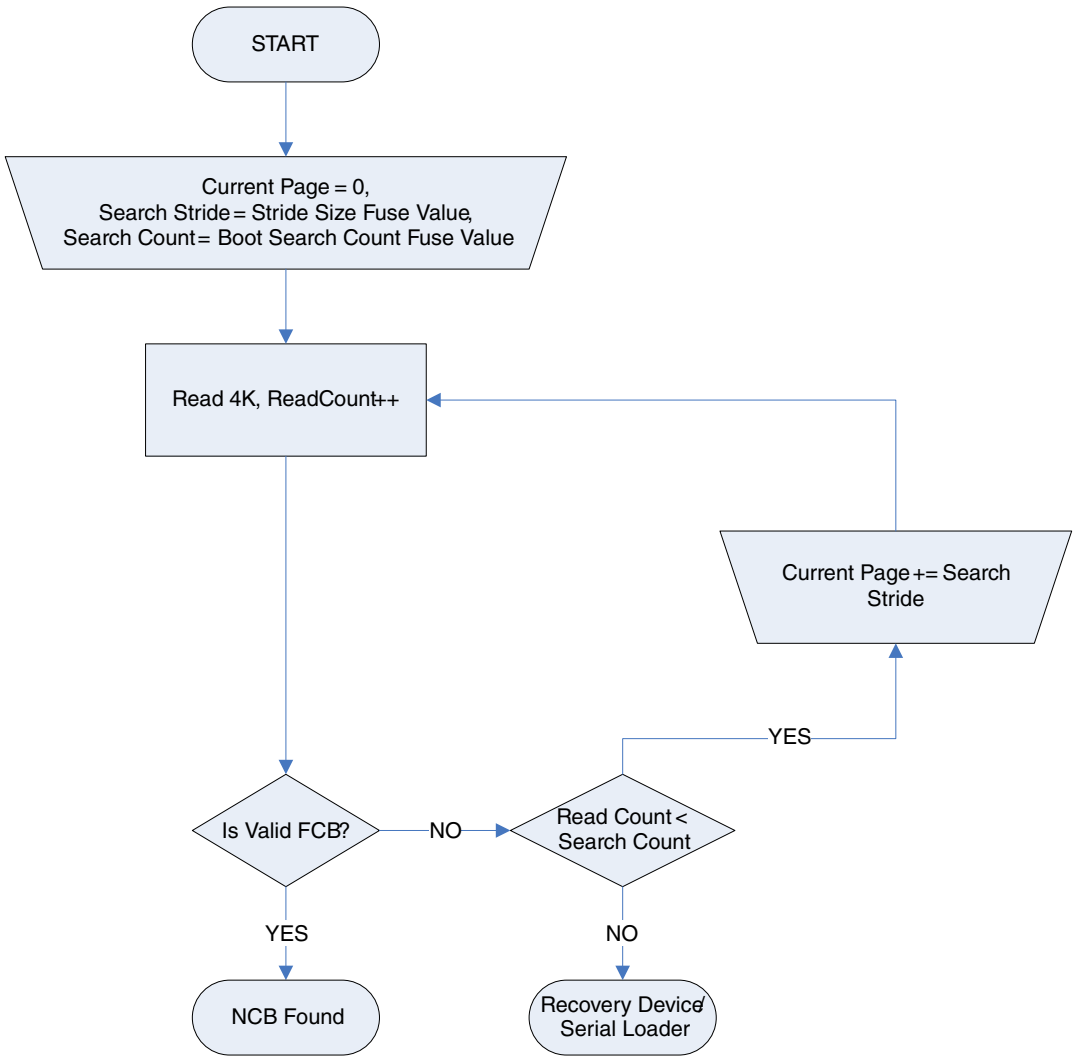


Figure 8-5. FCB Search Flow

Once FCB is found, the boot ROM searches for the Discovered Bad Blocks Table (DBBT). If DBBT Search Area is 0 in FCB, then ROM assumes that there are no bad blocks on NAND device boot area. See Figure 8-6 for the DBBT search flow.

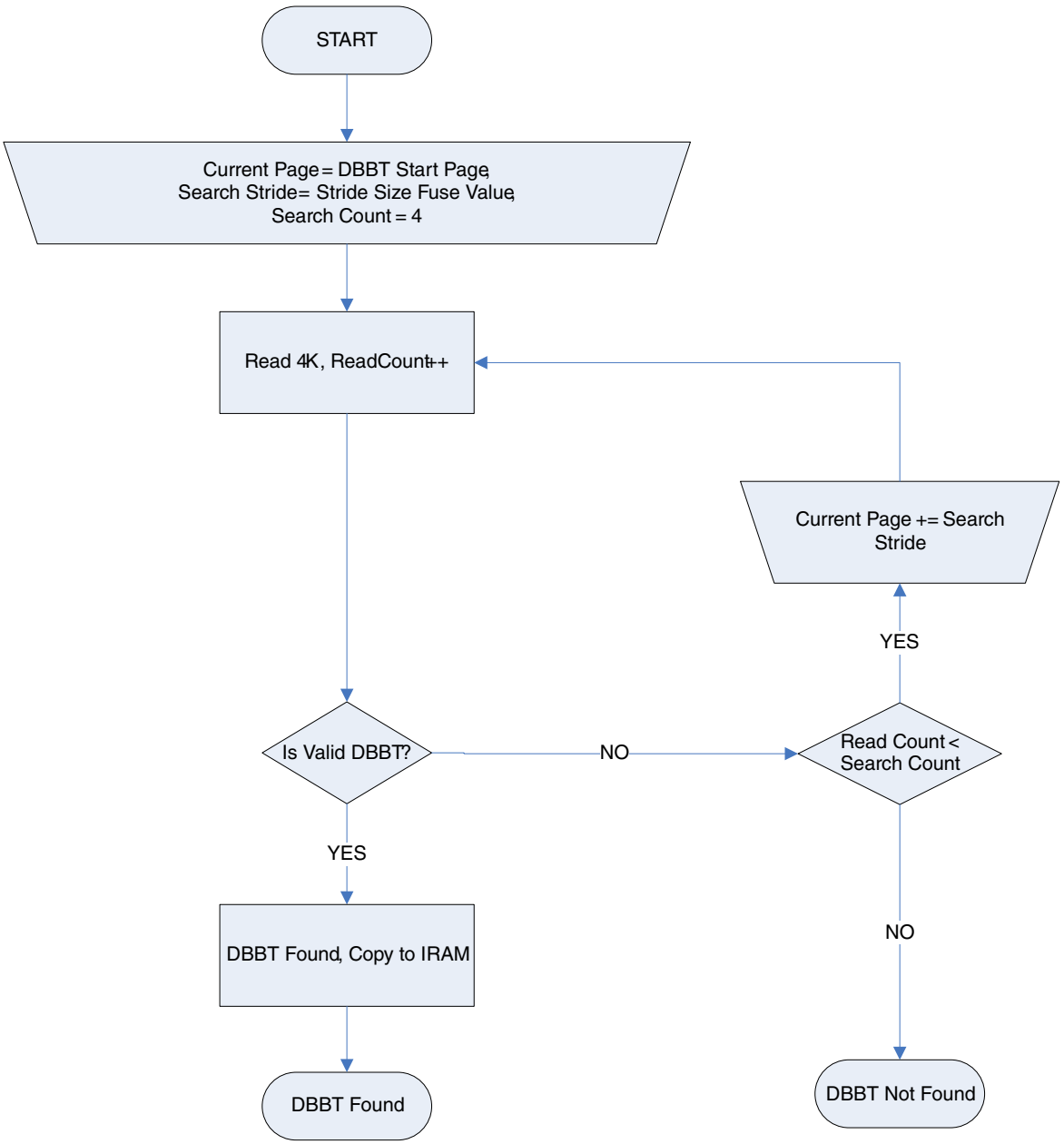


Figure 8-6. DBBT Search Flow

The BCB search and load function also monitors the ECC correction threshold and sets the PERSIST_BLOCK_REWRITE persistent bit if the threshold exceeds the maximum ECC correction ability.

If during primary image read there is a page with a number of errors higher than ECC can correct, the boot ROM will turn on PERSIST_SECONDARY_BOOT bit and perform SW reset (After SW reset, secondary image is used).

If during secondary image read there is a page with number of errors higher than ECC can correct, the boot ROM goes to serial loader.

8.5.2.3 Firmware Configuration Block

The FCB is the first sector in the first good block. The FCB should be present at each search stride of the search area.

The search area contains copies of the FCB at each stride distance, so in case the first NAND block becomes corrupted, the ROM will find its copy in the next NAND block. The search area should span over at least two NAND blocks. The location information for DBBT search area, FW1, and FW2 are all specified in the FCB. Table 8-11 shows the Flash Control Block Structure.

Table 8-11. Flash Control Block Structure

Name	Start Byte	Size in Bytes	Description
Reserved	0	4	Reserved for Fingerprint #1(Checksum)
FingerPrint	4	4	32 bit word with a value of 0x4E434220, in ascii "FCB"
Version	8	4	32-bit version number; this version of FCB is 0x00000001
m_NANDTiming	12	8	8 bytes of data for 8 NAND Timing Parameters from NAND datasheet. The 8 parameters are: m_NandTiming[0]=data_setup, m_NandTiming[1]=data_hold, m_NandTiming[2]=address_setup, m_NandTiming[3]=dsample_time, m_NandTiming[4]=nand_timing_state, m_NandTiming[5]=REA, m_NandTiming[6]=RLOH, m_NandTiming[7]=RHOH. ROM only uses first 4 parameters but FCB provides space for other 4 parameters to be used by boot-loader or other applications.
PageDataSize	20	4	Number of bytes of data in a page. Typically, this is 2048 bytes for 2112 bytes page size or 4096 bytes for 4314/4224 bytes page size or 8192 for 8568 bytes page size
TotalPageSize	24	4	Total number of bytes in page. Typically, 2112 for 2 KB page or 4224 or 4314 for 4 KB page or 8568 for 8 KB page.
SectorsPerBlock	28	4	Number of pages per block. Typically 64 or 128 or depending on NAND device type.

Table continues on the next page...

Table 8-11. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
NumberOfNANDs	32	4	Not used by ROM
TotalInternalDie	36	4	Not used by ROM
CellType	40	4	Not used by ROM
EccBlockNEccType	44	4	Value from 0 to 20 used to set BCH Error Corrrection level 0, 2, 4, .. or 40 for Block BN of ECC page, used in configuring BCH40 page layout registers
EccBlock0Size	48	4	Size of block B0, used in configuring BCH40 page layout registers
EccBlockNSize	52	4	Size of block BN, used in configuring BCH40 page layout registers
EccBlock0EccType	56	4	Value from 0 to 20 used to set BCH Error Corrrection level 0, 2, 4, .. or 40 for Block BN of ECC page, used in configuring BCH40 page layout registers
MetadataBytes	60	4	Size of metadata bytes used in configuring BCH40 page layout registers
NumEccBlocksPerPage	64	4	Number of ECC blocks BN not including B0. This value is used in configuring BCH40 page layout registers
EccBlockNEccLevelSDK	68	4	Not used by ROM
EccBlock0SizeSDK	72	4	Not used by ROM
EccBlockNSizeSDK	76	4	Not used by ROM
EccBlock0EccLevelSDK	80	4	Not used by ROM
NumEccBlocksPerPageSDK	84	4	Not used by ROM
MetadataBytesSDK	88	4	Not used by ROM
EraseThreshold	92	4	Not used by ROM
Firmware1_startingPage	104	4	Page number address where first copy of bootable firmware is located
Firmware2_startingPage	108	4	Page number address where second copy of bootable firmware is located
PagesInFirmware1	112	4	Size of first copy of firmware in pages
PagesInFirmware2	116	4	Size of second copy of firmware in pages
DBBTSearchAreaStartAddress	120	4	Page address for bad block table search area
BadBlockMarkerByte	124	4	This is an input offset in BCH page for ROM to swap with first byte of metadata after reading a page using BCH40. ROM supports restoration of manufacturer marked bad block markers in the page and this offset is the bad block marker offset location
BadBlockMarkerStartBit	128	4	This is an input bit offset in BadBlockMarkerByte for ROM to use when swapping 8 bits with fiirst byte of metadata.
BBMarkerPhysicalOffset	132	4	This is the offset where manufacturer leaves bad block marker on a page

Table continues on the next page...

Table 8-11. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
BCHType	136	4	0 for BCH20 and 1 for BCH40. The Chip is backward compatible to BCH20 and this field tell ROM to use BCH20 or BCH40 block
TMTiming2_ReadLatency	140	4	Toggle mode NAND timing parameter read latency, ROM use this value to configure timing2 register of GPMI
TMTiming2_PreambleDelay	144	4	Toggle mode NAND timing parameter Preamble Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_CEDelay	148	4	Toggle mode NAND timing parameter CE Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_PostambleDelay	152	4	Toggle mode NAND timing parameter Postamble Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_CmdAddPause	156	4	Toggle mode NAND timing parameter Cmd Add Pause. ROM use this value to configure timing2 register of GPMI
TMTiming2_DataPause	160	4	Toggle mode NAND timing parameter Data Pause. ROM use this value to configure timing2 register of GPMI
TMSpeed	164	4	This is the toggle mode speed for ROM to configure gpmi clock. 0 for 33 MHz, 1 for 40 MHz and 2 for 66 MHz
TMTiming1_BusyTimeout	168	4	Toggle mode NAND timing parameter Busy Timeout. ROM use this value to configure timing1 register of GPMI
DISBBM	172	4	If 0 ROM will swap BadBlockMarkerByte with metadata[0] after reading a page using BCH40. If the value set is 1 then ROM will not do swapping
BBMark_spare_offset	176	4	The offset in mata data place which stores the data in Bad block marker place.
Onfi_sync_enable	180	4	Enable the Onfi nand sync mode support
Onfi_sync_speed	184	4	Speed for onfi nand sync mode: 0 - 24 MHZ, 1 - 33 MHZ, 2 - 40 MHZ, 3 - 50 MHZ, 4 - 66 MHZ, 5 - 80 MHZ, 6 - 100 MHZ, 7 - 133 MHZ, 8 - 160 MHZ, 9 - 200 MHZ
Onfi_syncNANDData	188	28	parameters for onfi nand sync mode timing. They are read latency, ce_delay, preamble_delay, postamble_delay, cmdadd_pause, data_pause, busy_timeout
DISBB_Search	216	4	Disable the badblock search function when reading the firmware, only using DBBT.
Reserved	220	64	Reserved for future use

8.5.2.4 Discovered Bad Block Table (DBBT)

See the table below for DBBT format.

Table 8-12. DBBT Structure

Name	Start Byte	Size in Bytes	Description
reserved	0	4	-
FingerPrint	4	4	32-bit word with a value of 0x44424254,in ascii "DBBT"
Version	8	4	32-bit version number; this version of DBBT is 0x00000001
reserved	12	4	-
DBBT_NUM_OF_PAGES	16	4	Size of DBBT in pages
reserved	20	4*PageSize-20	-
reserved	4*PageSize	4	-
Number of Entries	4*PageSize + 4	4	Number of bad blocks
Bad Block Number	4*PageSize + 8	4	First bad block number
Bad Block Number	4*PageSize + 12	4	Second bad block number
...-	-	-	...next bad block number
...-	-	-	...-
Last bad block number	-	-	last bad block number

8.5.2.5 Bad Block Handling in the ROM

During firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block.

If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block checked.

If Bad Block table start page is null, check the manufactory made Bad Block marker. The location of Bad Block maker is at the first 3 or last 3 pages in every block of the NAND flash. NAND manufacturers normally use one byte in the spare area of certain pages within a block to mark a block is bad or not. 0xFF means good block, non FF means bad block.

In order to preserve the BI (bad block information), flash updater or gang programmer applications need to swap Bad Block Information (BI) data to byte 0 of metadata area for every page before programming NAND flash. ROM when loading firmware copies back the value at metadata[0] to BI offset in page data. The Figure 8-7 shows how the factory bad block marker is preserved.

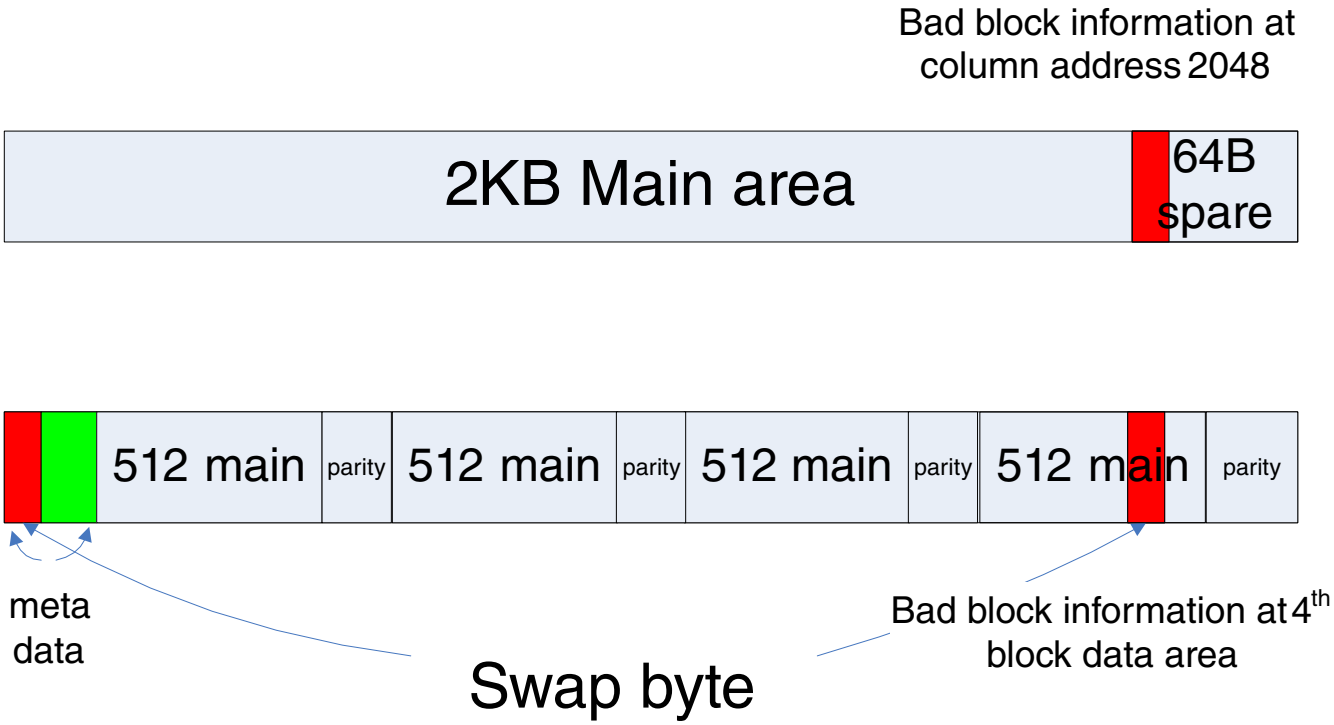


Figure 8-7. Factory Bad Block Marker Preservation

In the FCB structure, there are two elements m_u32BadBlockMarkerByte and m_u32BadBlockMarkerStartBit to indicate the byte and bit place in the page data, that manufacturer marked the bad block marker.

8.5.2.6 Read Retry Handling in the ROM

Read retry is used to recover from NAND bit errors beyond the ECC correction threshold. If reading a page failed and the read_retry_enable field in FCB is set as 1, ROM will issue read retry command sequence to shift the read level before reading the page again. If the previous reading failed, ROM keep shifting the read level until reading succeeds or all levels had been tried. The state diagram of read retry is shown in the figure below.

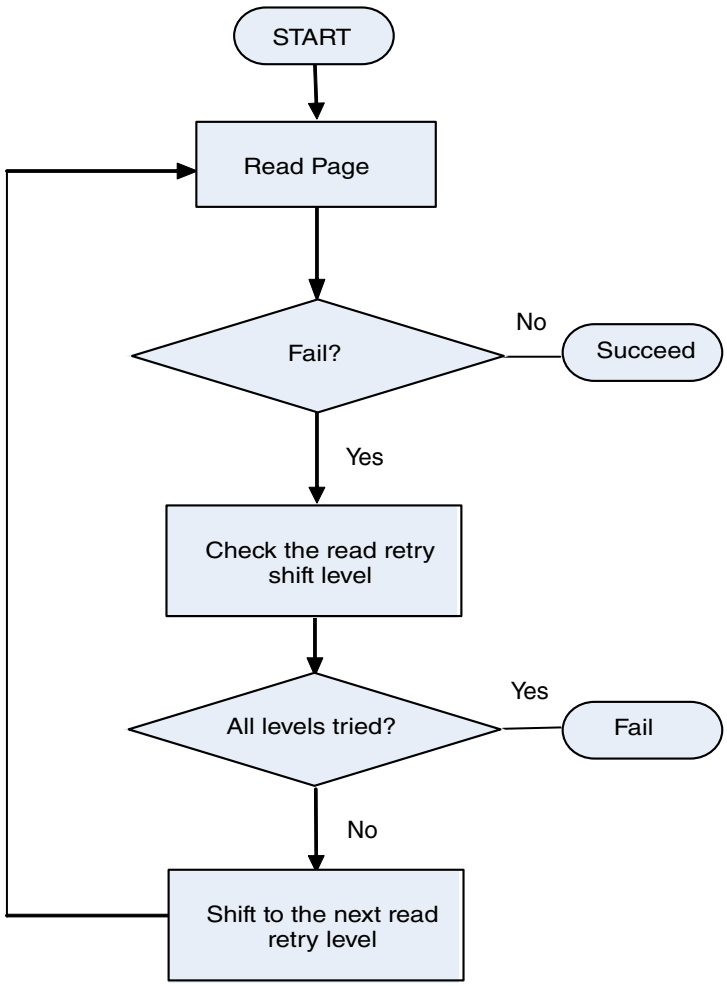


Figure 8-8. Read Retry Flow

Different vendors and different process may have different read retry sequences. At present, ROM supports five (5) read retry sequences. Blowing the fuse READ_RETRY_SEQ_ID[3:0] can select to use which sequence.

The detailed read retry sequences are listed in the table below.

Table 8-13. Read retry sequences

Vendor	Process	READ_RETRY_SEQ_ID[3:0]	Comment
Micron	20nm	2'b0001	The detail of this RR sequence is documented in "64Gb, 128Gb, 256Gb, 512Gb Asynchronous/ Synchronous NAND Features(Release: 4/20/12)", please contact Micron for this doc.

Table continues on the next page...

Table 8-13. Read retry sequences (continued)

Vendor	Process	READ_RETRY_SEQ_ID[3:0]	Comment
Toshiba	A19nm	2'b0010	The detail of this RR sequence is documented in "TOSHIBA Technical Information A19nm MLC NAND Retry Read Sequence", please contact Toshiba for this doc.
	19nm	2'b0011	The detail of this RR sequence is documented in "TOSHIBA Technical Information 19nm MLC NAND Read Retry Sequence Rev1.6", please contact Toshiba for this doc.
SanDisk	19nm	2'b0100	The detail of this RR sequence is documented in "App Note 023 (v1.0) 19nm eX2 ABL Dynamic Read Sequence & Parameter Table", please contact SanDisk for this doc.
	1ynm	2'b0101	The detail of this RR sequence is documented in "Application Note 1y_023 19nm eX2 ABL Dynamic Read Sequence & Parameter Table", please contact SanDisk for this doc.
Hynix	26nm	2'b0111	The detail of this RR sequence is documented in " 26nm 32Gb MLC RAWNAND", please contact Hynix for this doc.
	20nm A Die	2'b0110	The detail of this RR sequence is documented in " 20nm 32Gb MLC C-die Application Note", please contact Hynix for this doc.
	20nm B Die	2'b1000	The detail of this RR sequence is documented in " 20nm 32Gb MLC C-die Application Note", please contact Hynix for this doc.
	20nm C Die	2'b1001	The detail of this RR sequence is documented in " 20nm 32Gb MLC C-die Application Note", please contact Hynix for this doc.

8.5.2.7 Toggle Mode DDR NAND Boot

If BT_TOGGLEMODE efuse is blown then ROM does the following to boot from Samsung's toggle mode DDR NAND.

8.5.2.7.1 GPMI and BCH Clocks Configuration

ROM sets the clock source and the dividers in CCM registers.

If BOOT_CFG1[6] is set(toggle mode), GPMI/BCH CLK source is PLL2PFD4, and running at 66 MHZ, otherwise GPMI/ BCH CLK souce is PLL3, running at 24 MHz. ROM sets default values to timing0, timing1 and timing2 gpmi registers for 24 MHz clock speed. It uses fuse BOOT_CFG to configure GPMI timing2 register parameters preamble delay and read latency, the default value for these parameters is 2 when fuses are not blown.

Default timing parameter values used by ROM for toggle-mode device:

- Timing0.ADDRESS_SETUP = 5
- Timing0.DATA_SETUP = 10
- Timing0.DATA_HOLD = 10
- Timing1.DEVICE_BUSY_TIMEOUT = 0 x 500
- Timing2.READ_LATENCY = BOOT_CFG2[7:5] if blown, otherwise 2
- Timing2.CE_DELAY = 2
- Timing2.PREAMBLE_DELAY = BOOT_CFG2[7:5] if blown, otherwise 2
- Timing2.POSTAMBLE_DELAY = 3
- Timing2.CMDADD_PAUSE = 4
- Timing2.DATA_PAUSE = 6

Default timing parameters can be overriden by TMTiming2_ReadLatency, TMTiming2_PreambleDelay, TMTiming2_CEDelay, TMTiming2_PostambleDelay, TMTiming2_CmdAddPause, TMTiming2_DataPause parameters of FCB.

8.5.2.7.2 Setup DMA for DDR Transfers

In DMA descriptors GPMI is configured to read page data at double data rate, the word length is set to 16 and transfer count to half of page size.

8.5.2.7.3 Reconfigure Timing and Speed Using Values in FCB

After reading FCB page with GPMI set to default timings and speed 33 MHz, ROM reconfigures CCM dividers to run gpmi/bch clks to desired speed specified in FCB for rest of boot process. The GPMI timing registers are also reconfigured to values specified in FCB.

The GPMI speed can be configured using FCB parameter TMSpeed:

- 0 - 24 MHZ
- 1 - 33 MHZ
- 2 - 40 MHZ
- 3 - 50 MHz
- 4 - 66 MHz
- 5 - 80 MHz
- 6 - 100 MHz
- 7 - 133 MHz
- 8 - 160 MHz
- 9 - 200 MHz

The GPMI timing0 register fields data_setup, data_hold and address_setup are set to values specified for data_setup, data_hold and address_setup in FCB member m_NANDTiming.

The GPMI timing1.DEVICE_BUSY_TIMEOUT is set to value specified in FCB member TMTiming1_BusyTimeout.

The GPMI timing2 register values are set using FCB members TMTiming2.READ_LATENCY, CE_DELAY, PREAMBLE_DELAY, POSTAMBLE_DELAY, CMDADD_PAUSE and DATA_PAUSE.

8.5.2.8 Typical NAND Page Organization

8.5.2.8.1 BCH ECC Page Organization

The first data block is called block 0 and the rest of the blocks are called block N. Separate ECC level scan be used for block 0 and block N.

The metadata bytes should be located at the beginning of a page, starting at byte 0, followed by data block 0, followed by ECC bytes for data block 0, followed by block 1 and its ECC bytes, and so on until N data blocks. The ECC level for block 0 can be different from the ECC level of rest of the blocks.

For NAND boot, with page size restrictions and data block size restricted to 512 bytes, only few combinations of ECC for block 0 and block N are possible.

The figure below shows the valid layout for 2112 byte sized page.

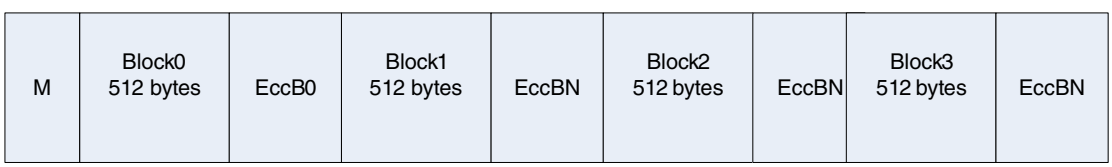


Figure 8-9. Valid Layout for 2112 bytes Sized Page

The example below is for 13 bits of parity(GF13). The number of ECC bits required for a data block is calculated using (ECC_Correction_Level * 13) bits.

In the above layout the ECC size for EccB0 and EccBN should be selected to not exceed a total page size of 2112 bytes. EccB0 and EccBN can be one of 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 bits ECC correction level. The total bytes would then be:

[M + (data_block_size x 4) + ([EccB0 + (EccBN x 3)] x 13) / 8] <= 2112;

M = metadata bytes and data_block_size is 512.

There are 4 data blocks of 512 bytes each in a page of 2k page sized NAND. The values of EccB0 and EccBN should be such that the above calculation would not result in a value greater than 2112 bytes.

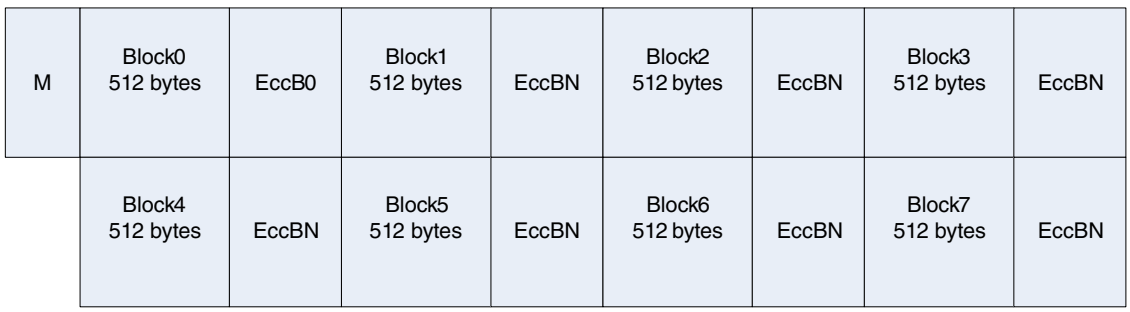


Figure 8-10. Valid Layout for 4 Kbytes Sized Page

Different NAND manufacturers have different sizes for a 4K page; 4314 bytes is typical.

[M + (data_block_size x 8) + ([EccB0 + (EccBN x 7)] x 13) / 8] <= 4314;

M= metadata bytes and data_block_size is 512.

There are 8 data blocks of 512 bytes each in a page of a 4k page sized NAND. The values of EccB0 and EccBN should be such that above calculation should not result in a value greater than the size of a page in a 4k page NAND.

8.5.2.8.2 Metadata

The number of bytes used for metadata is specified in FCB. Metadata for BCH encoded pages will be placed at the beginning of a page. ROM only cares about the first byte of metadata to swap it with bad block marker byte in page data after each page read; it is important to have at least one byte for the metadata bytes field in FCB data structure.

8.5.2.9 IOMUX Configuration for NAND

The table below shows the RawNAND IOMUX pin configuration.

Table 8-14. NAND IOMUX Pin Configuration

Signal	Pad Name
NAND.CLE	NAND_CLE.alt0
NAND.ALE	NAND_ALE.alt0
NAND.DATA00	NAND_DATA00.alt0
NAND.DATA01	NAND_DATA01.alt0
NAND.DATA02	NAND_DATA02.alt0

Table continues on the next page...

Table 8-14. NAND IOMUX Pin Configuration (continued)

Signal	Pad Name
NAND.DATA03	NAND_DATA03.alt0
NAND.DATA04	NAND_DATA04.alt0
NAND.DATA05	NAND_DATA05.alt0
NAND.DATA06	NAND_DATA06.alt0
NAND.DATA07	NAND_DATA07.alt0
NAND.RE_B	NAND_RE_B.alt0
NAND.WE_B	NAND_WE_B.alt0
NAND.CE1_B	NAND_CE1_B.alt0
NAND.CE0_B	NAND_CE0_B.alt0
NAND.DQS	NAND_DQS.alt0
NAND.READY_B	NAND_READY_B.alt0
NAND.CE2_B	CSI_MCLK.alt2
NAND.CE3_B	CSI_PIXCLK.alt2
NAND.WP_B	NAND.WP_B.alt0

8.5.3 Expansion Device

The ROM supports booting from MMC/eMMC and SD/eSD compliant devices.

8.5.3.1 Expansion Device eFUSE Configuration

SD/MMC/eSD/eMMC/SDXC boot can be performed using either USDHC ports, based on setting of the BOOT_CFG2[4:3] (Port Select) fuse or it's associated GPIO input value at boot.

All USDHC ports support eMMC4.4 and eMMC4.5 fast boot. See the table below for details.

Table 8-15. USDHC Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO	Shipped value	Settings
0x450[7:6]	OEM	Boot Device Selection	Yes	00	01 - Boot from USDHC Interface
0x450[5]	OEM	SD/MMC Selection	Yes	0	0 - SD/eSD/SDXC 1 - MMC/eMMC
0x450[4]	OEM	Fast Boot Support	Yes	0	0 - Normal Boot 1 - Fast Boot

Table continues on the next page...

Table 8-15. USDHC Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO	Shipped value	Settings
0x450[3:2]	OEM	SD/MMC Speed Mode, and eMMC Acknowledge Enabled Selection	Yes	00	MMC 0x – Normal Speed Mode 1x - High Speed Mode x0 - eMMC Fast boot acknowledge enable x1 - eMMC Fast boot acknowledge disable SD 00 – Normal/SDR12 01 – High/SDR25 10 - SDR50 11 - SDR104
0x450[1]	OEM	SD Power Cycle Enable/ eMMC Reset Enable	Yes	0	MMC 0 – No action 1 - eMMC reset enabled via SD_RST pad SD 0 - No power cycle 1 - Power cycle enabled via SD_RST pad
0x450[0]	OEM	SD Loopback Clock Source Sel(for SDR50 and SDR104 only)	Yes	0	0 - through SD pad 1 - direct
0x450[15:13]	OEM	SD MMC Bus Width Selection /SD Calibration Step	Yes	00	SD/eSD/SDXC (BOOT_CFG1[5]=0) Bus Width xx0 - 1-bit xx1 - 4-bit SD Calibration Step 00x - 1 delay cells 01x - 1 delay cells 10x - 2 delay cells 11x - 3 delay cells MMC/eMMC (BOOT_CFG1[5]=1) 000 - 1-bit 001 - 4-bit 010 - 8-bit

Table continues on the next page...

Table 8-15. USDHC Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO	Shipped value	Settings
					101 - 4-bit DDR (MMC 4.4) 110 - 8-bit DDR (MMC 4.4) Else - reserved.
0x450[12:11]	OEM	USDHC Port Selection	Yes	00	00 - USDHC-1 01 - USDHC-2 1x - Reserved
0x450[9]	OEM	USDHC1 Voltage selection	Yes	0	0 - 3.3V 1 - 1.8V
0x460[31:30]	OEM	Power Cycle Selection	Yes	00	00 - 20ms 01 - 10ms 10 - 5ms 11 - 2.5ms
0X460[29]	OEM	Power Stable Cycle Selection	Yes	0	0 - 5ms 1 - 2.5ms
0X460[24]	OEM	SD/MMC DLL Enable Selection	Yes	0	0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/Emmc
0X470[7]	OEM	DLL Override Selection	Yes	0	0 – Not Override 1 - DLL Override Mode for SD/eMMC (Override by MMC_DLL_DLY, 0x470[19:16])
0X470[6]	OEM	USDHC1 Reset Polarity Selection	Yes	0	0 - Reset active low 1 - Reset active high
0X470[5]	OEM	USDHC2 Voltage selection	Yes	0	0 - 3.3V 1 - 1.8V
0X470[0]	OEM	SD/MMC Pad Settings Override Selection	Yes	0	0 - Not Override 1 – Override (Override by PAD_SETTINGS, 0x6d0[5:0])
0X470[15]	OEM	USDHC2 Reset Polarity Selection	Yes	0	0 - Reset active low 1 - Reset active high
0X470[14]	OEM	eMMC4.4 Pre-idle Enabled Selection	Yes	0	0 - Issue pre-idle command 1 - Not issue
0x470[13]	OEM	Override HYS bit for SD/MMC pads	Yes	0	0 - Not override 1 - override HYS bit with 1
0X470[30:24]	OEM	MMC_DLL_DLY	Yes	0000000	Override number
0X6D0[5:0]	OEM	PAD_SETTINGS	Yes	000000	Override number

Boot code supports following standards.

- MMCv4.4 or less
- eMMCv4.4 or less
- SDv2.0 or less
- eSDv2.10 rev-0.9, with or without FAST_BOOT.
- SDXCv3.0

MMC/SD/eSD/SDXC/eMMC can be connected to any of the USDHC blocks and can be booted by copying 4Kbyte of data from MMC/SD/eSD/eMMC device to internal RAM. After checking the Image Vector Table header value (0xD1) from Program Image, the ROM code performs a DCD check. After successful DCD extraction, the ROM code extracts from Boot Data Structure the destination pointer and length of image to be copied to RAM device from where code execution occurs.

The maximum image size to load in SD/MMC boot is 32MB. This is due to the limited number of uSDHC ADMA Buffer Descriptors allocated by ROM.

NOTE

The Initial 4Kbyte of Program Image must contain the IVT, DCD and the Boot Data structures.

Table 8-16. SD/MMC Frequencies

	SD	MMC	MMC (DDR Mode)
Identification (KHz)	347.22		
Normal Speed Mode (MHz)	25	20	25
High Speed Mode (MHz)	50	40	50
UHSI SDR50 (MHz)	100		
UHSI SDR104 (MHz)	200		

NOTE

The boot ROM code reads application image length and application destination pointer from image.

8.5.3.2 MMC and eMMC Boot

The following table provides MMC and eMMC boot details.

Table 8-17. MMC and eMMC Boot Details

Normal Boot Mode	During initialization (normal boot mode) the MMC frequency is set to 347.22 KHz. When the MMC card enters the identification portion of the initialization, voltage validation is performed and the ROM boot code checks high voltage settings and card capacity. The ROM boot code supports both high capacity and low capacity MMC/eMMC cards. After

Table continues on the next page...

Table 8-17. MMC and eMMC Boot Details (continued)

	initialization phase is complete, the ROM boot code switches to a higher frequency (20 MHz in Normal boot mode or 40 MHz in High Speed mode). eMMC is also interfaced via USDHC and follows the same flow as MMC. The boot partition can be selected for an MMC4.x card after the card initialization is complete. The ROM code reads the BOOT_PARTITION_ENABLE field in the Ext_CSD[179] to get the boot partition to be set. If there is no boot partition mentioned in BOOT_PARTITION_ENABLE field or the user partition has been mentioned, ROM boots from the user partition.
eMMC4.3 or eMMC4.4 Device Supporting Special Boot Mode	If using an eMMC4.3 or eMMC4.4 device supporting special boot mode, it can be initiated by pulling the CMD line low. If BOOT ACK is enabled, the eMMC4.3/eMMC4.4 device sends the BOOT ACK via DATA lines and ROM can read the BOOT ACK [S010E] to identify the eMMC4.3/eMMC4.4 device. eMMC4.3/eMMC4.4 device with "Boot mode" feature can only be supported via ESDHCV3-3 and with or without BOOT ACK. If BOOT ACK is enabled ROM waits 50 ms to get the BOOT ACK and if BOOT ACK is received by ROM. If BOOT ACK is disabled ROM waits 1 second for data. If BOOT ACK or data was received then eMMC4.3/eMMC4.4 is booted in "Boot mode", otherwise eMMC4.3/eMMC4.4 boots as a normal MMC card from the selected boot partition. This boot mode can be selected by BOOT_CFG1[4] (Fast Boot) fuse. BOOT ACK is selected by BOOT_CFG2[1].
eMMC4.4 Device	If using eMMC4.4 device, Double Data Rate (DDR) mode can be used. This mode can be selected by BOOT_CFG2[7:5] (Bus Width) fuse.

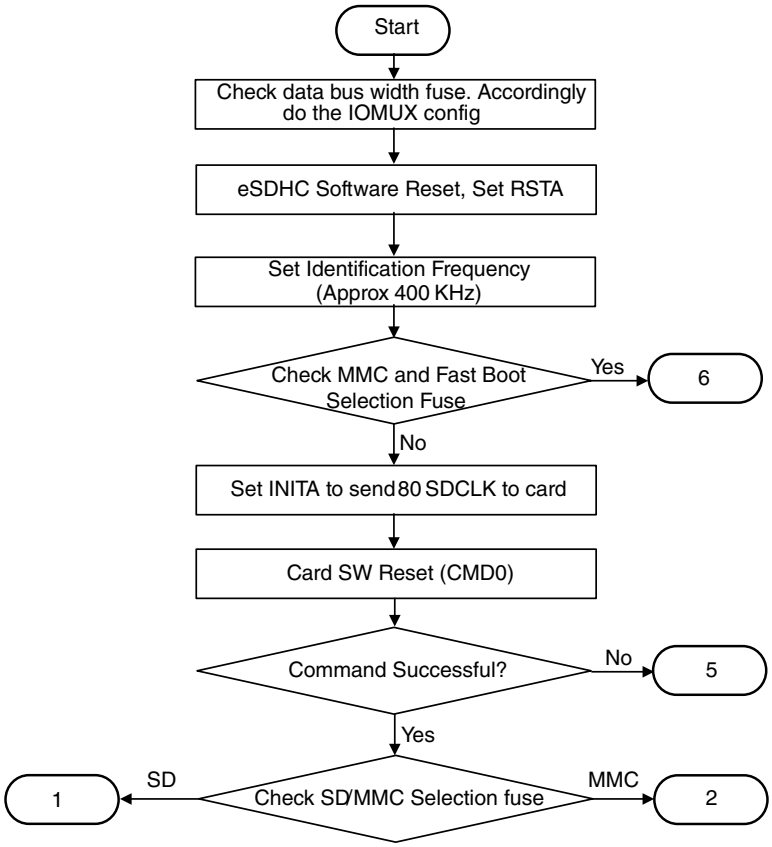


Figure 8-11. Expansion Device Boot Flow (1 of 6)

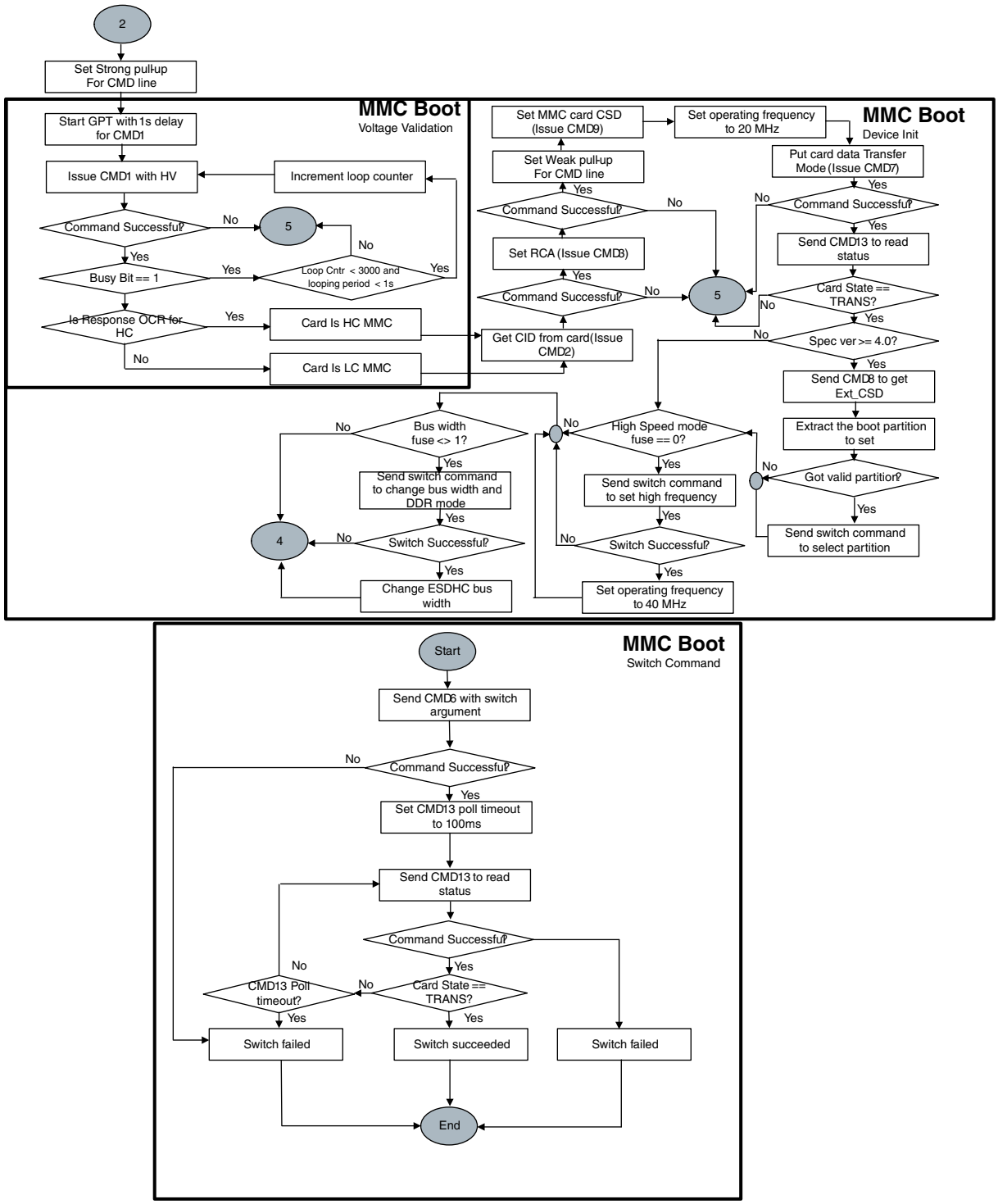


Figure 8-12. Expansion Device (MMC) Boot Flow (2 of 6)

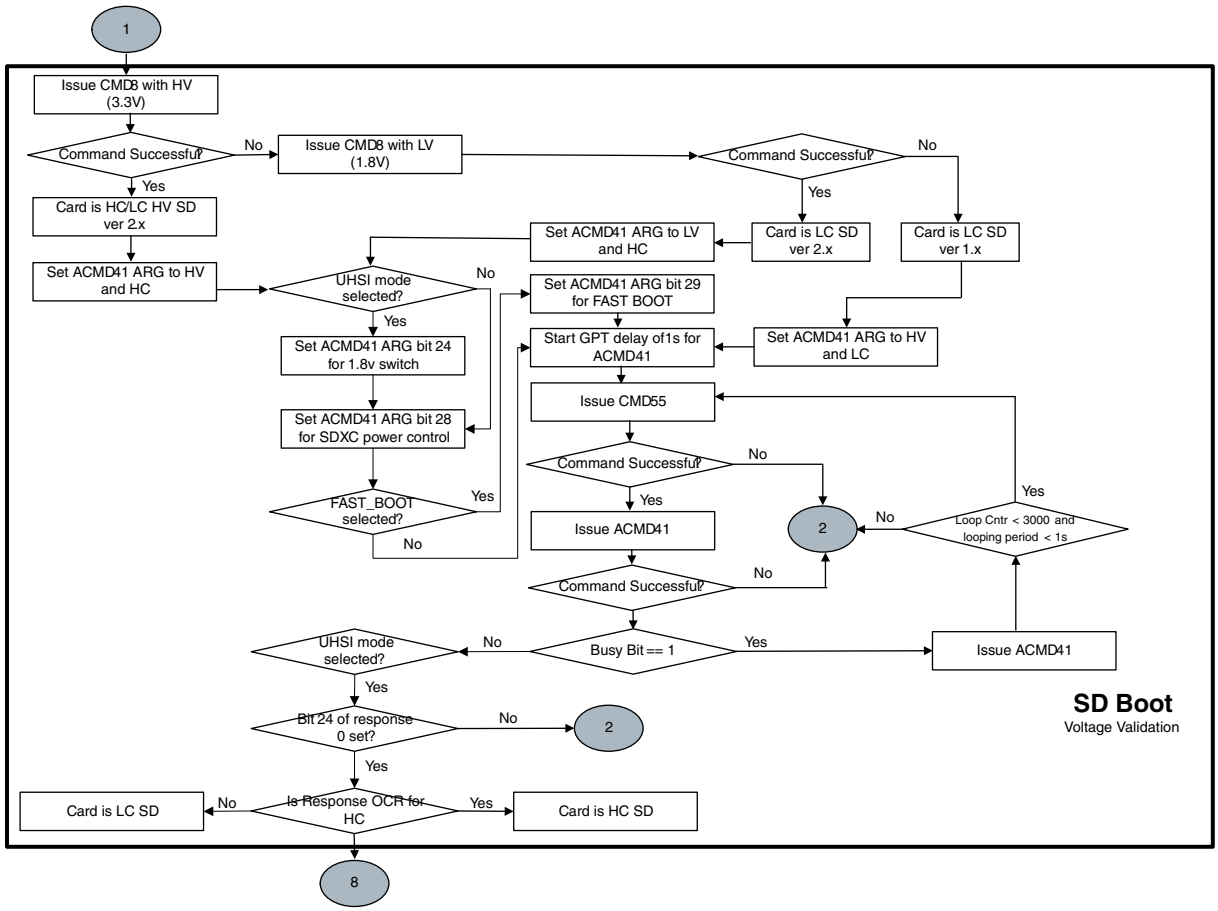


Figure 8-13. Expansion Device (SD/eSD/SDXC) Boot Flow (3 of 6) Part 1

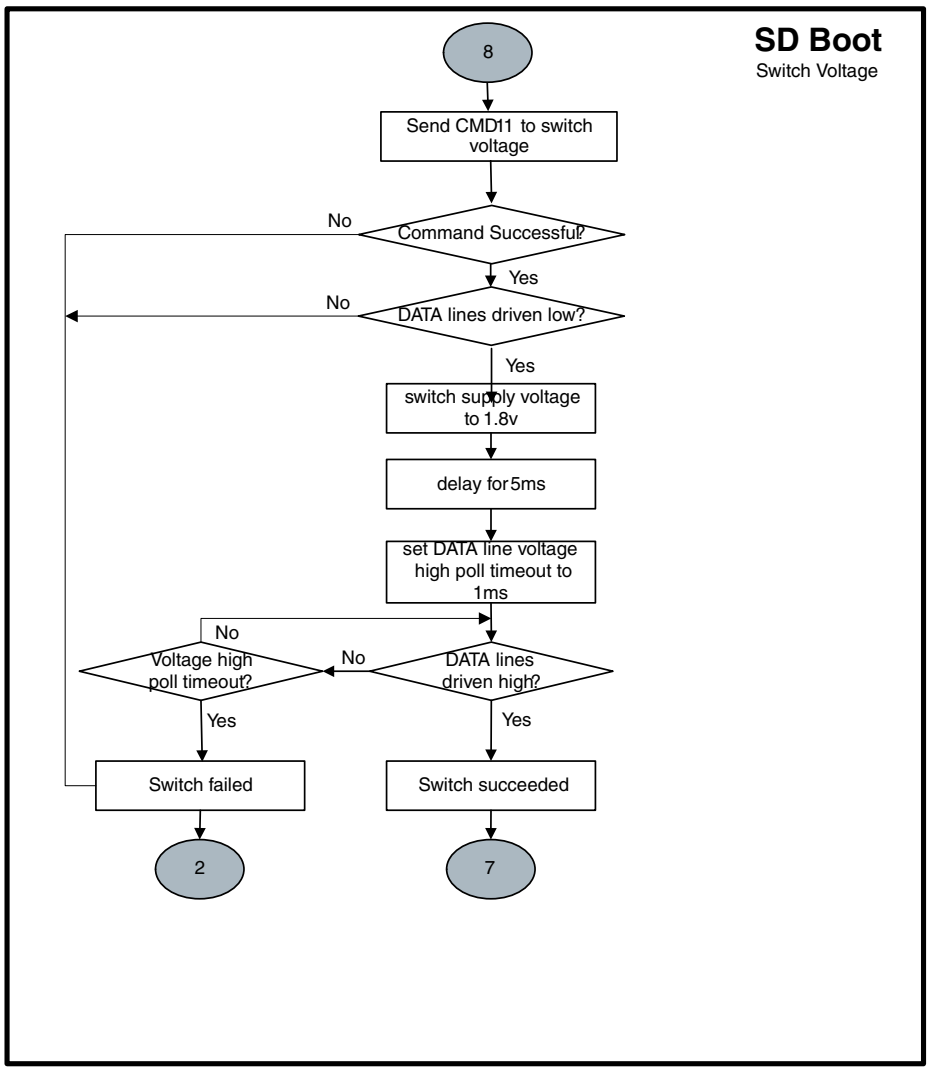
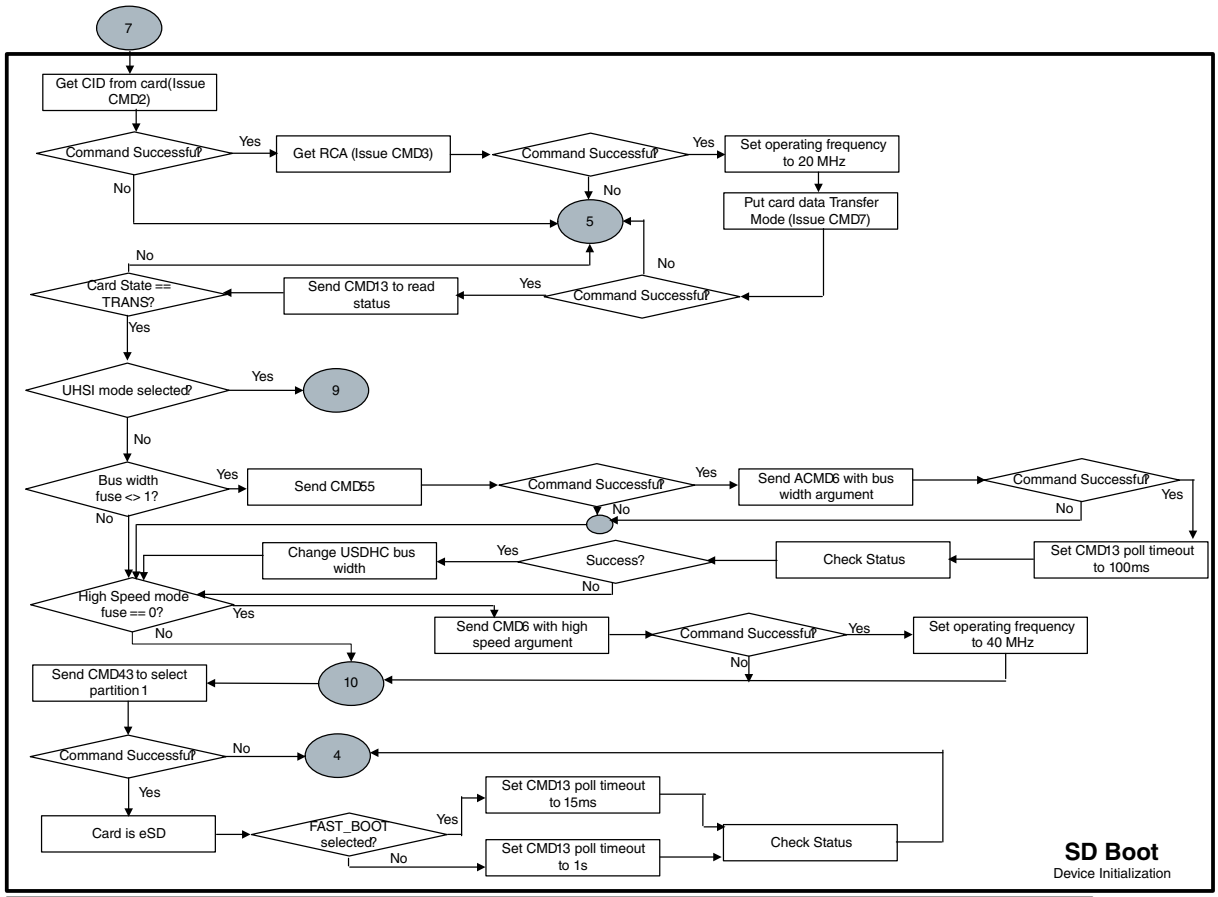


Figure 8-14. Expansion Device (SD/eSD/SDXC) Boot Flow (3 of 6) Part 2



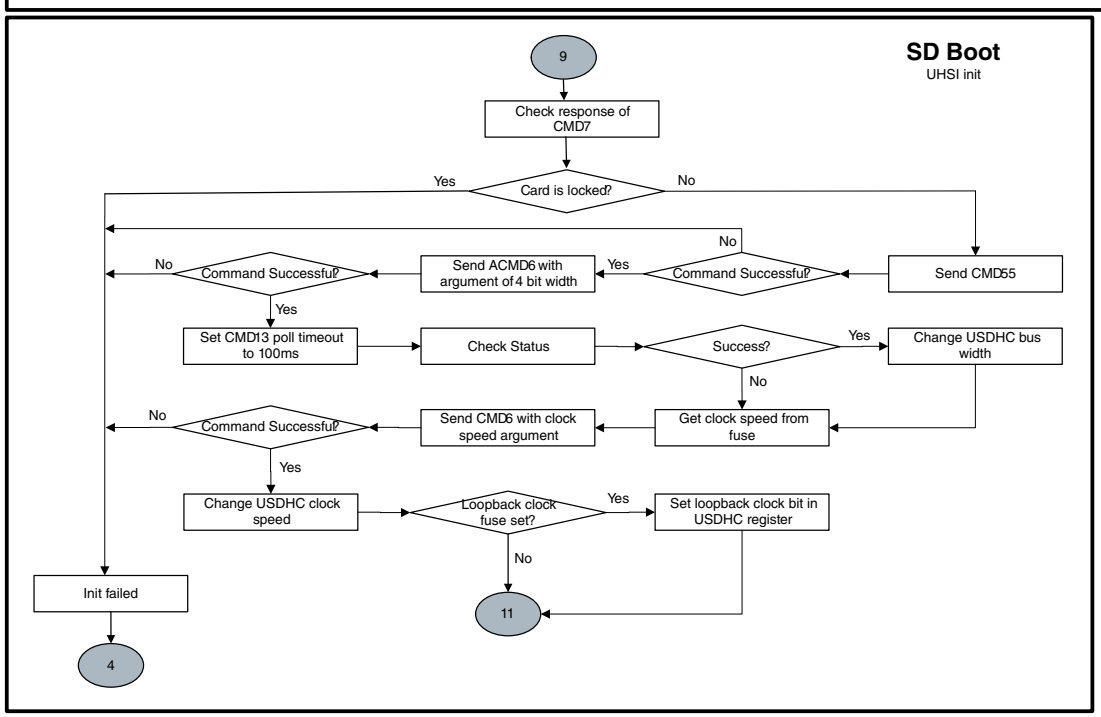


Figure 8-15. Expansion Device (MMCSD/eSD/SDXC) Boot Flow (4 of 6)

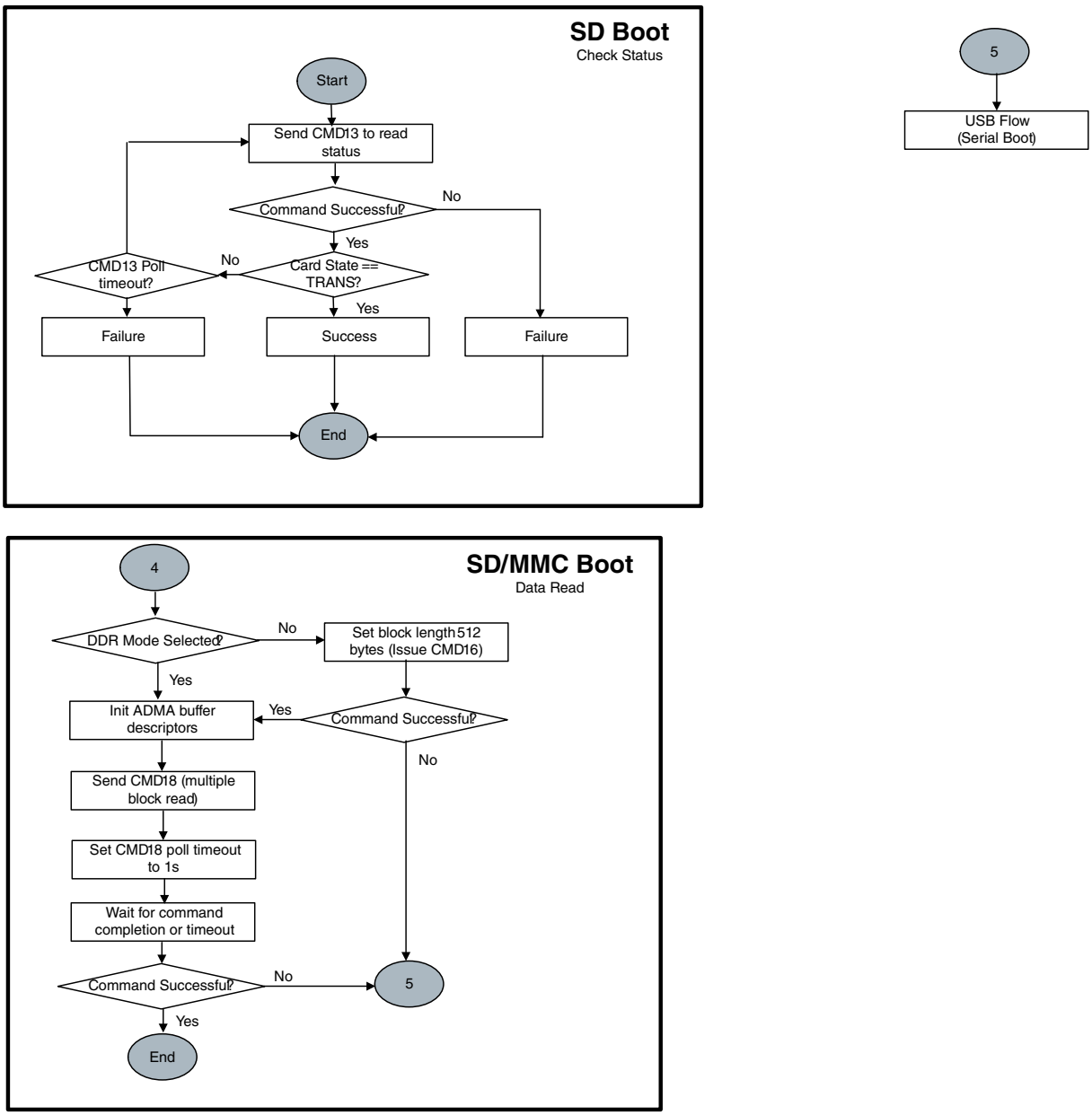
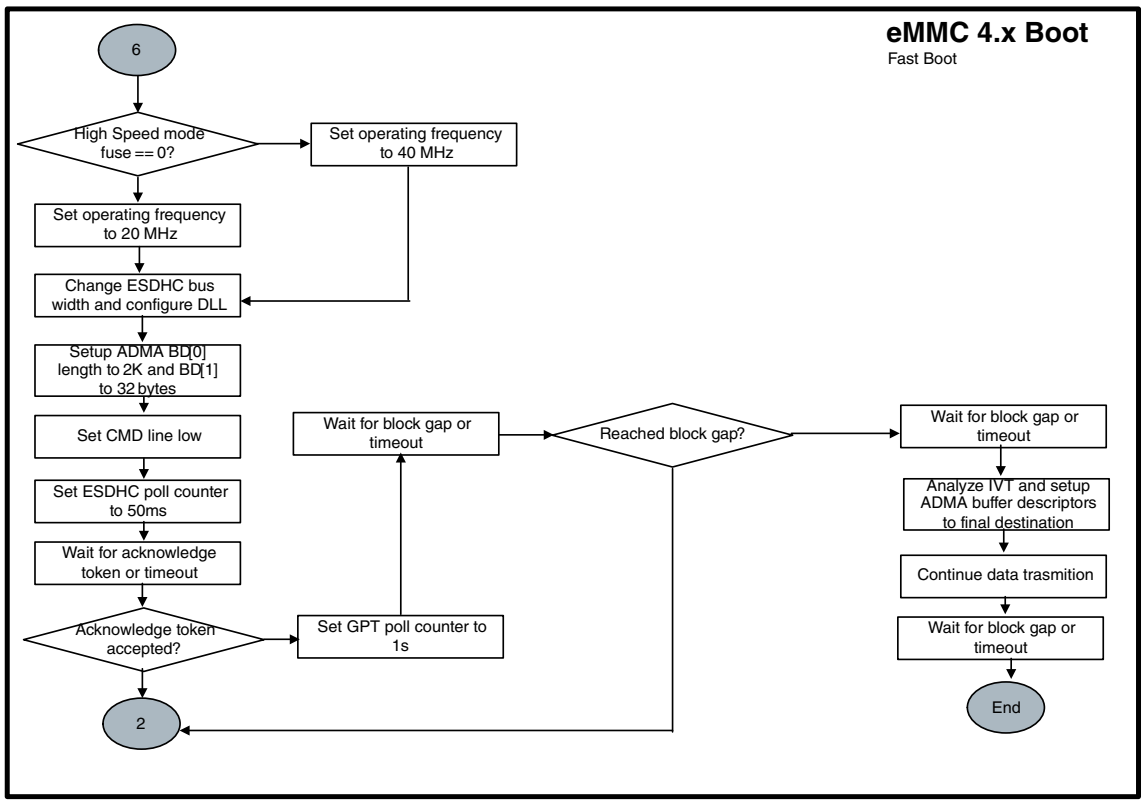


Figure 8-16. Expansion Device (SD/eSD) Boot Flow (5 of 6)



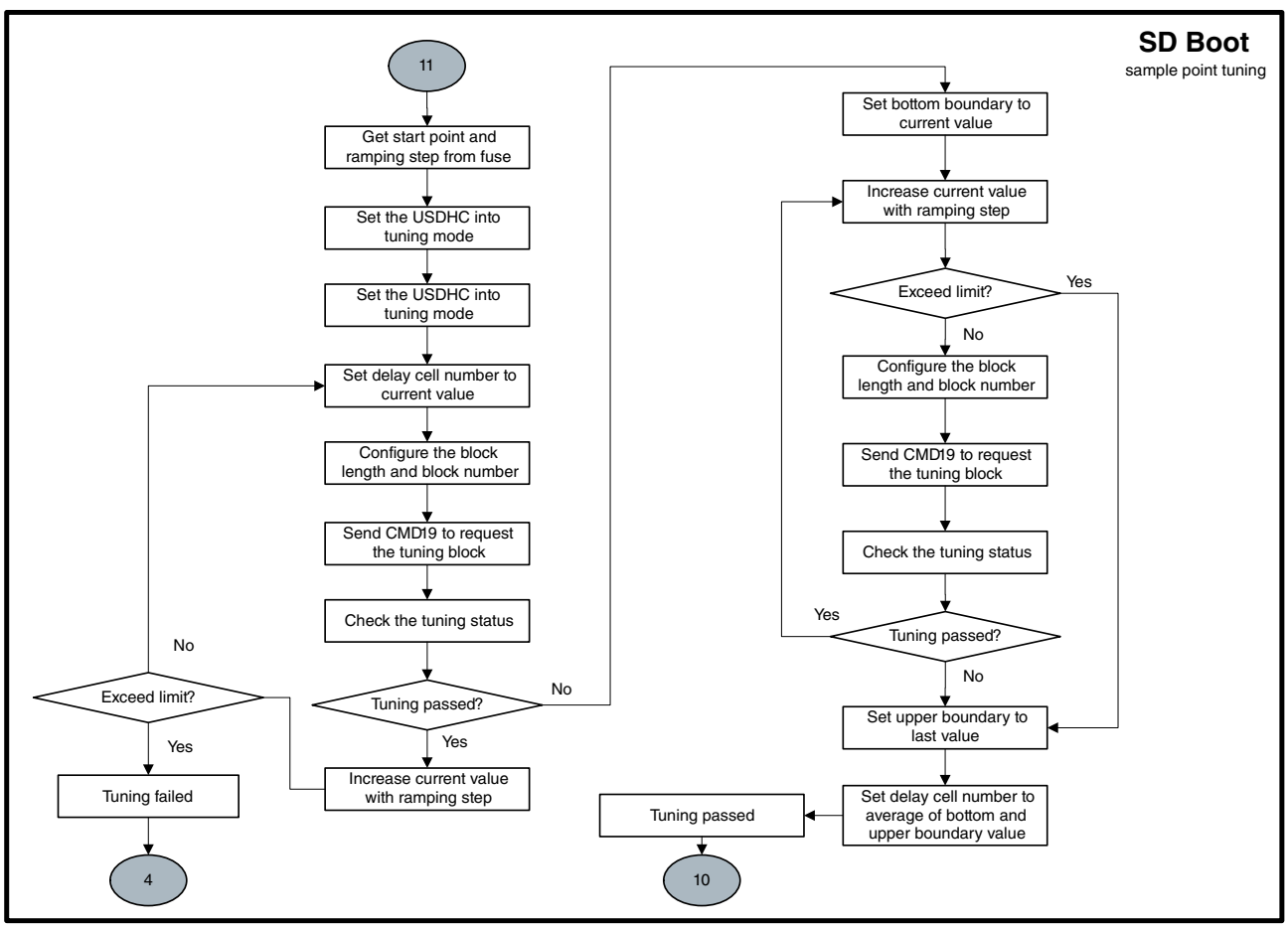


Figure 8-17. Expansion Device Boot Flow (6 of 6)

8.5.3.3 SD, eSD and SDXC

After the normal boot mode initialization begins, the SD/eSD/SDXC frequency is set to 347.22 kHz. During the identification phase, SD/eSD/SDXC card voltage validation is performed. During voltage validation, boot code first checks with high voltage settings; if that fails, it checks with low voltage settings.

The capacity of the card is also checked. Boot code supports high capacity and low capacity SD/eSD/SDXC cards after voltage validation card initialization is done.

During card initialization, the ROM boot code attempts to set the boot partition for all SD, eSD, and SDXC devices. If this fails, the boot code assumes the card is a normal SD card or SDXC card. If it does not fail, the boot code assumes it is an eSD card. After the initialization phase is over, boot code switches to a higher frequency (25 MHz in Normal Speed mode or 50 MHz in High Speed Mode). ROM also supports FAST_BOOT mode booting from eSD card. This mode can be selected by BOOT_CFG1[4] (Fast Boot) fuse described in Table 8-15.

For UHSI cards, clock speed fuses can be set to SDR50 or SDR104 on USDHC1, USDHC2, ports. This will enable the voltage switch process to set the signaling voltage to 1.8V during voltage validation. The bus width is fixed at 4 bits wide and a sampling point tuning process is needed to calibrate the number of delay cells. If SD Loopback Clock eFuse is set, the feedback clock will come directly from the loopback SD clock, instead of the card clock (by default). The SD clock speed can be selected by BOOT_CFG1[3:2], and the SD Loopback Clock is selected by BOOT_CFG1[0].

UHSI calibration start value (MMC_DLL_DLY[6:0]) and step value (BOOT_CFG2[7:5]) can be set to optimize the sample point tuning process.

If SD Power Cycle Enable eFuse is 1, ROM will set SD_RST pad low, wait 5ms and then set SD_RST pad high. If SD_RST pad is connected to SD power supply enable logic on board, it enables power cycle of SD card. This may be crucial in case when SD logic is in 1.8V states and must be reset to 3.3V states.

SDR50 and SDR104 boot are not supported on the USDHC1 and USDHC2 ports because there are no reset signals for those ports when connected in the IOMUX.

8.5.3.4 IOMUX Configuration for SD/MMC

Table 8-18. SD/MMC IOMUX Pin Configuration

Signal	USDHC1	USDHC2
CLK	SD1_CLK.alt0	NAND_RE_B.alt1
CMD	SD1_CMD.alt0	NAND_WE_B.alt1
DATA0	SD1_DATA0.alt0	NAND_DATA00.alt1
DATA1	SD1_DATA1.alt0	NAND_DATA01.alt1
DATA2	SD1_DATA2.alt0	NAND_DATA02.alt1
DATA3	SD1_DATA3.alt0	NAND_DATA03.alt1
DATA4	NAND_READY_B.alt1	NAND_DATA04.alt1
DATA5	NAND_CE0_B.alt1	NAND_DATA05.alt1
DATA6	NAND_CE1_B.alt1	NAND_DATA06.alt1
DATA7	NAND_CLE.alt1	NAND_DATA07.atl1
VSELECT	GPIO1_IO05.alt4	GPIO1_IO08.alt4
RESET_B	GPIO1_IO09.alt5	NAND_ALE.alt5
CD_B	UART1_RTS_B.alt2	—

8.5.3.5 Redundant Boot Support for Expansion Device

ROM supports redundant boot for expansion device. Primary or Secondary image is selected depending on PERSIST_SECONDARY_BOOT setting (see Table 8-6).

If PERSIST_SECONDARY_BOOT is 0, the boot ROM uses address 0x0 for primary image.

If PERSIST_SECONDARY_BOOT is 1, the boot ROM will read secondary image table from address 0x200 on boot media and will use address specified in the table.

Table 8-19. Secondary Image Table Format

Reserved (chipNum)
Reserved (driveType)
tag
firstSectorNumber
Reserved (sectorCount)

Where:

- tag: used as indication of valid secondary image table. Must be 0x00112233.
- firstSectorNumber is the first 512B sector number of the secondary image.

For secondary image support, the primary image must reserve space for secondary image table. See the figure below for typical structures layout on expansion device.

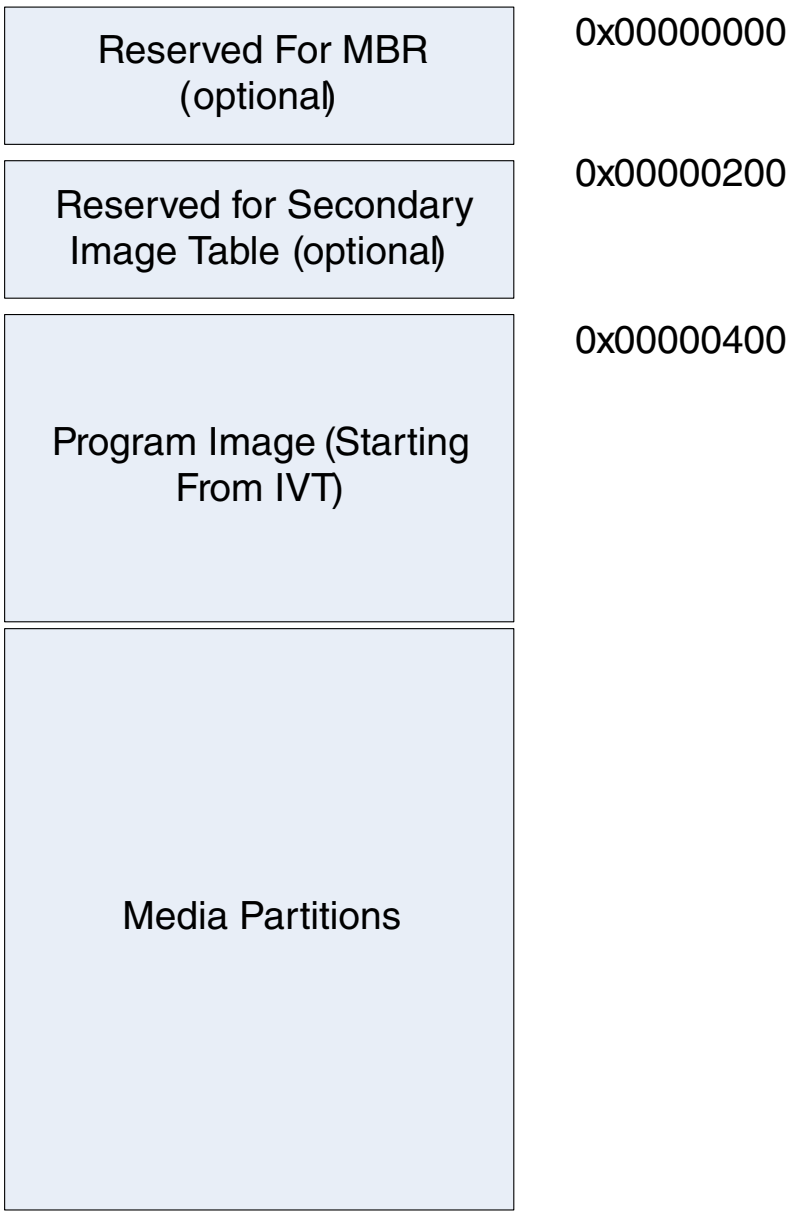


Figure 8-18. Expansion Device Structures Layout

For Closed mode, if there are failures during primary image authentication, the boot ROM will turn on PERSIST_SECONDARY_BOOT bit (see Table 8-6) and perform software reset. (After software reset, secondary image will be used.)

8.5.4 Serial ROM through SPI

The chip supports boot from serial memory devices, such as EEPROM and Serial Flash using the SPI.

The following ports are available for serial boot: eCSPI (eCSPI1, eCSPI2, eCSPI3, eCSPI4) interfaces.

8.5.4.1 Serial ROM eFUSE Configuration

The boot ROM code determines the type of device using the following parameters, either provided by eFUSE settings or sampled on the I/O pins, during boot.

See the table below for details:

Table 8-20. Serial ROM Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG1[7:4]	OEM	Boot Device Selection	Yes	0000	0011 - Boot from Serial ROM
BOOT_CFG4[6]	OEM	EEPROM Recovery Enable	Yes	0	0 - Disabled EEPROM recovery 1 - Enabled EEPROM recovery
BOOT_CFG4[5:4]	OEM	CS select (SPI only)	Yes	00	00 - ECSPIx_SS0 01 - ECSPIx_SS1 10 - ECSPIx_SS2 11 - ECSPIx_SS3
BOOT_CFG4[3]	OEM	SPI Addressing (SPI only)	Yes	0	0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)
BOOT_CFG4[2:0]	OEM	Port Select	Yes	00	000 - ECSPI-1 001 - ECSPI-2 010 - ECSPI-3 011 - ECSPI-4 1xx- Reserved

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See GPIO Boot Overrides for corresponding GPIO pin.

The ECPSI-1/ECPSI-2/ECPSI-3/ECPSI-4 block can be used as boot device using ECSPI interface for serial ROM boot. The SPI interface is configured to operate at 15 MHz for 3-byte addressing device and 3.75 MHz for 2-byte addressing devices.

The boot ROM will copy 4Kbyte of data from Serial ROM device to internal RAM. After checking the Image Vector Table header value (0xD1) from Program Image, the ROM code performs a DCD check. After successful DCD extraction, the ROM code extracts from Boot Data Structure the destination pointer and length of image to be copied to RAM device from where code execution occurs.

NOTE

The Initial 4K of Program Image must contain the IVT, DCD and the Boot Data structures.

8.5.4.2 ECSPI Boot

The Enhanced Configurable SPI (ECSPI) interface is configured in master mode and the EEPROM device is connected to ECSPI interface as a slave.

The boot ROM code copies 4 KB data from EEPROM device to the internal RAM. If DCD verification is successful, the ROM code copies the initial 4 KB data, as well as the rest of the image extracted from application image, directly to the application destination. The ECSPI can read data from EEPROM using 2 or 3 byte addressing. Its burst length is 32 bytes.

NOTE

The Serial ROM Chip Select Number is determined by BOOT_CFG4[5:4] (Chip Select) fuse.

When using the SPI as boot device, the Chip supports booting from both Serial EEPROM and Serial Flash devices. The boot code determines which device is being used by reading the appropriate eFUSE/I/O values at boot (see Table 8-20 for details).

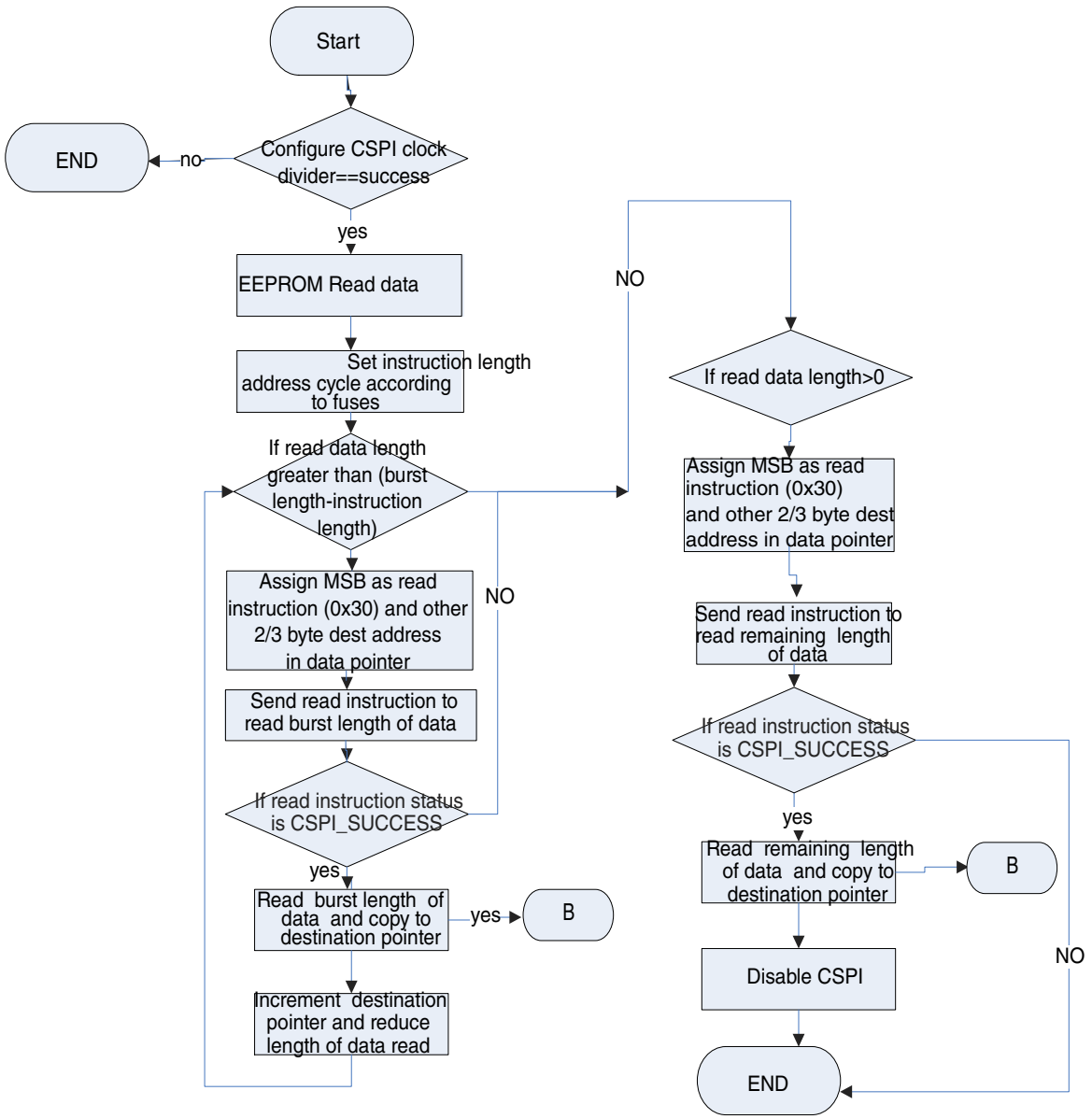


Figure 8-19. CSPI Flow chart

8.5.4.2.1 ECSPI IOMUX Pin Configuration

The contacts assigned to the signals used by the CSPI blocks is shown in the table below.

Table 8-21. ECSPI IOMUX Pin Configuration

Signal	eCSPI1	eCSPI2	eCSPI3	eCSPI4
MISO	CSI_DATA07.alt3	CSI_DATA03.alt3	UART2_RTS_B.alt8	ENET2_TX_CLK.alt3
MOSI	CSI_DATA06.alt3	CSI_DATA02.alt3	UART2_CTS_B.alt8	ENET2_TX_EN.alt3

Table continues on the next page...

Table 8-21. ECSPI IOMUX Pin Configuration (continued)

Signal	eCSPI1	eCSPI2	eCSPI3	eCSPI4
SCLK	CSI_DATA04.alt3	CSI_DATA00.alt3	UART2_RX_DATA.alt8	ENET2_TX_DATA1.alt 3
SS0	CSI_DATA05.alt3	CSI_DATA01.alt3	UART2_TX_DATA.alt8	ENET2_RX_ER.alt3
SS1	LCD_DATA05.alt8	LCD_HSYNC.alt8	NAND_ALE.alt8	NAND_DATA01.alt8
SS2	LCD_DATA06.alt8	LCD_VSYNC.alt8	NAND_RE_B.alt8	NAND_DATA02.alt8
SS3	LCD_DATA07.alt8	LCD_RESET.alt8	NAND_WE_B.alt8	NAND_DATA03.alt8

8.6 QuadSPI Serial Flash Memory Boot

8.6.1 QuadSPI eFUSE Configuration

Table 8-22. QSPI Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO	Shipped Value	Settings
BOOT_CFG1[7:4]	OEM	Boot Device Selection	Yes	0001	0001 - Boot from QuadSPI
BOOT_CFG1[3]	OEM	QuadSPI Interface Selection	Yes	0	0 - QSPI1 1 - Reserved

8.6.2 QuadSPI Serial Flash BOOT Operation

The Boot ROM attempts to boot from QuadSPI flash if the BOOT_CFG1[7:4] fuses are programmed to 0001 as shown in the QuadSPI eFUSE Configuration table. The ROM will initialize the requested the QuadSPI Interface as selected in Fuse bit BOOT_CFG1[3] in the QuadSPI eFUSE Configuration. QuadSPI interface initialization is a two step process.

The ROM expects the QuadSPI configuration parameters as explained in the QuadSPI Configuration Parameters to be present in the Serial Flash memory from offset 0x400 of serial flash of length 512 bytes. The ROM reads these configuration parameters using the default read command configured in the LUT of the QuadSPI interface with SCLOCK operating at 18 MHz.

In the second step, ROM configures the selected QuadSPI interface with the configuration parameters read from the serial flash and starts the boot procedure. Refer to Table 19-12 for details regarding QuadSPI configuration parameters and to the QuadSPI boot flow chart for detailed boot flow chart of QuadSPI.

Both booting an XIP and non XIP image is supported from serial flash. For XIP boot, the image has to be built for QuadSPI address space and for non XIP the image can be built to execute from DDR or OCRAM.

For QUAD mode boot, the Boot ROM expects the Quad Enable bit inside the QSPI Flash to be already set before booting starts. Therefore, the QUAD enable bit must be set in the non-volatile register of the flash at the time of programming.

NOTE

If the SPI flash device requires quad enable command, it can be sent via configuration structure fields: device_quad_mode_en, device_cmd, write_cmd_ipcr, write_enable_ipcr, busy_bit_offset, read_status_ipcr.

8.6.3 QuadSPI Configuration Parameters

The QuadSPI Configuration Parameters Table is built in boot image at fixed offset 0x400 from QSPI NOR A1 base address (368 bytes). Table below lists various QuadSPI Configuration Parameters.

Table 8-23. QuadSPI Configuration Parameters

Name	Offset	Size in Bytes	Description
DQS Loopback	0	4	DQS LoopBack Mode to enable Dummy Pad, 0 - Disable, 1 - Enable
Hold Delay	4	4	Hold Delay for QSPI[0,1] A/B Value / QSPI1 B / QSPI1 A 00 / Disable / Disable 01 / Disable / Enable 10 / Enable / Disable 11 / Enable / Enable
Reserved	8	4	Reserved to 0
Reserved	12	4	Reserved to 0
device_quad_mode_en	16	4	Send Quad enable command to SPI device.
device_cmd	20	4	Command to send to SPI device.

Table continues on the next page...

Table 8-23. QuadSPI Configuration Parameters (continued)

<table>
<tr><th>Name</th><th>Offset</th><th>Size in Bytes</th><th>Description</th></tr>
<tr><td>write_cmd_ipcr</td><td>24</td><td>4</td><td>IPCR register value for write command</td></tr>
<tr><td>write_enable_ipcr</td><td>28</td><td>4</td><td>IPCR register value for Enable</td></tr>
<tr><td>Chip Select hold time</td><td>32</td><td>4</td><td>This is chip select hold time in terms of Serial clock (For Example 1 serial clock cycle 0-15).</td></tr>
<tr><td>Chip Select setup time</td><td>36</td><td>4</td><td>Chip select setup time in terms of Serial clock (For example 1 serial clock).</td></tr>
<tr><td>Serial Flash A1 size</td><td>40</td><td>4</td><td>Serial Flash A1 size in units of bytes</td></tr>
<tr><td>Serial Flash A2 size</td><td>44</td><td>4</td><td>Serial Flash A2 size in units of bytes</td></tr>
<tr><td>Serial Flash B1 size</td><td>48</td><td>4</td><td>Serial Flash B1 size in units of bytes</td></tr>
<tr><td>Serial Flash B2</td><td>52</td><td>4</td><td>Serial Flash B2 size in units of bytes</td></tr>
<tr><td>Serial Clock Frequency</td><td>56</td><td>4</td><td>This is serial clock frequency select parameter.
<table><tr><th>Value</th><th>Clock</th></tr><tr><td>00</td><td>18 MHz</td></tr><tr><td>01</td><td>49 MHz</td></tr><tr><td>02</td><td>55 MHz</td></tr><tr><td>03</td><td>60 MHz</td></tr><tr><td>04</td><td>66 MHz</td></tr><tr><td>05</td><td>76 MHz</td></tr><tr><td>06</td><td>99 MHz (only SDR mode)</td></tr></table></td></tr>
<tr><td>busy_bit_offset</td><td>60</td><td>4</td><td>SPI Flash device busy bit offset in its status register, used for enabling Quad mode of SPI device</td></tr>
<tr><td>Mode of operation of serial Flash</td><td>64</td><td>4</td><td>This field describes the mode of operation of Serial flash
<table><tr><th>Value</th><th>Mode</th></tr><tr><td>01</td><td>Single</td></tr><tr><td>02</td><td>Dual</td></tr><tr><td>04</td><td>Quad</td></tr></table></td></tr>
<tr><td>Serial Flash Port B Selection</td><td>68</td><td>4</td><td>Port A is always available. This field informs the device ROM the availability of Port B.
0 – Port B is not used
1 – Port B is used</td></tr>
<tr><td>Dual Data Rate mode enable</td><td>72</td><td>4</td><td>This field enables the device ROM to enable DDR mode.
0 – DDR mode is disabled
1 – DDR mode is enabled</td></tr>
</table>

Table continues on the next page...

Table 8-23. QuadSPI Configuration Parameters (continued)

Name	Offset	Size in Bytes	Description
Data Strobe Signal enable in Serial Flash	76	4	This field enables Data Strobe signal in Serial Flash which supports it. 0 – Disable DQS 1 – Enable DQS
Parallel Mode enable	80	4	This field enables parallel mode. Data will be read from serial Flash in parallel mode. Refer to QSP chapter for detail. 0 – Disable Parallel mode in QSPI 1 – Enable Parallel Mode in QSPI
CS1 on Port A	84	4	This field enables CS1 on port A 0 – Disable CS1 on Port A 1 – Enable CS1 on Port A
CS1 on Port B	88	4	This field enables CS1 on port B 0 – Disable CS1 on Port B 1 – Enable CS1 on Port B
Full Speed Phase Selection	92	4	Select the edge of the sampling clock valid for full speed commands: 0: Select sampling at non-inverted clock 1: Select sampling at inverted clock This bit is also used to shift the dqs_enable when DQS mode is selected
Full Speed Delay Selection	96	4	Select the delay w.r.t. the reference edge for the sample point valid for full speed commands: 0: One clock cycle delay 1: Two clock cycles delay This bit is also used to shift the dqs_enable when DQS mode is selected
DDR Sampling Point	100	4	Select the sampling point for incoming data when serial flash is in DDR mode. **NOTE:** Valid Values are (b000-b111)
LUT program sequence	104	256	256 Bytes of Look up table program sequence. ROM programs the LUT of QuadSPI with this parameter supplied. It assumes that the optimize read command sequence which will be used to read data from Serial flash and fill the AHB buffer is programmed at index 0.
read_status_ipcr	360	4	IPCR value of Read Status Reg
enable_dqs_phase	364	4	Enable DQS Phase
Reserved	368	36	Not used
dqs_pad_setting_override	404	4	DQS pin pad setting override
sclk_pad_setting_override	408	4	SCLK pin pad setting override
data_pad_setting_override	412	4	DATA pins pad setting override
cs_pad_setting_override	416	4	CS pins pad setting override

Table continues on the next page...

Table 8-23. QuadSPI Configuration Parameters (continued)

Name	Offset	Size in Bytes	Description
dqs_loopback_internal	420	4	0: dqs loopback from pad 1: dqs loopback internally
dqs_phase_sel	424	4	dqs phase selection
dqs_fa_delay_chain_sel	428	4	dqs fa delay chain selection
dqs_fb_delay_chain_sel	432	4	dqs fb delay chain selection
sclk_fa_delay_chain_sel	436	4	sclk fa delay chain selection
sclk_fb_delay_chain_sel	440	4	sclk fb delay chain selection
Reserved	444	64	Reserved
tag	508	4	End flag of QSPI parameters area

8.6.4 IOMUX Configuration for QSPI Devices

The QSPI interface uses dedicated contacts on the IC. The contacts assigned to the data signals used by QSPI are shown in the table below.

Table 8-24. QuadSPI IOMUX Pin Configuration

Signal	QSPI1
A_SCLK	NAND_WP_b.alt2
A_SS0_B	NAND_DQS.alt2
A_SS1_B	NAND_DATA07.alt2
A_DATA0	NAND_READY_B.alt2
A_DATA1	NAND_CE0_B.alt2
A_DATA2	NAND_CE1.B.alt2
A_DATA3	NAND_CLE.alt2
A_DQS	NAND_ALE.alt2
B_SCLK	NAND_RE_B.alt2
B_SS0_B	NAND_WE_B.alt2
B_SS1_B	NAND_DATA00.alt2
B_DATA0	NAND_DATA02.alt2
B_DATA1	NAND_DATA03.alt2
B_DATA2	NAND_DATA04.alt2
B_DATA3	NAND_DATA05.alt2
B_DQS	NAND_DATA01.alt2

8.6.5 QuadSPI boot flow chart

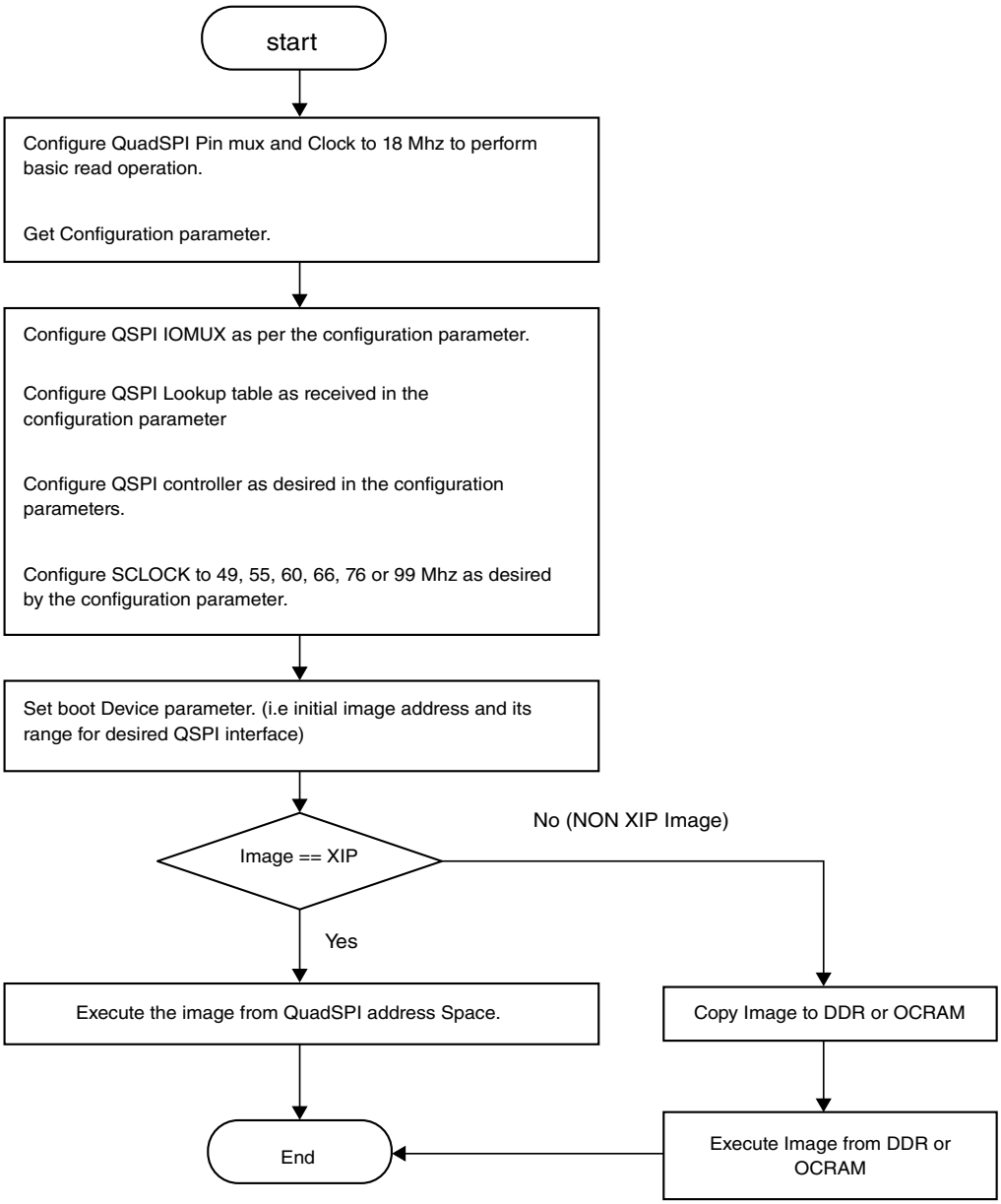


Figure 8-20. QuadSPI boot flow chart

NOTE

If flash is configured for "High performance mode (where command is generated only once)" in LUT program sequence. then external reset should be routed to flash reset to allow rebooting in case of any device reset other than Power On Reset. Also this high performance mode must be exited by application before any Low power mode entry where the device

is supposed to reboot from QSPI flash on Low power mode exit. In general, any preserved configuration in external flash will not be understood by device after reset.

8.7 Program image

This section describes the data structures that are required to be included in a user's program image. A program image consists of:

- Image vector table—A list of pointers located at a fixed address that the ROM examines to determine where other components of the program image are located
- Boot data—A table indicating the program image location, program image size in bytes, and the plugin flag
- Device configuration data—IC configuration data
- User code and data

8.7.1 Image Vector Table and Boot Data

The Image Vector Table (IVT) is the data structure that the ROM reads from the boot device supplying the program image containing the required data components to perform a successful boot.

The IVT includes the program image entry point, a pointer to Device Configuration Data (DCD) and other pointers used by the ROM during the boot process.The ROM locates the IVT at a fixed address that is determined by the boot device connected to the Chip. The IVT offset from the base address and initial load region size for each boot device type is defined in the table below. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

Table 8-25. Image Vector Table Offset and Initial Load Region Size

Boot Device Type	Image Vector Table Offset	Initial Load Region Size
NOR	4 Kbyte = 0x1000 bytes	Entire Image Size
OneNAND	256 bytes = 0x100 bytes	1 Kbyte
SD/MMC/eSD/eMMC/SDXC	1 Kbyte = 0x400 bytes	4 Kbyte
SPI EEPROM	1 Kbyte = 0x400 bytes	4 Kbyte

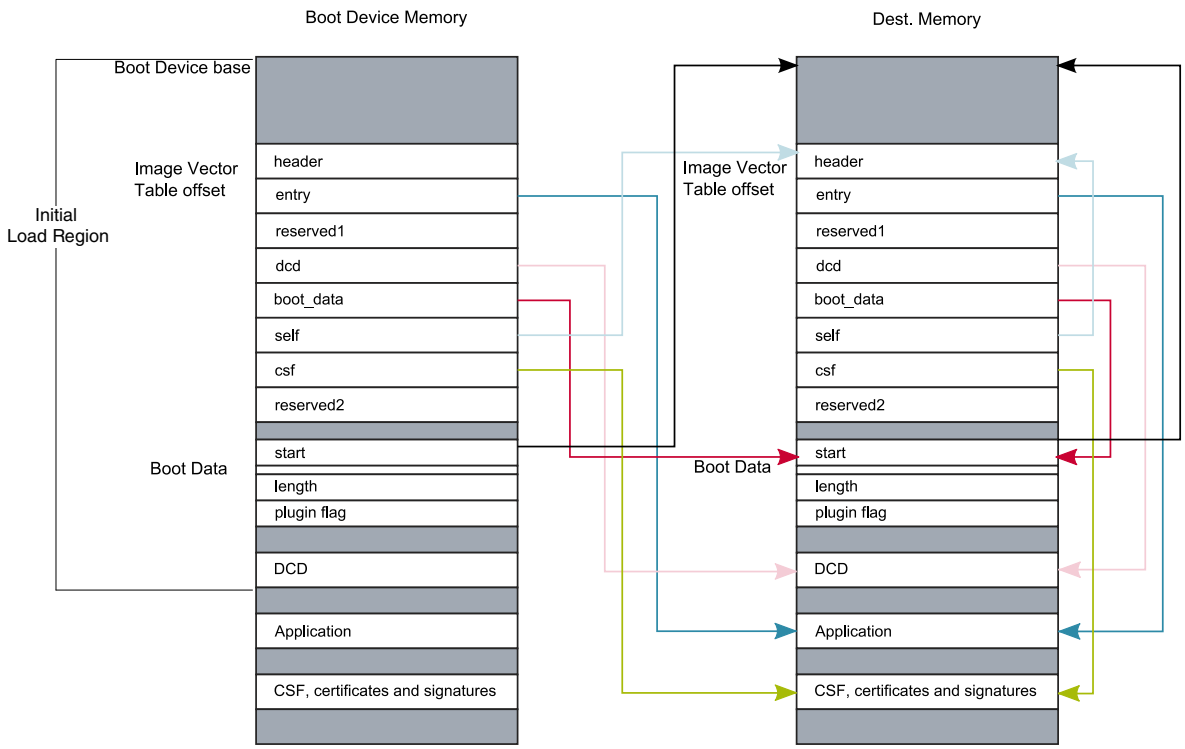


Figure 8-21. Image Vector Table

8.7.1.1 Image Vector Table Structure

The IVT has the following format where each entry is a 32 bit word:

Table 8-26. IVT Format

header
entry: Absolute address of the first instruction to execute from the image
reserved1: Reserved and should be zero
dcd: Absolute address of the image DCD. The DCD is optional so this field may be set to NULL if no DCD is required. See Device Configuration Data (DCD) for further details on DCD.
boot data: Absolute address of the Boot Data
self: Absolute address of the IVT. Used internally by the ROM
csf: Absolute address of Command Sequence File (CSF) used by the HAB library. See High Assurance Boot (HAB) for details on secure boot using HAB. This field must be set to NULL when not performing a secure boot
reserved2: Reserved and should be zero

Figure 8-22 shows the IVT header format:



Figure 8-22. IVT Header Format

where:

```
Tag: A single byte field set to 0xD1
Length: a two byte field in big endian format containing the overall length of the IVT,
in bytes, including the header. (the length is fixed and must have a value of
32 bytes)
Version: A single byte field set to 0x40 or 0x41
```

8.7.1.2 Boot Data Structure

The Boot Data must follow the format defined in the table found here, each entry is a 32-bit word.

Table 8-27. Boot Data Format

start	Absolute address of the image
length	Size of the program image
plugin	Plugin flag (see Plugin Image)

8.7.2 Device Configuration Data (DCD)

Upon reset, the Chip uses the default register values for all peripherals in the system. However, these settings typically are not ideal for achieving optimal system performance and there are even some peripherals that must be configured before they can be used.

The DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various peripherals on the Chip.

For example, the EIM default settings allow the core to interface to a NOR flash device immediately out of reset. This allows the Chip to interface with any NOR flash device, but has the cost of slow performance. Additionally, some components such as DDR require some sequence of register programming as part of configuration before it is ready to be used. The DCD feature can be used to program the EIM registers and MMDC registers to the optimal settings.

The ROM determines the location of the DCD table based on information located in the Image Vector Table (IVT). See Image Vector Table and Boot Data for more details. The DCD table shown below is a big endian byte array of the allowable DCD commands. The maximum size of the DCD limited to 1768 bytes.



Figure 8-23. DCD Data Format

The DCD header is 4 bytes with the following format:

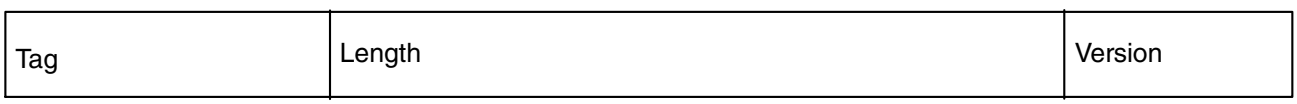


Figure 8-24. DCD Header Format

where:

```
Tag: A single byte field set to 0xD2
Length: a two byte field in big endian format containing the overall length of the DCD,
in bytes, including the header
Version: A single byte field set to 0x41
```

8.7.2.1 Write Data Command

The Write Data Command is used to write a list of given 1-, 2- or 4-byte values or bitmasks to a corresponding list of target addresses.

The format of Write Data Command, again a big endian byte array, is shown in the table below.

Table 8-28. Write Data Command Format

Tag	Length	Parameter
Address		

Table continues on the next page...

Table 8-28. Write Data Command Format (continued)

Value/Mask
[Address]
[Value/Mask]
...
[Address]
[Value/Mask]

where:

```
Tag: A single byte field set to 0xCC
Length: A two byte field in big endian format containing the length of the Write Data
Command, in bytes, including the header
Address: target address to which data should be written
Value/Mask: data value or bitmask to be written to preceding address
```

The Parameter field is a single byte divided into bitfields as follows:

Table 8-29. Write Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

One or more target address and value/bitmask pairs can be specified. The same bytes and flags parameters apply to all locations in the command.

When successful, this command writes to each target address in accordance with the flags as follows:

Table 8-30. Interpretation of Write Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	*address = val_msk	Write value
0	1	*address = val_msk	Write value
1	0	*address &= ~val_msk	Clear bitmask
1	1	*address |= val_msk	Set bitmask

NOTE

If any of the target addresses does not have the same alignment as the data width indicated in the parameter field, none of the values are written.

If any of the values is larger or any of the bitmasks is wider than permitted by the data width indicated in the parameter field, none of the values are written.

If any of the target addresses do not lie within an allowed region, none of the values are written. The list of allowable blocks and target addresses for the Chip are given below.

Table 8-31. Valid DCD Address Ranges

Address range	Start address	Last Address
IOMUX Control (IOMUXC) registers	0x020E0000	0x020E3FFF
IOMUXC GPR	0x020E4000	0x020E7FFF
CCM register set	0x020C4000	0x020C7FFF
ANADIG registers	0x020C8000	0x020C8FFF
MMDC register set	0x021B0000	0x021B3FFF
OCRAM free space	0x00907000	0x00937FF0
EIM registers	0x021B8000	0x021BBFFF
DDR	0x10000000	0xFFFF7FFF
EPIT	EPIT1: 0x020D0000	EPIT1: 0x020D3FFF

8.7.2.2 Check Data Command

The Check Data Command is used to test for a given -1, 2- or 4-byte bitmasks from a source address.

The Check Data Command is a big endian byte array with format shown in the table below.

Table 8-32. Check Data Command Format

Tag	Length	Parameter
Address		
Mask		
[Count]		

where:

```
Tag: A single byte field set to 0xCF
Length: A two byte field in big endian format containing the length of the Check Data
Command, in bytes, including the header
Address: source address to test
Mask: bit mask to test
Count: optional poll count. If count is not specified this command will poll indefinitely
until the exit condition is met. If count = 0, this command behaves as for NOP.
```

The Parameter field is a single byte divided into bitfields as follows:

Table 8-33. Check Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

This command polls the source address until either the exit condition is satisfied, or the poll count is reached. The exit condition is determined by the flags as follows:

Table 8-34. Interpretation of Check Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	(*address & mask) == 0	All bits clear
0	1	(*address & mask) == mask	All bits set
1	0	(*address & mask)!= mask	Any bit clear
1	1	(*address & mask)!= 0	Any bit set

NOTE

If the source address does not have the same alignment as the data width indicated in the parameter field, the value is not read.

If the bitmask is wider than permitted by the data width indicated in the parameter field, the value is not read.

8.7.2.3 NOP Command

This command has no effect.

The format of NOP Command is a big endian four byte array as shown in the table below.

Table 8-35. NOP Command Format

Tag	Length	Undefined

where:

```
Tag: A single byte field set to 0xC0
Length: A two byte field in big endian containing the length of the NOP Command in bytes.
Fixed to a
value of 4.
Undefined: This byte is ignored and can be set to any value.
```

8.7.2.4 Unlock Command

The Unlock Command is used to prevent specific engine features being locked when exiting ROM.

The format of Unlock Command, again a big endian byte array, is shown in the table below.

Table 8-36. Unlock Command Format

Tag	Length	Eng
Value		
Value		
...		
Value		

where:

```
Tag: A single byte field set to 0xB2
Eng: Engine to be left unlocked.
Values: [optional] unlock values required by engine.
```

NOTE

This command may not be used in DCD structure if the SEC_CONFIG is configured as closed.

8.8 Plugin Image

The ROM supports a limited number of boot devices. For using other devices as boot source (for example, Ethernet, CDROM, or USB), the supported boot device must be used (typically serial ROM) for firmware with the missing boot drivers. >Additionally plugin can customize supported boot drivers. It is more flexible when doing device initialization, such as condition judging, delay assertion, applying custom settings to boot device and memory system.

>In addition to standard images, the chip also supports plugin images. Plugin images return execution to the ROM whereas a standard image does not.

The boot ROM detects the image type using the plugin flag of the boot data structure (see Boot Data Structure). If the plugin flag is 1, then the ROM uses the image as a plugin function. The function must initialize the boot device and copy the program image to the final location. At the end the plugin function must return with the program image parameters. (See High level boot sequence for details about boot flow).

The boot ROM authenticates the plugin image prior to running the plugin function and then authenticates the program image.

The plugin function must follow the API described below:

typedef BOOLEAN (*plugin_download_f)(void **start, size_t *bytes, UINT32 *ivt_offset);

ARGUMENTS PASSED:

- start - Image load address on exit.
- bytes - Image size on exit.
- ivt_offset - Offset in bytes of the IVT from the image start address on exit.

RETURN VALUE:

- 1 - on success
- 0 - on failure

8.9 Serial Downloader

The Serial Downloader provides a means to download a Program Image to the chip over USB and UART serial connection.

In this mode the ROM programs WDOG1 for a time-out specified by fuse WDOG Time-out Select (See fusemap for details) if WDOG_ENABLE eFuse is 1 and continuously polls for USB and UART connection. If no activity is found on USB OTG1, UART 1/2 and the watchdog timer expires, the ARM core is reset.

NOTE

After being loaded, the downloaded image is responsible to manage the watchdog resets properly.

Figure 8-2 shows USB and UART boot flow.

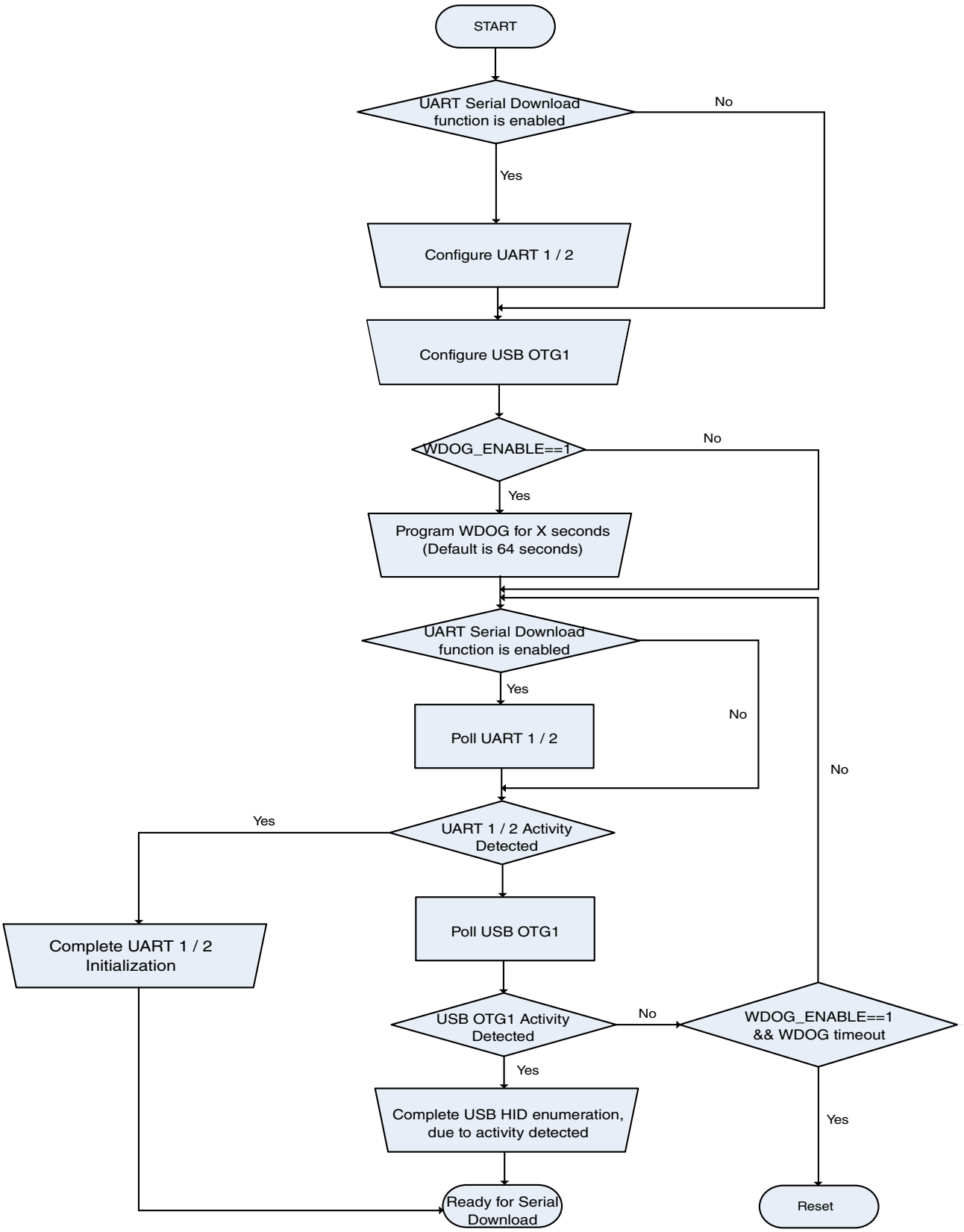


Figure 8-25. Serial Download Boot Flow

8.9.1 USB

USB support is composed of the USBO2 (USB OTG core controller, compliant with the USB 2.0 specification) and the USBPHY (HS USB transceiver).

The ROM supports the USB OTG port for boot purposes. The other USB ports on the chip are not supported for boot purposes.

The USB Driver is implemented as a USB HID class. A collection of 4 HID reports are used to implement SDP protocol for data transfers as described in Table 8-37.

Table 8-37. USB HID Reports

Report ID (first byte)	Transfer Endpoint	Direction	Length	Description
1	control OUT	Host to device	17 bytes	SDP command from host to device
2	control OUT	Host to device	Up to 1025 bytes	Data associated with report 1 SDP command
3	interrupt	Device to host	5 bytes	HAB security configuration. Device sends 0x12343412 in closed mode and 0x56787856 in open mode.
4	interrupt	Device to host	Up to 65 bytes	Data in response to SDP command in report 1

8.9.1.1 USB Configuration Details

The USB OTG function device driver supports a high speed (HS for UTMI) non-stream mode with a maximal packet size of 512 B and a low-level USB OTG function.

The VID/PID and strings for USB device driver are listed in the table below.

Table 8-38. VID/PID and Strings for USB Device Driver

Descriptor	Value
VID	0x15A2 (Freescale vendor ID)
PID[1]	0x007D
String Descriptor1 (manufacturer)	Freescale Semiconductor, Inc.
String Descriptor2 (product)	S Blank SE Blank

Table continues on the next page...

Table 8-38. VID/PID and Strings for USB Device Driver (continued)

Descriptor	Value
	NS Blank
String Descriptor4	Freescale Flash
String Descriptor5	Freescale Flash

1. Allocation based on BPN (Before Part Number)

8.9.1.2 IOMUX Configuration for USB

The interface signals of the UTMI PHY are not configured in the IOMUX. The UTMI PHY interface uses dedicated contacts on the IC. See the Chip data sheet for details.

8.9.2 UART

ROM supports UART1 and UART2 port for boot purpose, others UART ports on the chip are not supported for boot purpose.

If the event of data ready in UART FIFO is detected, ROM will enter UART serial download boot mode, and start to read data from the UART port which reports data ready event. The noise on UART RX data pad may cause a “data ready” event reported by UART controller, and further cause ROM entering UART serial download mode. Hardware designer should take care to avoid such exception.

8.9.2.1 UART Configuration Details

Table 8-39. UART Configuration Details

Transmission Protocol	RS232
Start bit	1 bit
Stop bit	1 bit
Parity bit	No needed
Flow control	No needed
Baud Rate	115200 bps

8.9.2.2 UART eFUSE Configuration

Table 8-40. UART eFUSE Configuration

Fuse	Config	Definition	GPIO	Shipped Value	Settings
0x470[4]	OEM	UART Serial Download Disabled Selection	Yes	0	0 - UART serial download enabled 1 - disabled

8.9.2.3 IOMUX Configuration for UART

Table 8-41. IOMUX Configuration for UART

Signal	UART1	UART2
TXD	UART1_TX_DATA.alt0	UART2_TX_DATA.alt0
RXD	UART1_RX_DATA.alt0	UART2_RX_DATA.alt0

8.9.3 Serial Download protocol

The 16 byte SDP command from host to device is sent using HID report 1.

The table below describes 16 byte SDP command data structure:

Table 8-42. 16 Byte SDP Command Data Structure

BYTE Offset	Size	Name	Description
0	2	COMMAND TYPE	The following commands are supported for ROM: • 0x0101 READ_REGISTER • 0x0202 WRITE_REGISTER • 0x0404 WRITE_FILE • 0x0505 ERROR_STATUS • 0x0A0A DCD_WRITE • 0x0B0B JUMP_ADDRESS • 0x0C0C SKIP_DCD_HEADER
2	4	ADDRESS	Only relevant for following commands: READ_REGISTER, WRITE_REGISTER, WRITE_FILE, DCD_WRITE, and JUMP_ADDRESS. For READ_REGISTER and WRITE_REGISTER commands, this field is address to a register. For WRITE_FILE and JUMP_ADDRESS commands, this field is an address to internal or external memory address.

Table continues on the next page...

Table 8-42. 16 Byte SDP Command Data Structure (continued)

BYTE Offset	Size	Name	Description
6	1	FORMAT	Format of access, 0x8 for 8-bit access, 0x10 for 16-bit and 0x20 for 32-bit access. Only relevant for READ_REGISTER and WRITE_REGISTER commands.
7	4	DATA COUNT	Size of data to read or write. Only relevant for WRITE_FILE, READ_REGISTER, WRITE_REGISTER and DCD_WRITE commands. For WRITE_FILE and DCD_WRITE commands DATA COUNT is in byte units.
11	4	DATA	Value to write. Only relevant for WRITE_REGISTER command.
15	1	RESERVED	Reserved

8.9.3.1 SDP Command

SDP commands are decribed in the following sections.

8.9.3.1.1 READ REGISTER

The transaction for command READ_REGISTER consists of following reports: Report1 for command, Report3 for security configuration and Report4 for response or register value.

The register to read is specified in ADDRESS field of SDP command. First device sends Report3 with security configuration followed by Report4 with bytes read at given address. If count is greater than 64 then multiple reports with report id 4 are sent until entire data requested by host is sent. The STATUS is either 0x12343412 for closed parts and 0x56787856 for open or field return parts.

Report1, Command, Host to Device:

1	Valid values for READ_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host: first response report

4	Register Value

ID 4 bytes of data containing register value. If number of bytes requested is less than 4 then remaining bytes should be ignored by host.

Multiple reports of report id 4 are sent until entire data requested is sent

Report4, Response, Device to Host: Last response report

4	Register Value

ID 64 bytes of data containing register value. If number of bytes requested is less than 64 then remaining bytes should be ignored by host.

8.9.3.1.2 WRITE REGISTER

The transaction for command WRITE_REGISTER consists of the following reports: Report1 for command, Report3 for security configuration and Report4 for write status.

Host sends Report1 with WRITE_REGISTER command. The register to write is specified in ADDRESS field of SDP command of Report1, with FORMAT field set to data type (number of bits to write 8, 16 or 32) and value to write in DATA field of SDP command. Device writes the DATA to register address and returns WRITE_COMPLETE code using Report4 and security configuration using Report3 to complete the transaction.

Report1, Command, Host to Device:

1	Valid values for WRITE_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT and DATA

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes data with first 4 bytes to indicate write is completed with code 0x128A8A12. On failure device will report HAB error status.

8.9.3.1.3 WRITE_FILE

The transaction for command WRITE_FILE consists of following reports: Report1 for command-phase, Report2 for data-phase, Report3 for hab mode and Report4 to indicate data received in full.

The size of each Report2 is limited to 1024 bytes (limitation of USB HID protocol) hence multiple Report2 packets will be sent by host in data phase until entire data is transferred to device. Once entire data (DATA_COUNT bytes) is received then device sends report 3 with hab mode and report 4 with 0x88888888, indicating file download completed.

Report1, Host to Device:

1	Valid values for WRITE_FILE COMMAND, ADDRESS, DATA_COUNT

ID 16 byte SDP Command

==========Optional Begin==========

Host sends ERROR_STATUS command to query if HAB rejected the address

========== Optional End==========

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report3, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	COMPLETE (0x88888888) status

ID 64 bytes data with first 4 bytes to indicate file download has completed with code 0x88888888. On failure device will report HAB error status.

8.9.3.1.4 ERROR_STATUS

The transaction for SDP command ERROR_STATUS consists of three reports.

Report1 is used by host to send the command; device sends global error status in 4 bytes of Report4 after returning security configuration in Report3. When device receives ERROR_STATUS command it will return global error status that is updated for each command. This command is useful to find out if last command resulted in device error or succeeded.

Report1, Command, Host to Device:

1	ERROR_STATUS COMMAND

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	4 bytes Error status

ID first 4 bytes status in 64 bytes report 4

8.9.3.1.5 DCD WRITE

The SDP command DCD_WRITE is used by host to send multiple register writes in one shot. This command is provided to speed up the process of programming register writes such as to configure external RAM device.

The command goes with Report1 from host with COMMAND TYPE set to DCD_WRITE, ADDRESS which is used for temporary location of DCD data and DATA_COUNT to number of bytes sent in data out phase. In data phase host sends data

for number of registers using Report2. Device completes the transaction with Report3 indicating security configuration and report 4 with WRITE_COMPLETE code 0x12828212.

Report1, Command, Host to Device:

1	DCD_WRITE COMMAND, ADDRESS, DATA_COUNT

ID 16 byte SDP Command

Report2, Data, Host to Device:

2	DCD binary data

ID Max 1024 bytes per report

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes report with first 4 bytes to indicate write is completed with code 0x128A8A12. On failure device will report HAB error status.

See Device Configuration Data (DCD) for DCD format description.

8.9.3.1.6 SKIP_DCD_HEADER

The SDP command SKIP_DCD_HEADER is used by host to inform device skip the DCD configuration within download image.

Normally, if the download image should be run on DDR, the DCD configuration data should be built-in the image. In case that host has issued DCD_WRITE to push DCD configuration data to device for DDR initialization, the image with DCD information will cause ROM initializes DDR twice and may cause initialization processing hung.

The SKIP_DCD_HEADER command inform device to skip the DCD configuration with the download image, and avoid this issue.

This command should typically be sent after JUMP_ADDRESS. The command is sent by host in command-phase of transaction using Report1, there is no data phase for this command. Device completes the transaction with Report3 indicating security configuration and report 4 with OK_ACK code 0x900DD009.

Report1, Command, Host to Device:

1	SKIP_DCD_HEADER

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	OK_ACK (0x900DD009)

8.9.3.1.7 JUMP ADDRESS

The SDP command JUMP_ADDRESS will be the last command host can send to the device, after this command device will jump to the address specified in the ADDRESS field of SDP command and start executing.

This command should typically follow after WRITE_FILE command. The command is sent by host in command-phase of transaction using Report1, there is no data phase for this command but device send status report3 to complete the transaction. And if HAB authentication fails then it will also send report 4 with HAB error status.

Report1, Command, Host to Device:

1	JUMP_ADDRESS COMMAND, ADDRESS

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

This report is sent by device only in case of an error jumping to the given address, device reports error in Report4, Response, Device to Host:

4	4 bytes HAB error status

ID 4 bytes status, 64 bytes report length

8.10 Recovery Devices

The Chip supports recovery devices. If primary boot device fails, boot ROM will try to boot from recovery device using one of ECSPI ports.

For enabling recovery device BOOT_CFG4[6] fuse must be set. Additionally Serial EEPROM fuses must be set as described in Serial ROM through SPI.

8.11 USB Low Power Boot

ROM supports USB Low Power Boot. This feature enables a device with dead or weak battery to power up and boot if the device is connected to a USB upstream port, no matter the upstream port is a USB charger or USB host/hub.

If a USB dedicated charger or host/hub charger are connected, as soon as the device is connected to the upstream port, a stable current (Max.1.5A) can be supplied by charger. If USB host/hub are connected, the maximal 1 00mA current is supplied to the device, the device should be able to power up to boot the image with less than 100mA.

If LPB_BOOT fuses are blown, the Chip will check if there is low power condition via UART3_CTS (as GPIO1_26) pad. If there is low power boot condition USB charger detection will be activated. If there is no USB charger, ROM will initialize USB as device and apply division factors on ARM, DDR, AXI and AHB root clocks based on LPB_BOOT fuses value (see the table below). Polarity of low power boot condition on UART3_CTS (as GPIO1_26) pad is set by BT _LPB_POLARITY fuse (see the figure below).

Table 8-43. USB Low Power Bus Boot Frequencies

LPB_BOOT	Boot Frequencies=0	Boot Frequencies=1
0	MMDC_CLK_ROOT = 396 MHz FABRIC_CLK_ROOT = 198 MHz AHB_CLK_ROOT = 132 MHz	MMDC_CLK_ROOT = 307 MHz FABRIC_CLK_ROOT = 153 MHz AHB_CLK_ROOT = 102 MHz

Table continues on the next page...

Table 8-43. USB Low Power Bus Boot Frequencies (continued)

LPB_BOOT	Boot Frequencies=0	Boot Frequencies=1
1	MMDC_CLK_ROOT = 396 MHz FABRIC_CLK_ROOT = 198 MHz AHB_CLK_ROOT = 132 MHz	MMDC_CLK_ROOT = 307 MHz FABRIC_CLK_ROOT = 153 MHz AHB_CLK_ROOT = 102 MHz
10	MMDC_CLK_ROOT = 198 MHz FABRIC_CLK_ROOT = 99 MHz AHB_CLK_ROOT = 66 MHz	MMDC_CLK_ROOT = 153 MHz FABRIC_CLK_ROOT = 76 MHz AHB_CLK_ROOT = 51 MHz
11	Reserved	Reserved

Table 8-44. USB Low Power CPU Boot Frequencies

LPB_BOOT	Boot Frequencies=0	Boot Frequencies=1
0	ARM_CLK_ROOT = 396 MHz	ARM_CLK_ROOT = 198 MHz
1	ARM_CLK_ROOT = 396 MHz	ARM_CLK_ROOT = 198 MHz
10	ARM_CLK_ROOT = 198 MHz	ARM_CLK_ROOT = 99 MHz
11	Reserved	Reserved

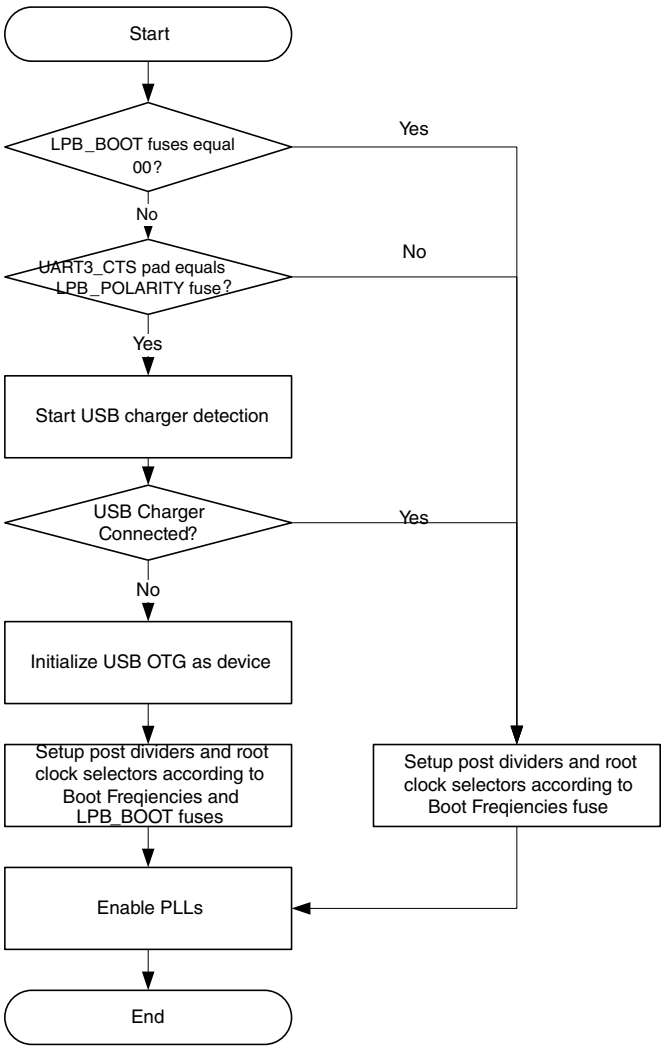


Figure 8-26. USB Low Power Boot Flow

8.12 SD/MMC Manufacture Mode

When internal boot and recover boot (if enabled) failed, SDMMC_MFG_DISABLE fuse bit isn't set, and EEPROM Recovery fuse bit is set, boot will go to SD/MMC manufacture mode before serial download mode. In manufacture mode, one bit bus width is used despite of the fuse setting.

In manufacture mode, SD or MMC card will be scanned on uSDHC1. If card is detected and valid boot image is found in card, then boot image will be loaded then executed. Pad of SD1_CD is used to detect whether card is inserted.

By default, SD/MMC manufacture mode is enabled, blow the fuse of DISABLE_SDMMC_MFG to disable it.

NOTE

Secondary boot is not supported on SD/MMC manufacture mode.

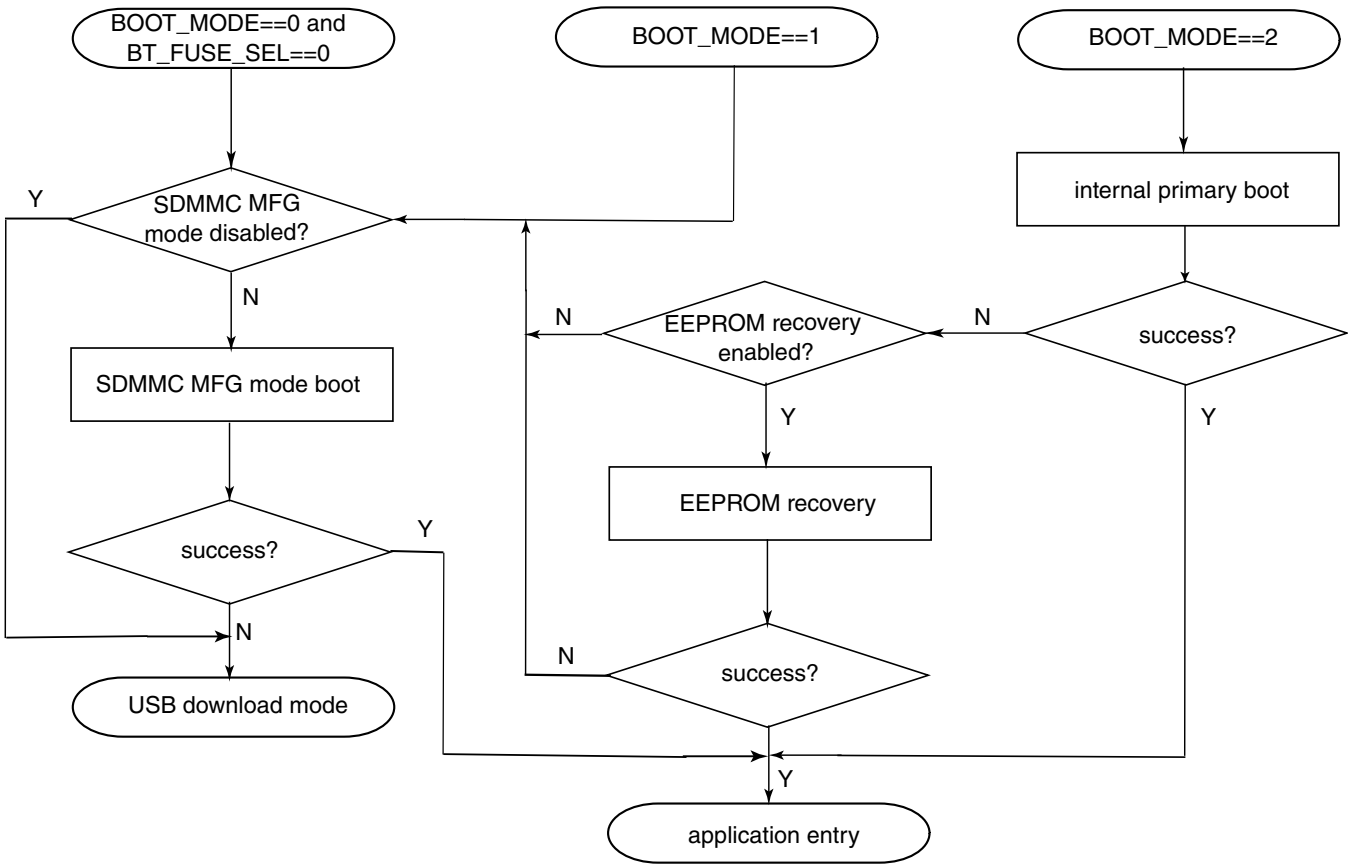


Figure 8-27. SD/MMC Manufacture boot flow

8.13 High Assurance Boot (HAB)

The High Assurance Boot (HAB) component of the ROM protects against the potential threat of attackers modifying areas of code or data in programmable memory to make it behave in an incorrect manner. The HAB also prevents attempts to gain access to features which should not be available.

The integration of the HAB feature with the ROM code ensures that Chip does not enter an operational state if the existing hardware security blocks have detected a condition that may be a security threat or areas of memory deemed to be important have been modified. The HAB uses RSA digital signatures to enforce these policies.

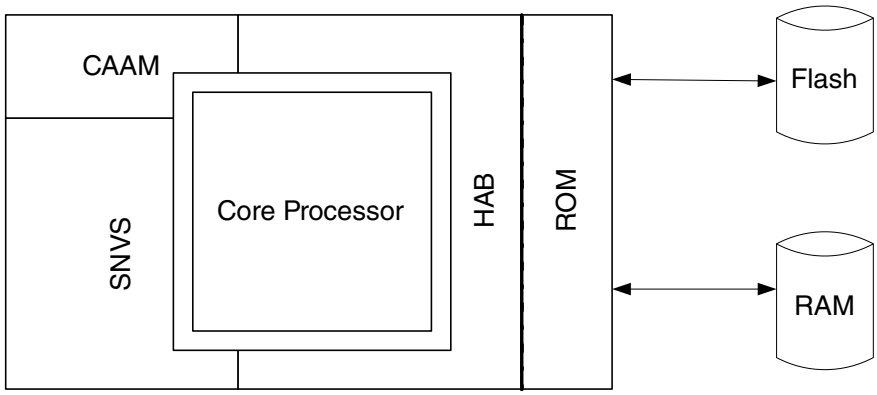


Figure 8-28. Secure Boot Components

The figure above illustrates the components used during a secure boot using HAB. The HAB interfaces with the SNVS to ensure the system security state is as expected. The HAB also makes use of CAAM hardware block to accelerate SHA-256 message digest operations performed during signature verifications and AES-128 operations for encrypted boot operations. The HAB also includes a software implementation of SHA-256 for cases where a hardware accelerator cannot be used. The RSA key sizes supported are 1024, 2048 and 3072 bits. The RSA signature verification operations are performed by a software implementation contained in the HAB library. The main features supported by HAB are:

- X.509 Public key certificate support
- CMS signature format support
- Proprietary encrypted boot support. Note that encrypted boot depends on the CAAM HW module. When CAAM is disabled (i.e. when the EXPORT_CONTROL fuse is blown) then encrypted boot is not available.

NOTE

Freescale provides a reference Code Signing Tool (CST) for key generation, certificate generation and code signing for use with the HAB library. The CST can be found by searching for "IMX_CST_TOOL" at http://www.freescale.com.

NOTE

For further details on making use of the secure boot feature using HAB contact your local Freescale representative.

8.13.1 HAB API Vector Table Addresses

For devices that perform a secure boot, the HAB library may be called by boot stages that execute after ROM code.

The RVT table contains the pointers to the HAB API functions and is located at 0x00000100.

NOTE

For additional information on secure boot including the HAB API, contact your local Freescale representative.

Chapter 9
Multimedia

9.1 Display and graphics subsystem

The chip display and graphics subsystem consists of the dedicated modules found here.

- LCD Interface (LCDIF): 24-bit parallel RGB LCD interface
- PXP pixel pipeline: pixel/image processing engine for LCD display
- CSI (camera sensor interface): up to 24-bit parallel interface for image sensor

The following figure shows the high-level integration scheme of the chip display and graphics system.

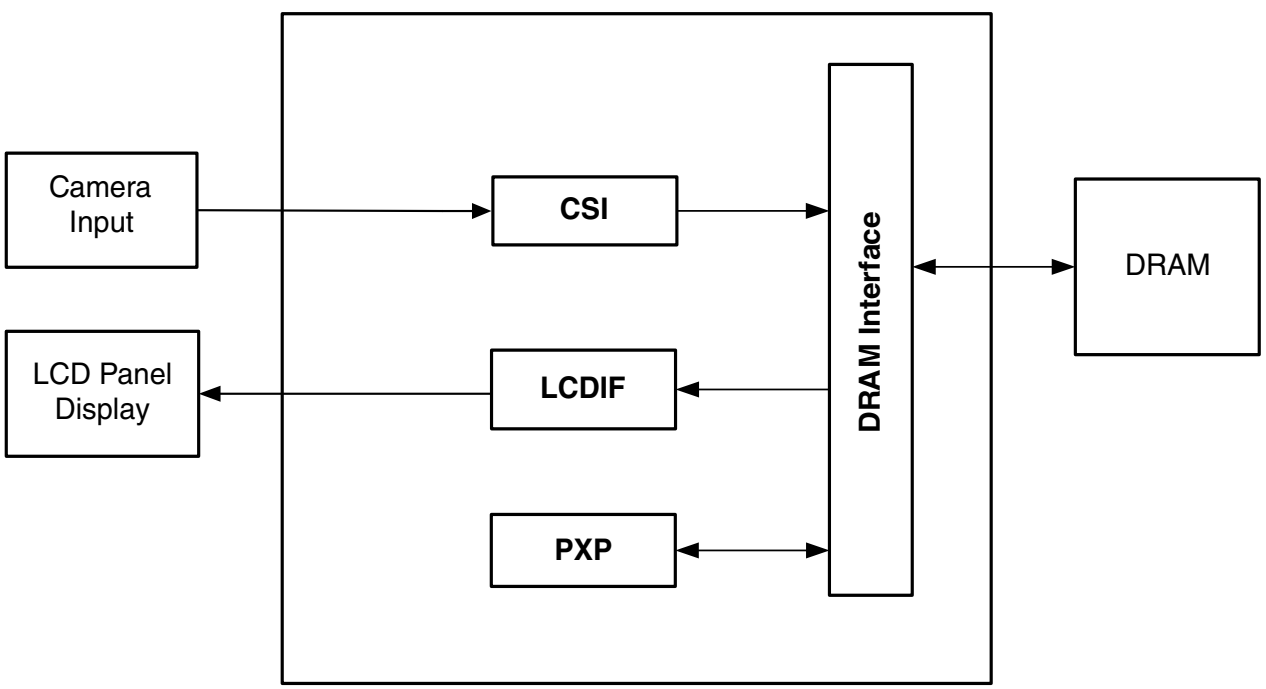


Figure 9-1. Chip display and graphics subsystem

9.1.1 PiXel Processing Pipeline

The pixel processing pipeline is used to perform image processing on image/video buffers before sending to an LCD display.

The main features of PXP include:

- Multiple input/output format support, including YUV/RGB/Grayscale
- Supports both RGB/YUV scaling
- Supports overlay with Alpha blending

9.1.2 LCD Interface

The LCDIF is a general purpose display controller that is used to drive a wide range of display devices. These displays can vary in size and capability. Many of these displays have had an asynchronous parallel MPU interface for command and data transfer to an integrated frame buffer. There are other popular displays that support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) or the VSYNC mode for high-speed data transfers. In addition to these displays, it is also common to provide support for digital video encoders that accept ITU-R BT.656 format 4:2:2 YCbCr digital component video and convert it to analog TV signals. The LCDIF block supports these different interfaces by providing fully programmable functionality.

The block has several major features:

- Bus master interface to source frame buffer data for display refresh and a DMA interface to manage input data transfers from the LCD requiring minimal CPU overhead.
- 8/16/18/24 bit LCD data bus support available depending on I/O mux options.
- Programmable timing and parameters for MPU, VSYNC and DOTCLK LCD interfaces to support a wide variety of displays.
- ITU-R BT.656 mode (called Digital Video Interface or DVI mode here) including progressive-to-interlace feature and RGB to YCbCr 4:2:2 color space conversion to support 525/60 and 625/50 operation.

9.1.3 Parallel CMOS Sensor Interface

The CSI enables the chip to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only Bayer and statistics data, while smart sensors support CCIR656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

The capabilities of the CSI include:

- Configurable interface logic to support most commonly available CMOS sensors.
- Support for CCIR656 video interface as well as traditional sensor interface.
- 8-bit data port for YCC, YUV, or RGB data input.
- 8-bit/10-bit/16-bit data port for Bayer data input.
- Full control of 8-bit/pixel, 10-bit/pixel or 16-bit/pixel data format to 64-bit receive FIFO packing.
- 256 × 64 FIFO to store received image pixel data. Receive FIFO overrun protection mechanism.
- Embedded DMA controllers to transfer data from receive FIFO or statistic FIFO through AHB bus.
- Support for 2D DMA transfer from the receive FIFO to the frame buffers in the external memory.
- Support for double buffering two frames in the external memory.
- Single interrupt source to interrupt controller from maskable interrupt sources:
  - Start of Frame
  - End of Frame
  - Change of Field
  - FIFO full
  - FIFO overrun
  - DMA transfer done
  - CCIR error
  - AHB bus response error
- Configurable master clock frequency output to sensor.
- Statistic data generation for Auto Exposure (AE) and Auto White Balance (AWB) control of the camera (only for Bayer data and 8-bit/pixel format).

9.2 Audio subsystem

Audio Subsystem Module Overview provides an overview of each of the audio subsystem component modules, followed by a module-specific section.

9.2.1 Audio Subsystem Module Overview

The following figure shows a high level block diagram of the audio subsystem.

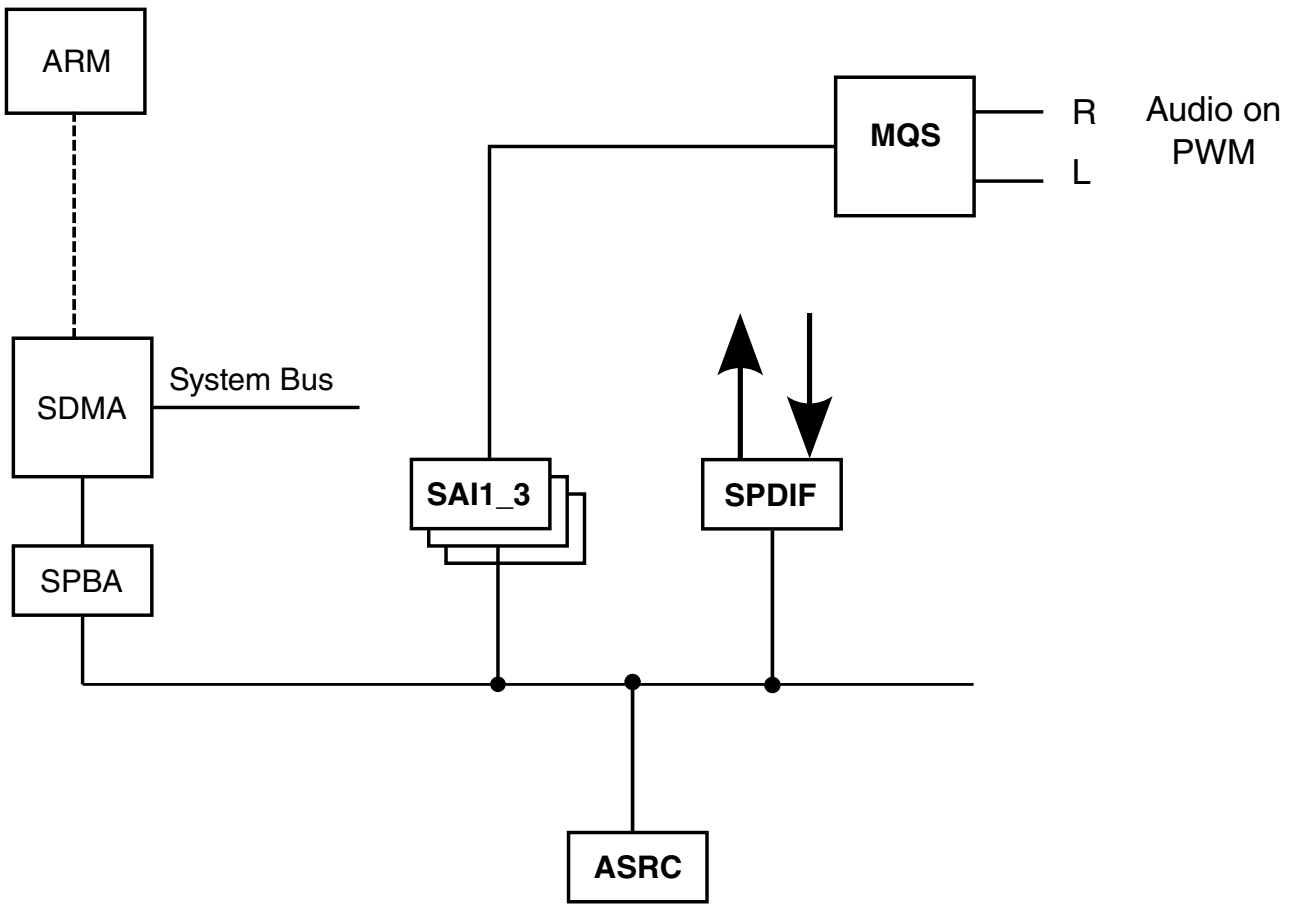


Figure 9-2. Audio subsystem block diagram

The SPDIF (Sony/Philips digital interface) audio module is a stereo transceiver that allows the processor to receive and transmit digital audio over it. The SPDIF receiver section includes a frequency measurement block that allows the precise measurement of an incoming sampling frequency. A recovered clock is provided by the SPDIF receiver section and may be used to drive both internal and external components in the system. SPDIF is connected to the shared peripheral bus.

ASRC (asynchronous sample rate converter) converts the sampling rate of a signal associated to an input clock into a signal associated to a different output clock. ASRC supports concurrent sample rate conversion of up to 10 channels of over 120dB THD+N.

The sample rate conversion of each channel is associated to a pair of incoming and outgoing sampling rates. ASRC supports up to 3 sampling rate pairs. ASRC is connected to the shared peripheral bus.

9.2.2 Medium Quality Sound (MQS)

MQS is used to generate medium quality audio via a standard GPIO in the pinmux. The user can connect stereo speakers or headphones to a power amplifier without an additional DAC chip.

- 2-channel, MSB-valid 16 bit, MSB first
- Frame sync aligned with the left channel data
- 44.1 kHZ or 48 kHZ I2S signals from SAI1
- SNR target as no more than 20 dB for the signals below 10 kHZ
- Signals Over 10 kHZ have worse THD + N values

9.2.3 Synchronous Audio Interface (SAI)

- Transmitter with independent Bit Clock and Frame Sync supporting 1 data line
- Receiver with independent Bit Clock and Frame Sync supporting 1 data line
- Maximum Frame Size of 32 Words
- Word size programmable from 8-bits to 32-bits
- Word size configured separately for first word and remaining words in frame.
- Asynchronous FIFO for each Transmit and Receive data line
- Graceful restart after FIFO Error

9.2.4 Sony/Philips Digital Interface (SPDIF)

The Sony/Philips Digital Interface (SPDIF) module is a stereo that allows the processor transmit digital audio over it using the IEC60958 standard, consumer format. The chip provides one SPDIF transmitter with one output and one SPDIF receiver with four inputs.

The SPDIF allows the handling of both SPDIF channel status (CS) and User (U) data.

For the SPDIF transmitter, the audio data is provided by the processor via the SPDIFTxLeft and SPDIFTxRight registers, and the data is stored in two 16-word-deep FIFOs, one for the right channel, the other for the left channel. The FIFOs support programmable watermark levels so that FIFO Empty service request can be triggered when the combined number of empty data words locations in both FIFOs is 8, 16, 24 or

32 words. It is recommended to program the watermark level to trigger a FIFO Empty service request when 16 word locations are empty. For optimal performance when servicing the FIFO Empty service request, the FIFOs should be written alternately, starting with the left channel FIFO. The Channel Status bits are also provided via the corresponding registers. The SPDIF transmitter generates an SPDIF output bitstream in the biphase mark format (IEC 60958), which consists of audio data, channel status and user bits.

The data handled by the SPDIF module is 24-bit wide. The 24-bit SPDIF data is aligned in the 24 least significant bits of the 32-bit shared peripheral bus data word. The 8 most significant bits of the 32-bit word are ignored by the SPDIF Transmitter when data is being stored in the Transmit FIFOs from the peripheral bus. The 8 most significant bits of the 32-bit word are zeroed by the SPDIF Receiver module when the data is being read from the Receiver FIFOs to the peripheral bus.

Note that 16-bit data is left-aligned in the 24-bit word format of the SPDIF. When 16-bit data is to be transmitted, the 32-bit word to be written to the SPDIF Transmit FIFOs should be created as follows: the 16-bit data should be located in the middle two bytes of the 32-bit data word and the 8 bits of the LSB must be set to zero, while the 8 bits of the MSB will be ignored.

The SPDIF Transmit clock is generated by the SPDIF internal clock generator module and the clock sources are from outside of the SPDIF block. The clock sources should provide a clock that is at least 64 x Fs, where Fs is the sampling frequency. The external clock source should provide at least 128 x Fs. Clocks of higher frequency may be provided as long as the multiplication factor is a power of 2 (for example, 128x, 256x or 512x). Also, clock frequency precision of 100ppm or better should be provided.

Figure 9-3. SPDIF Transceiver Clock Diagram

9.2.5 Asynchronous Sample Rate Converter (ASRC)

The incoming audio data may be received from various sources at different sampling rates. The outgoing audio data may have different sampling rates, and it can also be associated to output clocks that are asynchronous to the input clocks.

ASRC converts the sampling rate of a signal associated to an input clock into a signal associated to a different output clock. ASRC supports concurrent sample rate conversion of up to 10 channels of about -120dB THD+N. The sampling rate conversion of each channel is associated to a pair of incoming and outgoing sampling rates. ASRC supports up to 3 sampling rate pairs.

In the real-time audio use case, both input/output sampling rate clocks are activated. Both sampling rate clocks are directly connected to ASRC, and the ratio estimation of the input clocks to output clocks is used to perform the sample rate conversion in ASRC hardware.

The SAI1, SAI2, SAI3, and SPDIF audio modules can act as either a clock master or a clock slave. Multiplexors are provided on a chip level to select which of the module's clock signal, input or output, will be connected to ASRC input in dependence of the module's operational mode as it is shown in Figure 9-4 and Table 9-1. Figure 9-4 presents the multiplexor cell used for all ASRC input clocks. Table 9-1 shows the detailed clocks, multiplexor control and data bits for each ASRC input clock. For the audio blocks clocks that are directly conected to ASRC (without the multiplexor scheme) see Table 9-2.

In the non-real time streaming audio use case, the input sampling rate clock does not need to be provided. Instead the ideal-ratio value conversion is set in the ASRC interface registers. In this case only the output sampling rate clock must be provided, and the fixed ratio of the input to the output in the register is used to perform the sample rate conversion.

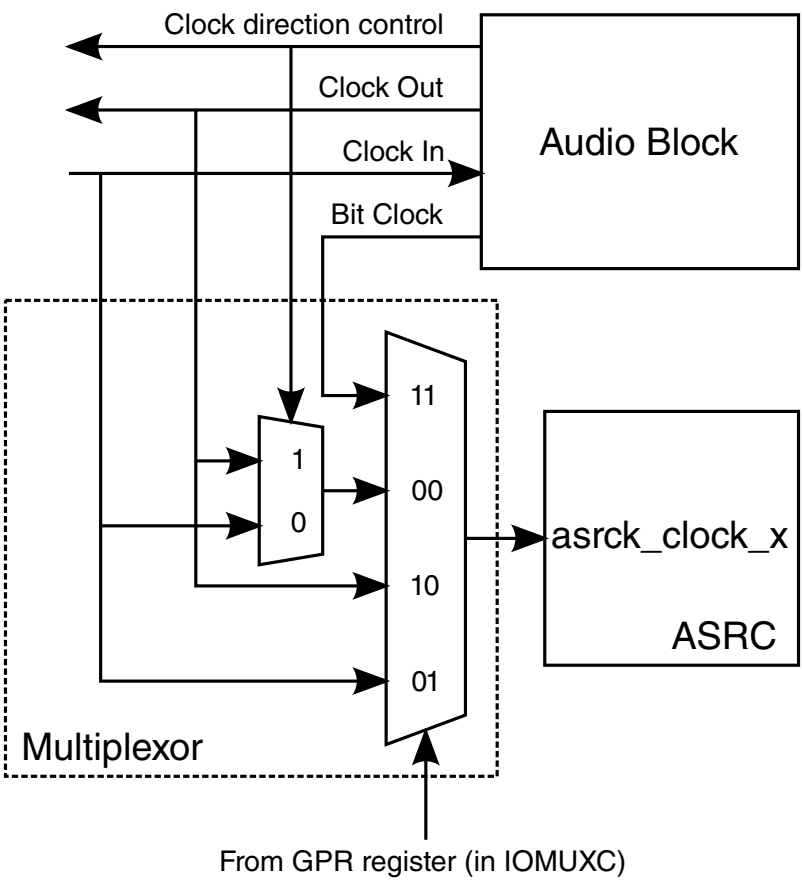


Figure 9-4. Muxing scheme template for ASRC input clocks

Table 9-1. ASRC Muxed Input Clocks

ASRC Clock Input	Audio block source	MUX2:1 Control	MUX2:1 Input 0	MUX2:1 Input 1	MUX4:1 Control	MUX4:1 Input 00	MUX4:1 Input 01	MUX4:1 Input 10	MUX4:1 Input 11
asrck_clock_1	SAI1-Rx	SRCR[RXDIR][1]	External SRCK	InternalSRCK	GPR0[17:16][2]	MUX2:1 output	External SRCK	InternalSRCK	RX bit clock
asrck_clock_4	SAI1-Rx	SRCR[RXDIR]	External SRCK	InternalSRCK	GPR0[17:16]	MUX2:1 output	InternalSRCK	InternalSRCK	RX bit clock
asrck_clock_7	SAI2-Rx	SRCR[RXDIR]	External SRCK	InternalSRCK	GPR0[19:18]	MUX2:1 output	External SRCK	InternalSRCK	RX bit clock
asrck_clock_9	SAI1-Tx	STCR[TXDIR]	External STCK	InternalSTCK	GPR0[19:18]	MUX2:1 output	External STCK	InternalSTCK	TX bit clock
asrck_clock_c	SAI1-Tx	STCR[TXDIR]	External STCK	InternalSTCK	GPR0[21:20]	MUX2:1 output	External STCK	InternalSTCK	TX bit clock
asrck_clock_d	spdif_clk_root	STCR[TXDIR]	External STCK	InternalSTCK	GPR0[21:20]	MUX2:1 output	External STCK	InternalSTCK	SPDIF bit clock
asrck_clock_e	SAI2-Tx	STCR[TXDIR]	External STCK	InternalSTCK	GPR0[23:22]	MUX2:1 output	External STCK	InternalSTCK	TX bit clock

1. Register[bit] format, SRCR is the regsiter name in the block specified in the "Audio block source" column, while RXDIR is name field/bit in the register.
2. GPR0 is the General Register 0 in the IOMUX controller. See IOMUXC chapter for more details.

Table 9-2. ASRC Direct Clocks

ASRC Clock Input	Block driving clock source	Block output clock
asrck_clock_4	SPDIF (Rx)	SPDIF Rx Clock
asrck_clock_c	SPDIF (Tx)	SPDIF Tx Clock
asrck_clock_6	External (from PAD)	Via IOMUX (from PAD) KEY_ROW3 (ALT1) [default] GPIO_0 (ALT3) GPIO_18 (ALT4)
asrck_clock_7	Reserved	Reserved[1]
asrck_clock_d	CCM	SPDIF1 clock root

1. Not in used, the clock value is "0".

Chapter 10
Clock and Power Management

10.1 Introduction

This chapter describes the Clock and Power Management architecture of the SoC.

The chip targets applications where low power consumption, long battery life, always-on and instant-on capabilities are paramount, and where there is no need for active cooling. To achieve these capabilities, the primary focus of the chip's design is reducing current consumption as much as possible, while simultaneously enabling the maximum level of peak performance and a balanced level of sustained performance for target applications. To achieve this, the chip architecture uses a wide range of power-management techniques and their combinations for maximum system design flexibility.

This chapter contains information about:

- Structural components of the power and clock management systems of the chip
- Power, clock and thermal management techniques supported by the chip

All given numerical values are typical or examples. For accurate values one should refer to the datasheet.

10.2 Device Power Management Architecture Components

To provide a clean and versatile architecture supporting a wide range of power-management techniques, clocks, and power rails are managed resources.

For each rail, two levels of management are defined: the first level is centralized or SoC-level resource management, and the second is a local or "module level" resource management.

The high level architectural view of the clock, power and thermal management system of the chip is presented in the figure below.

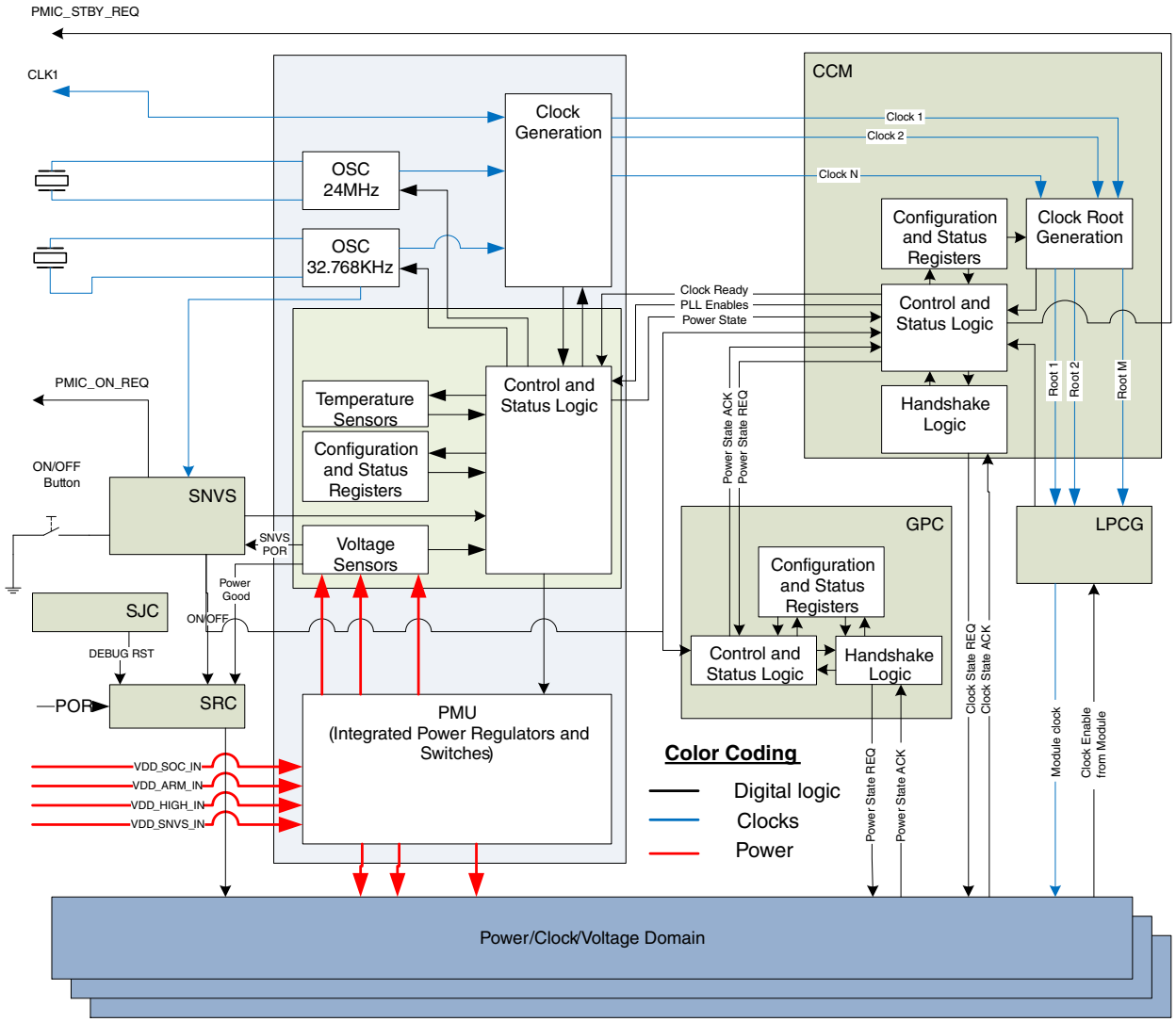


Figure 10-1. Power and clock management framework

10.2.1 Centralized components of clock generation and management

Centralized components of the clock generation and management sub-system are implemented in the following blocks:

- CCM (Clock Control Module): The CCM module provides control for primary (source level) and secondary (root level) clock generation, division, distribution, synchronization, and coarse-level gating. See Clock Controller Module (CCM)) for

information on the CCM architecture, functional description and programming model.
- LPCG (Low Power Clock Gating): This module distributes the clocks to all blocks in the SoC and handles block level software-controllable and automated clock gating. See Clock Controller Module (CCM) for information on the LPCG architecture and functional description.

10.2.2 Centralized components of power generation, distribution and management

Centralized components of the power generation, distribution and management sub-system are implemented in the following blocks:

- Power Management Unit (PMU). See Power Management Unit (PMU) for information on the PMU architecture, functional description and programming model.
- General Power Controller (GPC). See General Power Controller (GPC) for information on the GPC architecture, functional description and programming model.

10.2.3 Reset generation and distribution system

Power and clock management are accompanied with an appropriate reset generation and distribution system, centralized functions of which are implemented by the System Reset Controller (SRC). See General Power Controller (GPC) for information on the GPC architecture, functional description and programming model.

10.2.4 Power and clock management framework

Together, the modules listed above provide enhanced power-management features with centralized control for the clock, reset, and power-management signals on the SoC.

The centralized management framework defines the managed components of the power-management architecture. These components are called the clock, power, and voltage domains.

NOTE

A domain is a group of modules or functional blocks that share a common resource entity (for example, common clock root, common power source, or a common power switch). The software component managing shared resources should take

into account the joint constraints of all the modules belonging to that resource domain.

10.3 Clock Management

10.3.1 Centralized components of clock management system

The clock generation and management system is built around the CCM and LPCG blocks.

A high level block diagram of the clock management system in the SoC environment is shown in the following figure.

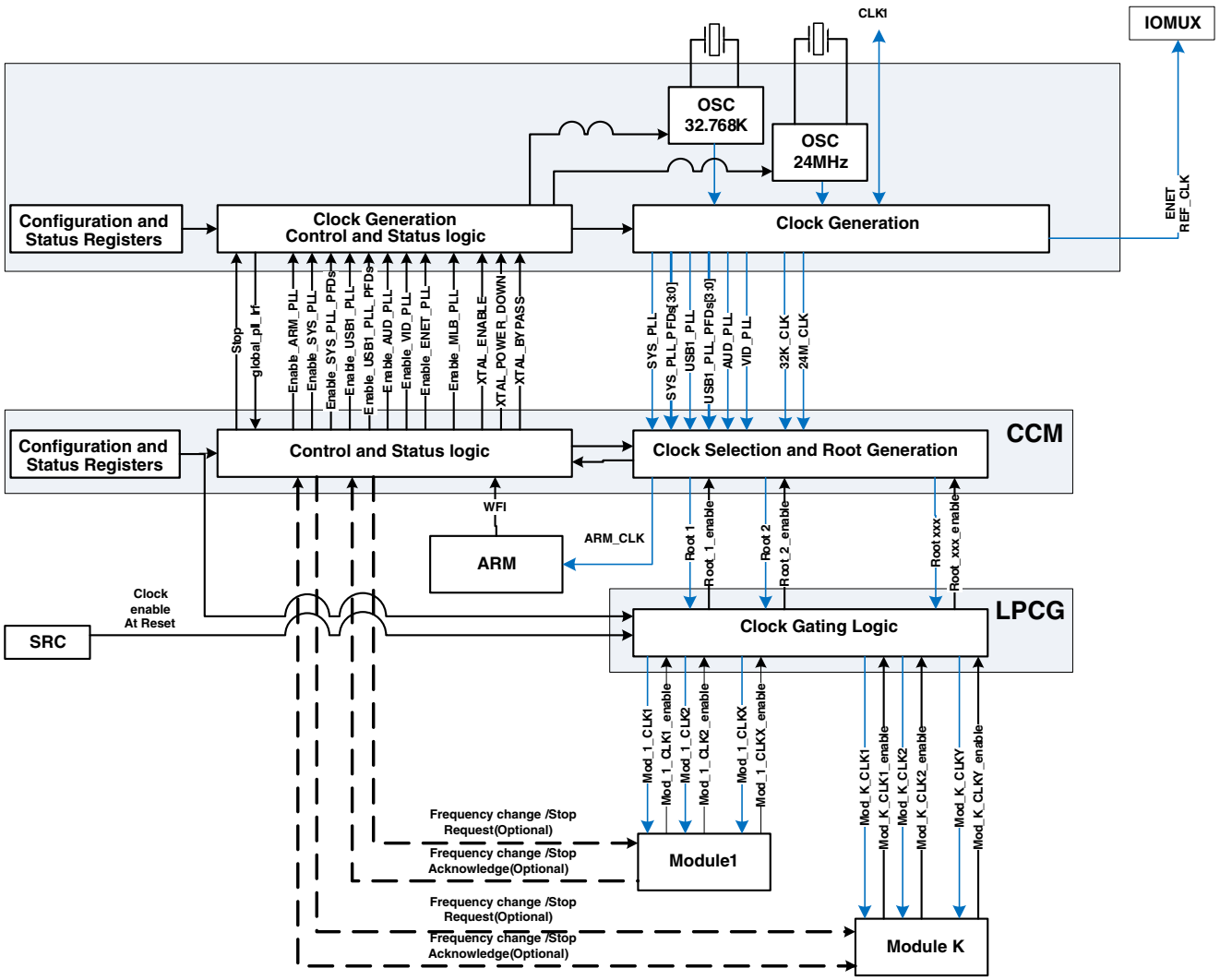


Figure 10-2. Clock Management System

A high level block diagram of the clock generation is shown in the figure below.

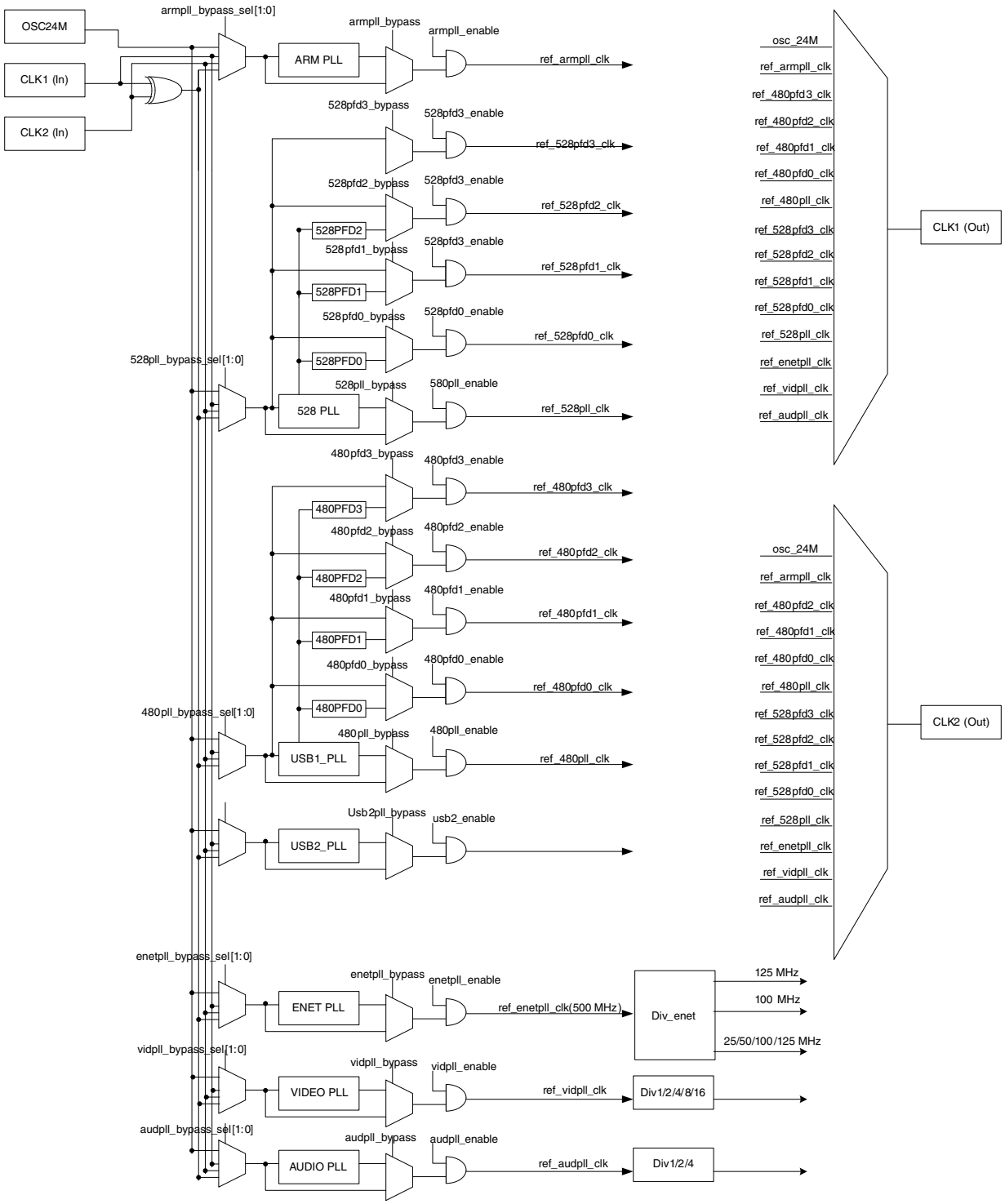


Figure 10-3. Primary Clock Generation

10.3.2 Clock generation

The clock generation section includes the components detailed in the following sections.

10.3.2.1 Crystal Oscillator (XTALOSC)

The Crystal Oscillator block is comprised of both the high frequency oscillator (typical frequency is 24 MHz) and the low frequency real time clock oscillator (typical frequency of 32.768 KHz). Each of these oscillators is implemented as a biased amplifier that, when combined with a suitable external quartz crystal and external load capacitors, implements an oscillator. See Crystal Oscillator (XTALOSC) for details of the XTALOSC block.

10.3.2.2 Low Voltage Differential Signaling (LVDS) I/O ports

There is one LVDS I/O port used for clock generation. The low jitter differential I/O port is provided to input and output clocks. It can take input clocks from outside of the SoC and provide them to the PLLs or to the other modules, or it can take the outputs of the PLLs and provide them outside of the SoC as a functional or reference clock.

10.3.2.3 PLLs

Seven PLLs are included in the clock generation section. Two of these PLLs are each equipped with four Phase Fractional Dividers (PFDs) in order to generate additional frequencies.

NOTE

Each PFD works independently by interpolating the VCO of the PLL to which it is connected. It effectively takes the PLL VCO frequency and produces 18/N x Fvco at its output where N ranges from 12 to 35. PFD is a completely digital design with no analog components or feedback loops. The frequency switch time is much faster than a PLL because keeping the base PLL locked and changing the integer N only changes the logical combination of the interpolated outputs of the VCO. Note that the PFD not only enables faster frequency changes than a PLL, but also allows the configuration to be safely changed "on-the-fly" without going through the output clock disabling/enabling process.

The seven PLLs are listed below:

- PLL1 (also referred to as ARM_PLL) - This is the PLL clocking the ARM core complex. It is a programmable integer frequency multiplier capable of output

frequency of up to 1.3 GHz. Note that this frequency is higher than the maximum chip supported frequency 1.0 GHz.

- PLL2 (also referred tp as System_PLL or 528_PLL) - PLL2 runs at a fixed multiplier of 22, producing 528 MHz output frequency with 24 MHz reference from XTALOSC. Besides the main output, this PLL drives four PFDs (PLL2_PFD0...PLL2_PFD3). The main PLL output and its PFD outputs are used as inputs for many clock roots. These do not require exact/constant frequency and can be changed as a part of dynamic frequency scaling procedure. Typically, this PLL or its PFDs are a clock source for internal system buses, internal processing logic, DDR interface, NAND/NOR interface modules, etc.
- PLL3 (also referred to as USB1_PLL) - PLL3 is used in conjunction with the first instance of USB PHY (USBPHY1, also known as OTG PHY). This PLL drives four PFDs (PLL3_PFD0...PLL3_PFD3) and runs at a fixed multiplier of 20. This results in a VCO frequency of 480 MHz with a 24 MHz oscillator. The main PLL output and its PFD outputs are used as inputs for many clock roots that require constant frequency, such as UART and other serial interfaces, audio interfaces, etc.
- PLL4 (also referred to as an Audio PLL) - This is a fractional multiplier PLL used for generating a low jitter and high precision audio clock with standardized audio frequencies. The PLLs oscillator frequency range is from 650 MHz to 1300 MHz, and the frequency resolution is better than 1 Hz. This clock is mainly used as a clock for serial audio interfaces and as a reference clock for external audio codecs. It is equipped with a divider on its output and can generate divided by 1, 2 or 4 from the PLL VCO frequency.
- PLL5 (also referred to as a Video PLL) - This is a fractional multiplier PLL used for generating a low jitter and high precision video clock with standardized video frequencies. The PLLs oscillator frequency range is from 650 MHz to 1300 MHz, and the frequency resolution is better than 1 Hz. This clock is mainly used as a clock for display and video interfaces. It is equipped with dividers on its output and can generate clock divided by 1, 2, 4, 8 or16 from the PLL VCO frequency
- PLL6 (also referred to as ENET_PLL) - This PLL implements a fixed 20+(5/6) multiplier. With a 24 MHz input, it has a VCO frequency of 500 MHz. This PLL is used to generate:
  - 50 or 25 MHz for the external ethernet interface.
  - 125 MHz for reduced gigabit ethernet interface.
  - 100 MHz for general purposes.
- PLL7 (also referred to as USB2_PLL) - This PLL provides clock exclusively to USB2 PHY (USBPHY2, also known as OTG PHY). It runs at a fixed multiplier of 20, resulting in a VCO frequency of 480 MHz with a 24 MHz oscillator.

10.3.2.3.1 General PLL Control and Status Functions

Each PLLs configuration and control functions are accessible individually through its PFDs and global configuration and status registers.

Reference input clock for any of the PLLs could be selected individually by the BYPASS_CLK_SRC field of the PLL control register. See CCM Analog Memory Map/ Register Definition for more information.

Each of the PLLs could be individually configured to "Bypass", "Output disabled" and "Power Down" modes.

When configured in "Bypass" PLL pass directly its input reference clocks to the PLL output. Bypassing the PLL is done by setting the BYPASS bit in the control register. For the PLL equipped with PFDs the input reference clock is also bypassed to all PFDs outputs.

When configured in output disabled mode (ENABLE=0), the PLL's output is completely gated and there is neither a bypass clock nor PLL generated clock that propagates to PLL output. Each PLL output has an individual "Output Enable" control bit. The PFDs are gated by the ENABLE bit of their associated PLL. Each PFD does have an associated clock gate bit that can be used to turn it off individually.

When configured in "Power Down mode" most of the PLL circuitry is switched off. Neither main PLL output nor PFD outputs are available in this mode.

When the related PLL is powered up from the power down state or made to go through a relock cycle due to PLL reprogramming, it is required that the related PFDx_CLKGATE bit in CCM_ANALOG_PFD_480n or CCM_ANALOG_PFD_528n, be cycled on and off (1 to 0) after PLL lock. The PFDs can be in the clock gated state during PLL relock but must be un-clock gated only after lock is achieved. See the engineering bulletin, Configuration of Phase Fractional Dividers (EB790) at www.freescale.com for procedure details.

Individual PLL status is reflected in "PLL Lock" bits of the PLL control registers. PLL enable logic which monitors the register value change is implemented to gate off the PLL outputs during the "lock in" period.

Outputs are generated to be sent out by monitoring the individual PLL lock flags and filtering out any random initial edges.

Individual PLL Lock ready flags are first "ORED" with "enables" and then "ANDED" together to generate the global PLL lock ready flag that reflects status of all PLLs enabled in certain moment.

CCM Memory Map/Register Definition and CCM Analog Memory Map/Register Definition contains detailed descriptions of the memory mapped registers and control functions of the clock generation sub-module.

10.3.2.4 CCM

CCM includes:

- Clock root generation logic - This sub-block provides the registers that control most of the secondary clock source programming, including both the primary clock source selection and the clock dividers. The clock roots are each individual clocks to the core, system buses (AXI, AHB, IPG) and all other SoC peripherals, among those are serial clocks, baud clocks, and special functional clocks. Most of clock roots are specific per module.
- CCM, in coordination with GPC, PMU and SRC, manages the Power modes, namely RUN, WAIT and STOP modes. The gating of the peripheral clocks is programmable in RUN and WAIT modes.

CCM manages the frequency scaling procedure for:

- ARM core clock - "on the fly" without clock interruption, by either shifting between PLL sources PLL clock change or by changing the divider ratio.
- Graceful changing of the DDR memory controller clock. See MMDC handshake and Self refresh and Frequency change entry/exit for more details.
- Peripheral root clock - by using programmable divider. The division factor can change on the fly without loss of clocks.

NOTE

On-the-fly frequency changing for synchronous interfaces like serial audio interfaces (SAI), Audio Sample Rate Converter (ASRC), or general purpose serial interfaces (UART, CAN) in general causes synchronization loss and should not be done.

10.3.2.5 Low Power Clock Gating unit (LPCG)

The LPCG block receives the root clocks from CCM and splits them to clock branches for each block. The clock branches are individually gated clocks.

The enables for those gates can come from three sources:

- Clock enable signal from CCM - This signal is generated depending on the power mode the system is in. For each power mode, it is defined in the software using the configuration of the CGR bits in CCM.
- Clock enable signal from the block - This signal is generated by the block based on its internal logic. Not every enable signal from the block is used. Each clock enable signal from the block can be overridden based on the programmable bit in CCM.
- Clock enable signal from the reset controller (SRC) - This signal will enable the clock during the reset procedure.

10.3.3 Peripheral components of clock management system

10.3.3.1 Interface and functional clock

Each block within the SoC has specific clock input characteristic requirements. Based on the characteristics of the clocks delivered to modules, the clocks are divided into two categories: bus interface clocks and functional clocks.

The bus interface clocks have the following characteristics:

- They ensure proper communication between any block/subsystem and the system buses.
- In most cases, they supply the system interface and configuration registers of the block.
- A typical block has one system bus clock, but blocks with multiple interface clocks may also exist (that is, when a block is connected to multiple buses).
- The bus interface clocks are always fed by the outputs of the CCM/LPCG.
- Clock management for this type of clock is always implemented at the system level because it requires coordinated clock management between the block and system buses.

Functional clocks have the following characteristics:

- They supply the functional part of a block or a subsystem.
- Typically, these clocks are completely asynchronous and independent from the bus interface clock of the same block.
- A block can have one or more functional clocks. Some functional clocks are mandatory, while others are optional for its functioning. A block needs its mandatory

clock(s) to be operational. The optional clocks are used for specific features and can be shut down without stopping the block activity.
- The functional clocks are fed either by a CCM/LPCG block functional clock output, or by some other clock source, such as a clock output of another block or an external signal coming from IOMUX.

10.3.3.2 Block level clock management

Each block in the system may also have specific clock requirements. Certain module clocks must be active when operating in some specific modes, or may be gated in some others. Generally, the activation and gating of the module clocks are managed by LPCG. Hence, the LPCG block must be programmed properly and, in case of hardware controllable clock gating, peripheral module should provide signals indicating when to activate and when to gate the module clocks.

The LPCG block differentiates the clock-management behavior for device modules based on whether the block can initiate transactions on the device interconnect (called master module), or if it cannot initiate transactions and only responds to the transactions initiated by the master (called slave module). Thus, two hardware-based clock-management protocols are used:

- Master protocol - Clock-management protocol between the CCM/LPCG and blocks that can be bus master
- Slave protocol - Clock-management protocol between the CCM/LPCG and slave modules

10.3.3.2.1 Master clock protocol

This protocol is used to indicate that a master module is ready to initiate a transaction on the device interconnect and requests specific (both functional and interface) clocks. The CCM/LPCG block ensures that the required clocks are active when the master module requests that the CCM/LPCG enable them. The module is said to be functional after the required clocks are activated.

Similarly, when the master module no longer requires the clocks, it informs the LPCG/CCM block and the LPCG/CCM can then gate the clocks to the module and all the clock precedents that are not used by other blocks. The master module is then said to be in clock-gated or partially clock gated mode.

Examples of module supporting master clock protocol USDHC. Please see details in chapters describing these modules and in the CCM enable override register (CCM_CMEOR).

10.3.3.2.2 Slave clock protocol

This hardware protocol allows CCM to control the state of a slave module. CCM informs the slave module, through assertion of a stop/change request, when its clocks (both interface and functional) can be changed or gated. The slave acknowledges the request and CCM is then allowed to gate or change the clocks to the block.

Similarly, a clock-gated slave module may need to be woken up because of some event or a service request from a master module. In this situation, CCM enables the clocks to the module and then de-asserts the stop request to signal the module to wake up.

Examples of modules supporting slave clock protocol are CAN, EPIT and GPT. Please see details in chapters describing these modules and in the CCM Module Enable Overide Register (CCM_CMEOR). See CCM Memory Map/Register Definition for more details.

The protocol in both "master" and "slave" cases is completely hardware-controlled, but software should configure the clock management behavior for the module in two places: in the CCM registers associated with the block and in the block configuration registers.

10.3.3.3 Clock Domain(s)

A clock domain is a group of blocks fed by clock signals controlled by the same clock controls in CCM. By gating the clocks in a clock domain, the clocks to all the blocks belonging to that clock domain can be gated/activated, either by software control or by hardware control associated with block activity. Thus, a clock domain allows efficient control of the dynamic power consumption of the domain.

The device is partitioned into multiple clock domains and each clock domain is controlled by an associated group of clock gating cells within the LPCG block. This allows the CCM/LPCG to individually activate and gate each clock domain of the system.

10.3.3.4 Domain level clock management

The domain clock manager can automatically (based on hardware conditions) and manage the bus interface clocks within the clock domain. The functional clocks within the clock domain are managed through software settings.

10.3.3.5 Domain dependencies

A domain dependency is a hierarchical relationship between two clock domains. Clock domain "X" is said to depend on a clock domain "Y" when a block in clock domain "Y" provides services (or even just a clock) to a block in clock domain "X". As a result, clock domain "Y" must be active whenever clock domain "X" is active.

The dependency between two clock domains may also exist if one clock domain serves to ensure communication between two blocks (for example, the clock domain of the device interconnect).

10.4 Power management

10.4.1 Centralized Components of Power Management System

The power generation and management system is built around the PMU and GPC blocks.

A high level block diagram of the power management system in the SoC environment is shown in the figure below.

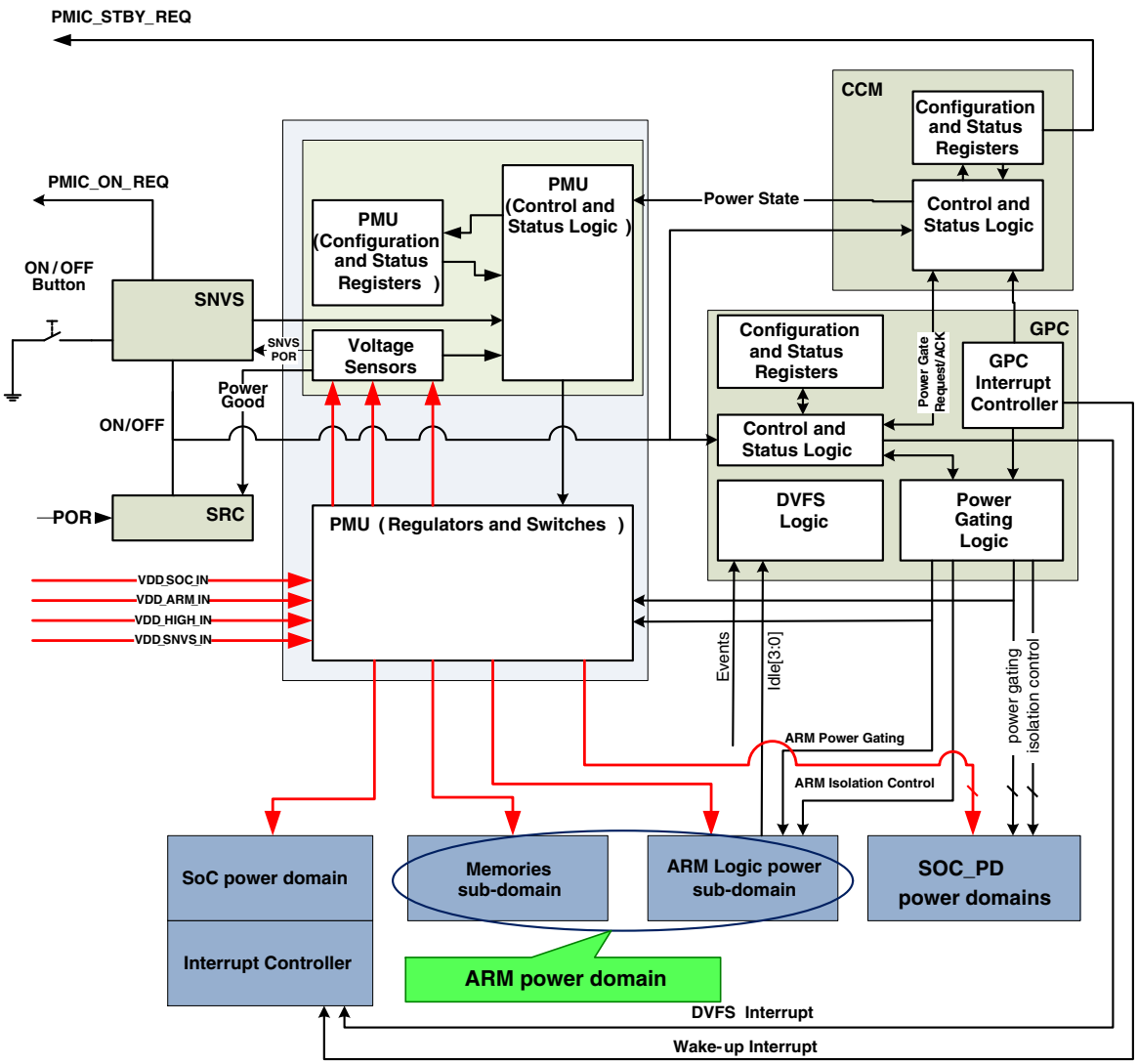


Figure 10-4. Power Management System

10.4.1.1 Integrated PMU

The first component of the power management system, referred to as the integrated PMU, is designed to simplify the external power interface.

It consists of a set of secondary power supplies that enable SoC operations from just two or three primary supplies. The high level block diagram of the power tree, utilizing the integrated PMU, is shown below.

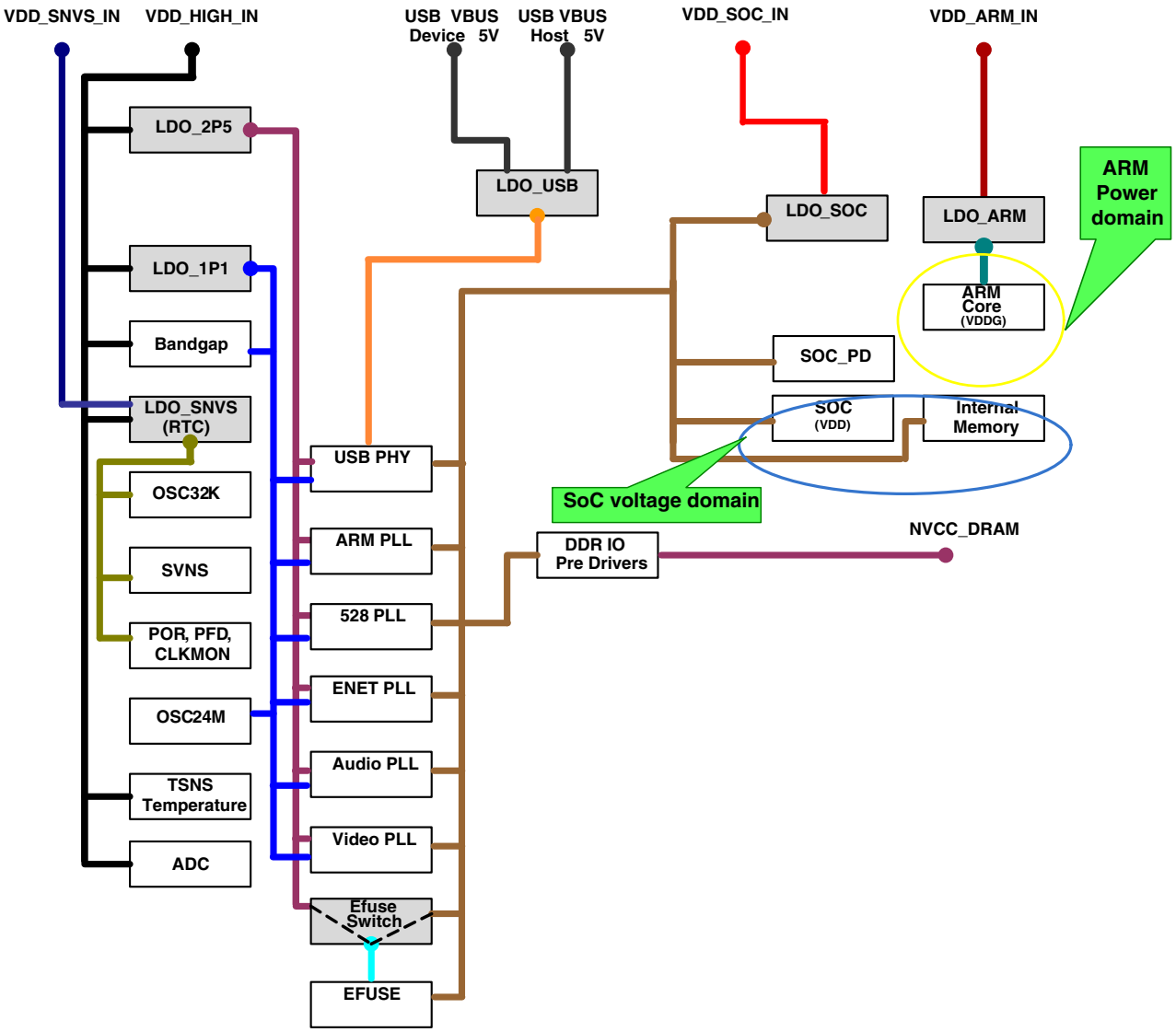


Figure 10-5. i.MX 6UltraLite Power Tree

The integrated PMU includes the following components:

- Two Digital LDO regulators
- Two Analog LDO regulators
- USB LDO
- SNVS regulator

See Power Management Unit (PMU) for further details on integrated PMU functional description and programmability.

10.4.1.1.1 Digital LDO Regulators

The integrated PMU includes three digital LDO regulators: LDO_ARM(VDD_ARM_IN, VDD_ARM_CAP), and LDO_SOC(VDD_SOC_IN, VDD_SOC_CAP). LDO_ARM and LDO_SOC regulators provide power to the ARM core power domain. LDO_SOC provides power to the SOC_PD and SOC power domains(except always-ON SNVS domain).

NOTE

The name "digital" only refers to the type of load. It is not related to the LDO design or feature set.

The digital LDO regulators can operate in the following modes:

- Internal Bypass - The regulation pass device (FET) is switched fully on, passing the external input voltage to the load unaltered. The analog part of the regulator is powered down in this state, removing any loss other than the IR drop through the power grid and the FET. Be aware that a period of time (see datasheet) is required to switch from the internal digital bypass mode to the analog regulation mode. Typically it takes less than 100us. Please refer to PMU for further details on bypass and power gate configuration.
- Power Gate - The regulation FET is switched fully off, limiting the current draw from the supply. The analog part of the regulator is powered down, limiting the power consumption. The output voltage will fall to a level where the residual leakage of the power FET balances with the leakage of the load.
- Analog regulation mode - The regulation FET is controlled such that the output voltage of the regulator equals the programmed target voltage. The target voltage is fully programmable in 25mV steps.

These modes allow the regulators to implement voltage scaling and power gating, and allow bypass when an external high power efficient regulator is used as a direct source for some of the SoC loads.

These digital regulators also feature brownout detection, which is helpful to sense when supplies are starting to collapse. Note that the core will be interrupted on a brownout. Please see details in Miscellaneous Control Register (PMU_MISC2*n*).

For further details of LDO programming and configuration please refer to Digital Regulator Core Register (PMU_REG_CORE).

The power management system is built under assumption that in typical applications the single (and simple) shared power supply will be used for ARM core domain and SoC domain. The combined load gains some efficiency, especially in low power modes and saves BoM significantly.

The DVFS in a typical cost/complexity optimized application is considered by mean of internal LDO. In"full speed" modes LDO bypass is considered in both domains. The dynamic voltage scaling to low load workpoints for ARM domain is implemented by programming associated LDO.

10.4.1.1.2 Analog LDO regulators

There are two analog LDO regulators used for general system purposes:

- LDO_1P1 - The LDO_1P1 (VDD_HIGH_IN, NVCC_PLL) linearly regulates down a higher supply voltage (2.8V-3.3V) to produce a nominal 1.1V output voltage. This regulator supplies digital portions of USB PHYs, PLLs, and the internal 24 MHz oscillator.
- LDO_2P5 - The LDO_2P5 (VDD_HIGH_IN, VDD_HIGH_CAP) linearly regulates down a higher supply voltage (2.8V-3.3V) to produce a nominal 2.5V output voltage. The regular 2.5V LDO is combined with an alternate self-biased low-precision weak regulator which can be enabled for applications that need to keep the 2.5V output voltage alive during low power modes, where the main regulator and its associated global bandgap reference module are disabled to save power. The output of this weak-regulator is not programmable and is a function of its input power supply as well as its load current. Typically with a 3V input power supply, the weak-regulator output is 2.525V and its output impedance is approximately 40Ohm. Special procedure is recommended to move load back and forth between the main and low power regulators. This regulator supplies most of the analog circuitry of the integrated PHYs, special I/Os (DDR I/O), and other analog and mix signal components integrated into the SoC.

10.4.1.1.3 USB LDO

The USB_LDO linearly regulates down the USB VBUS input voltages (typically 5V) to produce a nominal 3.0V output voltage. This regulator has a built in power-mux that allows the user to run the regulator from either one of the VBUS supplies when both are present. If only one of the VBUS voltages is present, the regulator automatically selects that supply. Current limit is also included to help the system keep the in-rush current within limits as required in USB 2.0 specification. This regulator supplies only low speed and full speed transceivers of USB PHYs.

10.4.1.1.4 SNVS regulator

The SNVS regulator takes the SNVS_IN supply and generates the SNVS_CAP supply, which in turn powers the real time clock and low power section of the SNVS blocks. If VDDHIGH_IN is present, then the SNVS_IN supply is internally shorted to the VDDHIGH_IN supply to allow coin cell recharging if necessary.

10.4.1.2 GPC - General Power Controller

The GPC block provides hardware assistance to Dynamic Voltage Frequency Scaling (DVFS) and power gating, and includes the sub-blocks listed here.

- DVFS load tracking block - This block allows hardware tracking on the core load and generates an interrupt when a frequency change is requested. It does not generate any request for voltage and/or frequency changes made by a hardware signal. The frequency/voltage changing process requires interaction with the CCM block, as well as either the integrated PMU modules or the external programmable regulator. This process should be completed by either the CPU interrupt routine or a DMA transaction.
- Power Gating Controller (PGC) - This sub-block of GPC has the following functions:
  - Provides the user with the ability to switch off power to a target subsystem.
  - Generates power-up and power-down control sequences. This includes interaction with CCM/LPCG and SRC, and control for clock and reset generation for power domains affected by power gating.
  - Provides programmable registers that adjust the timing of the power control signals.
  - Controls the CPU power domain and SOC_PD power domain.
- Wake-up interrupt controller - This controller initiates the system wake-up from low power modes when only low frequency real time clock remains active, and thus the Generic Interrupt Controller (GIC) can not handle synchronous interrupt signals. Additional features are as follows:
  - Supports up to 128 interrupts
  - Provides an option to mask/unmask each interrupt
  - Detects interrupts and generates the wake up signal

See General Power Controller (GPC) for further details on GPC, its sub-blocks, and information on its functional description and programmability.

10.4.1.3 SRC - System reset Controller

The reset controller is responsible for the generation of all reset signals and boot configuration decoding.

It determines the source and the type of reset, such as POR, WARM, and COLD, and performs the necessary reset signal qualifications. SRC is capable of generating reset sequences in the following conditions:

- in interaction with external PMIC, based on external POR_B signal and "power ready" signals generated by the integrated PMU
- or in interaction with the integrated PMU only, based on its "power ready" signal.

Based on the type of reset, the reset logic generates the reset sequence for either the entire SoC or for the blocks that are power-gated.

See System Reset Controller (SRC) for further details on SRC functional description and programmability.

10.4.1.4 Power domain(s)

A power domain is a group of blocks or sub-blocks fed by power sources controlled by the same power controls in GPC.

Some power domains can be split into a logic sub-domain and a memory sub-domain. The memory sub-domain in such case may contain two entities:

- Memory array(s) - Powered by a dedicated voltage rail enabling memory retention while core is OFF.
- Memory interface logic - Powered by the same voltage source as the logic sub-domain of the power domain.

Signals crossing power domain boundaries or sub-domain boundaries are passed through proper isolation and/or level-shifting cells to ensure robust operations of the SoC when some of domains are power gated or working at a reduced voltage.

Figure below shows the power domain interface in the system

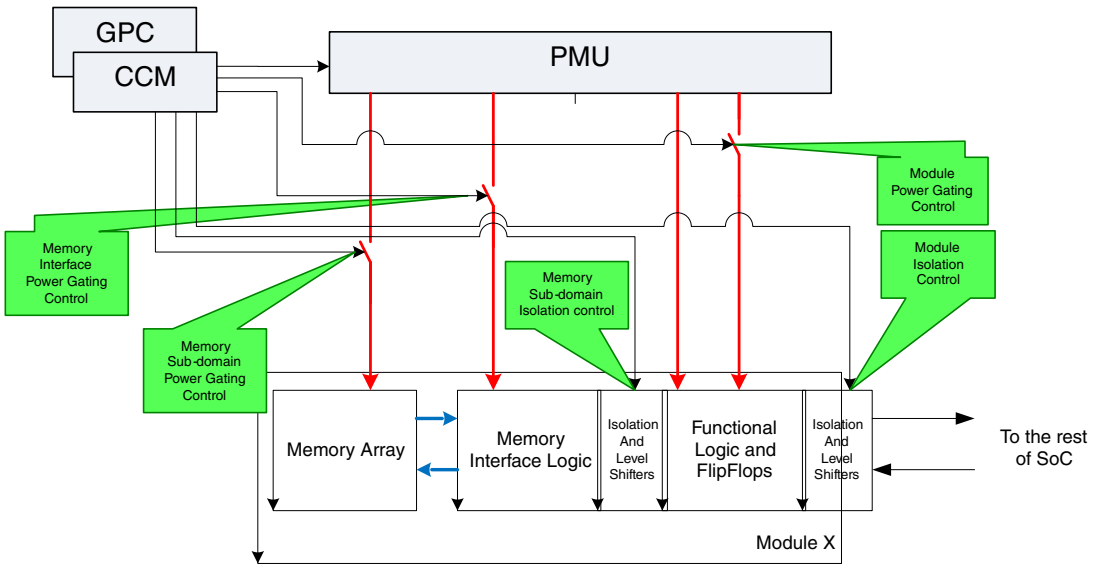


Figure 10-6. Power domain interface in system

10.4.1.4.1 Power distribution

The power distribution tree is comprised of multiple power domains. The main power domains are:

- ARM - The ARM domain contains the ARM Core platform (except for memory arrays and interface logic). This domain can be supplied either from an integrated power supply or from an external controllable regulator, preferably high efficiency DCDC converter.
- ARM Memory array - Memory arrays are connected to a separate and dedicated power domain (separated from the Main logic and ARM domain). In normal operation mode (functional, non-DVFS mode), the memory arrays domain voltage level should be kept equal to (same as) the rest of the core logic domains (Main, ARM). Please refer to the Datasheet for further information about voltage level difference between domains allowed in different power modes.
- SOC_PD domain -- The SOC_PD domain contains MMDC, APBHDMA, CSI, ENET, LCDIF, PXP, RawNAND, SDMA, uSDHC and USB. It is supplied by external regulator.
- SNVS/RTC low power domain - The SRTC domain contains only counter, comparator and compared data of the on-chip RTC. This domain should be supplied from an external single cell LiION battery and/or an external pre-regulated power supply.
- Analog domain - The analog domain contains the PLLs, LDOs and USB PHY. The domain supplies should be constant to allow continuous clock during any dynamic voltage scaling techniques. The digital supply should be provided from an internal

regulator, and can be combined with the memory array supply. The analog supply should be provided from internal low noise regulator.
- Main SoC logic - The main SoC logic domain contains the rest of the logic of the SoC. It is supplied by an internal regulator.

From a DVFS and Power Gating standpoint, the following digital logic domains are affected:

- Cortex-A7 Core Platform - DVFS and power gating.
- ARM Cortex-A7 memories - Power gating only.
- SOC_PD - Power gating only.

See table below for details of the system power domains layout and dependencies.

10.4.1.4.2 Domain Memory and domain logic state retention in case of Power Gating

The following is the list of relevant memories and logic domains with the description of their state-retention support:

- Cortex-A7 Core Platform is sub-divided into three sub-domains listed below:
- Cortex-A7 Core Platform logic: The software state retention for all logic is implemented in this domain. That means that the content of relevant registers should be be stored in some memory retaining its state (L2 cache for example) while the logic domain is power-gated. Details on how to implement the software retention can be found in the Cortex-A7 Core Platform TRM.
- Cortex-A7 Core L1 memories - No retention. The L1 memories have a dedicated supply on the package (VDD_CACHE_CAP) which should be connected to the Cortex-A7 Core Platform supply. The L1 cache should be flushed prior to power gating in order to allow powering up of the CPU at the same state as before power gating.
- Cortex-A7 Core L2 memories - hardware state-retention since its supplies are driven by the SoC supplies. When Cortex-A7 core is power gated, L2 Cache can be either leaved on with its content retained or power gated together with the core.

- SOC_PD: This module can be powered gated (together) independently of Cortex-A7 power gating. SOC_PD is not allowed power down when Cortex-A7 is on. SOC_PD will be powered on automatically together with Cortex-A7.
- SoC - hardware state-retention in Standby mode.
- SNVS_LP - hardware state-retention even when SoC supplies are removed.

10.4.1.4.3 Power Gating Domain Management

The following bullets provide the sequence required for power-gating the relevant power-domains:

10.4.1.4.3.1 Cortex-A7 Core Platform

1. Copy through software all the Core configuration registers to a powered-on memory
2. Configure the GPC/PGC CPU registers in PGC Memory Map/Register Definition as follows to power-down the core on the next "WFI" instruction:
   - Configure the GPC/PGC PGC_CPU_PDNSCR Register ISO and ISO2SW bits. These bits determine the delay between the power-down request to enabling the platform isolation and the platform isolation to the actual power-off switch to the supplies accordingly.
   - Configure the GPC/PGC PGC_CPU_PUPSCR Register SW and SW2ISO bits. These bits determine the delay between the power-up request to the actual power-up of the supplies and the last to the platform isolation disabling.
   - Configure the GPC PGC PGC_CPU_CTRL PCR bit to allow the power down of the platform
   - Cortex-A7 Core Platform should execute a "WFI" instruction.

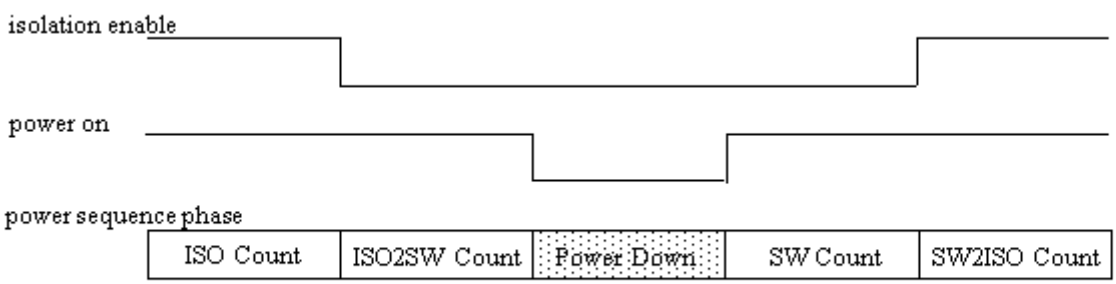


Figure 10-7. Cortex-A7 Core Platform isolation and power on switch flow

10.4.1.4.3.2 SOC_PD

1. Configure the CCM CGR bits (CCM Memory Map/Register Definition) to disable the SOC_PD clocks.
2. Configure the GPC/PGC Registers (GPC Memory Map/Register Definition) as follows to power-down isolate the SOC_PD logic from the rest of the SoC logic:

Configure the GPC/PGC PDNSCR Register ISO bits. These bits determine the delay between the power-down request to enabling the LDO domain isolation.

Configure the GPC/PGC PUPSCR Register SW2ISO bits. These bits determine the power-up request to the LDO domain isolation disabling.

Configure the GPC/PGC CTRL[PCR] bit to allow the power down of the block.

Configure the GPC/PGC GPC_CNTR to power down SOC_PD

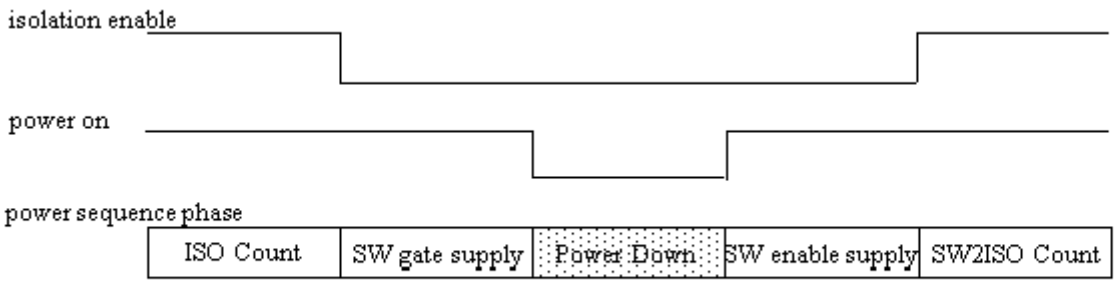


Figure 10-8. SOC_PD isolation and power on switch flow

10.4.1.4.3.3 SoC

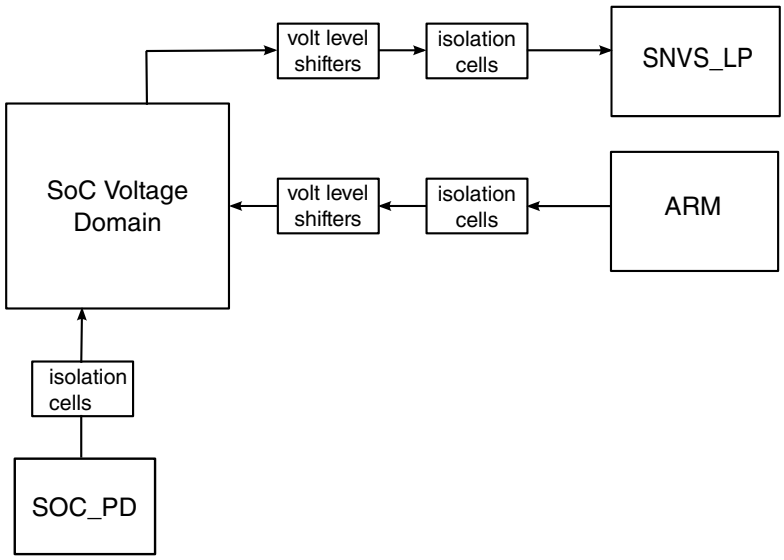
For additional power reduction it is possible to do the following:

- Power-down the internal oscillator by configuring the following bits CCM_CCR[COSC_EN] (CCM Memory Map/Register Definition). This can be done only in case there is no dependency on 24 MHz XTAL for wake-up.
- It is possible to turn off and turn on the PMIC supplies to the SoC even when the SoC supplies are off. Since SNVS_LP is powered through an "always on" supply, configuring the SNVS_LP DP_EN to "1" allows changing the PMIC_ON_REQ pad (SoC on/off supply indication to the PMIC) through the ONOFF pad.

10.4.1.4.4 Power Gating domain dependencies

There are 3 power domains that need to be isolated in different power-down cases:

- Cortex-A7 Core Platform and SOC_PD - Isolation needs to be enabled before power-down. This is taken care of automatically once CCM and PGC are configured and the Cortex-A7 Core Platform executes the "WFI" instruction.
- SNVS_LP - Different from the 2 cases above the SNVS_LP isolation isolates the signals coming from the SoC to the SNVS_LP. This is required for saving the contents of the SNVS_LP. (such as the real-time clock) The isolation is activated in 2 ways:
- Automatically through the power-fail detector in the PMU
- Through software configuration



Note: The arrows refer to the signal directions for the voltage level shifters and isolation cells

Figure 10-9. Isolation cells and Voltage level shifters placing

10.4.1.5 Voltage domains

The list found here states the different voltage domains and their scalability in regarding to power-saving in dynamic and static scenarios.

- ARM - Cortex-A7 Core Platform including L1 and L2 cache - Scalable voltage in both dynamic and static scenarios
- SoC Domains - Scalable voltage only in static scenarios
- ANALOG components including PLLs - Fixed voltage
- I/O - Fixed voltage
- SNVS_LP - Fixed voltage

10.4.1.6 Voltage domain management

10.4.1.6.1 Dynamic

10.4.1.6.1.1 DVFS

Dynamic Voltage and Frequency Scaling (DVFS) is a well-known technique to reduce power consumption in mobile devices. In order to improve power saving efficiency, DVFS is applied on the ARM core voltage domain.

In this scenario the Dynamic Voltage and Frequency Scaling (DVFS) block is used. More details can be found in DVFS-CORE (DVFS). The DVFS block can monitor separately the IDLE indication of each of the Cortex-A7 Cores. The DVFS performs the following:

- Simple, non-overlapping averaging providing a level-based average index of the tracked CPU load
- Sums the CPU load and the load detected from additional load indicators weighted according to software configurations
- Calculates an exponential moving average of the tracked load
- Provides up, down and "panic" threshold comparators and counters for generating interrupts to the Cortex-A7 Core Platform
- Frequency pattern generator is able to manage the frequency update requests periodically

The DVFS interrupts are then forwarded to the Cortex-A7 Core Platform. The Cortex-A7 Core Platform reacts according to the DVFS interrupt types. The following are 2 examples for handling different DVFS interrupt types:

- Upper threshold is reached, meaning that the Core is heavily loaded and the core frequency needs to be increased according to the following flow:
  - Configure PMU to raise the core voltage
  - Wait until the voltage is stable
  - Configure CCM to raise the frequency.
- Lower threshold is reached, meaning that the Core is "IDLE" most of the time and the core frequency can be decreased according to the following flow:
  - Configure CCM to reduce the frequency.
  - Configure PMU to lower the core voltage.

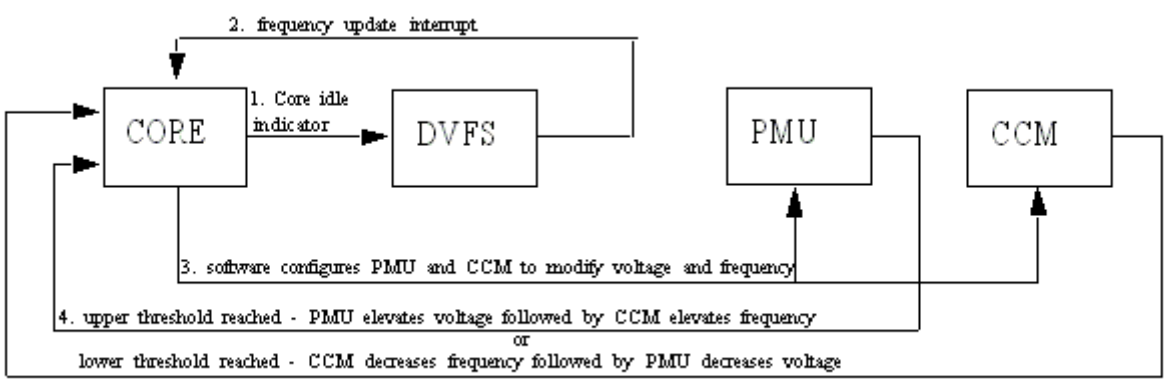


Figure 10-10. High Level DVFS working flow

NOTE

PMU cannot be used to change the voltage in case the LDO is bypassed.

10.4.1.6.1.2 Voltage Scaling

A simplistic way to reduce power consumption in dynamic scenarios is to scale down the ARM, SoC and LDOs voltage according to the allowed voltage points and corresponding frequencies specified in datasheet.

10.4.1.6.2 Static

10.4.1.6.2.1 Standby Leakage reduction (SLR)

Standby leakage reduction is a power-management technique utilizing:

- Reduced supply voltage for relevant domains

With SLR, the device switches into low-power active system modes automatically or in response to user requests during system Stop, Wait, or DSM modes (that is, in situations when no application is started and no system activity is presented).

When applying SLR, the system remains in the lowest static power mode while retaining logic and memory states. This technique trades static power consumption for wake-up latency while maintaining fast system response time suitable for most applications.

See CCM Control Register (CCM_CCR), CCM Low Power Control Register (CCM_CLPCR) and PMU Miscellaneous Register 0 (PMU_MISC0) for further details on SLR programmability options.

The following describes the flow for applying standby voltage:

- Configure the external PMIC standby voltage, refer to chip datasheet.
- Configure CCM_CCR[RBC_EN] bits to bypass and disable PMU regulators in the next ARM "WFI" execution.
- Configure CCM_CCR[REG_BYP_COUNT] bits to allow proper voltage restoration by the external PMIC when exiting standby.
- Cortex-A7 Core Platform executes the "WFI" instruction that completes the software sequence putting the SoC into low power mode

10.4.1.6.2.2 ANALOG PHYs IPs -

Details on how to put the IPs in low power mode can be found in each of the IP documentation. Further reduction can be achieved by settings the ENABLE_WEAK_LINREG in the PMU PMU_REG_2P5 register.

10.4.1.6.3 Voltage domain dependencies

When applying voltage changes for power saving the following voltage domain limitations should be taken into account:

- In dynamic scenario's the ARM supply should not be higher than the SoC supply by more the 50mV.

The above limitationsare also mentioned in datasheet.

10.4.1.6.4 IO voltage

10.4.1.7 System domains layout

The following table describes the different power modes.

Table 10-1. Low Power Mode Definition (LDO Enabled Mode)

	System IDLE	Low Power IDLE	SUSPEND
CCM LPM Mode	WAIT	WAIT	STOP
ARM Core	Low Voltage	Power Down	Power Down
L1 Cache	ON	Power Down	Power Down
L2 Cache	ON	ON	Power Down
SOC	Nominal Voltage	Nominal Voltage	Standby Voltage
SOC_PD	ON	ON	ON/Power Down
PLL	ON/Power Down	Power Down	Power Down
XTAL	ON	OFF	OFF
RC OSC	OFF	ON	OFF
DRAM	Self_Refresh[1]	Self-Refresh[2]	Self-Refresh[2]
DRAM IO Low Power	No	Yes	Yes
LDO ARM	ON	Power Gate	Power Gate
LDO SOC	ON	ON	Digital Bypass
LDO2P5	ON	OFF	OFF
LDO1P1	ON	OFF	OFF
WEAK2P5	OFF	ON	OFF
WEAK1P1	OFF	ON	OFF
Bandgap	ON	ON	OFF
Low Power Bandgap	ON	ON	OFF
MMDC Clock	24 MHz	OFF	OFF
AHB clock	24 MHz	3 MHz	OFF
IPG clock	12 MHz	1.5 MHz	OFF
PER clock	24 MHz	1 MHz	OFF
Module clocks	ON as needed	ON as needed	OFF

Table continues on the next page...

Table 10-1. Low Power Mode Definition (LDO Enabled Mode) (continued)

	System IDLE	Low Power IDLE	SUSPEND
GPIO wakeup	yes	Yes	Yes
RTC wakeup	Yes	Yes	Yes
USB remote wakeup	Yes	Yes[3]	Yes[3]
Other wakeup source	Yes	Yes[4]	No

1. Automatic enter self-refresh when there is no DRAM access;
2. Put into self-refresh mode by SW before entering low power mode;
3. Need SOC_PD power on to support USB remote wakeup;
4. When a module is inside SOC_PD domain, need SOC_PD power on to support wakeup from it.

Table 10-2. Low Power Mode Definition (LDO Bypass Mode)

	System IDLE	Low Power IDLE	SUSPEND
CCM LPM Mode	WAIT	WAIT	STOP
ARM Core	Low Voltage	Power Down	Power Down
L1 Cache	ON	Power Down	Power Down
L2 Cache	ON	ON	Power Down
SOC	Nominal Voltage	Nominal Voltage	Standby Voltage
SOC_PD	ON	ON	ON/Power Down
PLL	ON/Power Down	Power Down	Power Down
XTAL	ON	OFF	OFF
RC OSC	OFF	ON	OFF
DRAM	Self_Refresh[1]	Self-Refresh[2]	Self-Refresh[2]
DRAM IO Low Power	No	Yes	Yes
LDO ARM	Digital Bypass	Power Gate	Power Gate
LDO SOC	Digital Bypass	Digital Bypass	Digital Bypass
LDO2P5	ON	OFF	OFF
LDO1P1	ON	OFF	OFF
WEAK2P5	OFF	ON	OFF
WEAK1P1	OFF	ON	OFF
Bandgap	ON	OFF	OFF
Low Power Bandgap	OFF	ON	OFF
MMDC Clock	24 MHz	OFF	OFF
AHB clock	24 MHz	3 MHz	OFF
IPG clock	12 MHz	1.5 MHz	OFF
PER clock	24 MHz	1 MHz	OFF
Module clocks	ON as needed	ON as needed	OFF
GPIO wakeup	Yes	Yes	Yes
RTC wakeup	Yes	Yes	Yes
USB remote wakeup	Yes	Yes[3]	Yes[3]
Other wakeup source	Yes	Yes[4]	No

1. Automatic enter self-refresh when there is no DRAM access;

2. Put into self-refresh mode by SW before entering low power mode;
3. Need SOC_PD power on to support USB remote wakeup;
4. When a module is inside SOC_PD domain, need SOC_PD power on to support wakeup from it.

There is a single hardware signal coming into PMU which sets the PMU in either of two "STOP" states. The STOP state is implemented is controlled by the PMU_MISC0[STOP_MODE_CONFIG] bit (See PMU Memory Map/Register Definition). It is recommended that the blocks be configured for safe powerdown/up through the registers before asserting the stop_mode signal. Blocks not described in the section below are unaffected by stop_mode.

If the stop_mode_config is set to zero, thus in the STOP mode all blocks powered down in minimum power configuration.

If the stop_mode_config is set to one, thus in the STOP mode some of the blocks remain powered and in different states as defined in the table below.

Table 10-3. STOP mode configuration

Block	STOP_MODE_CONFIG=0	STOP_MODE_CONFIG=1
reg1p1	off	on
reg2p5	off	on
reg3p0	off/on depending on vbus. Uses crude local reference if vbus is present	off/on depending on vbus . Uses analog central bandgap if VBUS is present.
reg_core	bypassed if not power gated.	bypassed if not power gated
reg_soc	bypassed	bypassed
bandgap	off	functional
temp_sensor	off	off
well_bias	hardware controlled	hardware controlled
All PLLs	off	off
OSC24M	off	Controlled by CCM configuration

10.4.2 Power management techniques

The device supports the power-management techniques with the features found here.

- Partitioning of the device into voltage, power, clock, and reset domains
- Domain isolation that allows flexible configurations of domains on/off states to form use cases targeting various applications
- Clock tree with selective clock-gating conditions and almost independent clock roots
- Power, reset, and clock control hardware mechanism to manage sleep and wake-up dependencies of power domains
- Software-controllable and hardware-controllable clock gating for functional modules and buses

- Memory retention and state retention capability (Software State Retention for ARM A7) for preserving memory contents and device state in low-power modes
- Dynamic Voltage and Frequency Scaling (DVFS) support for the ARM A7 processor cluster
- Support for low-power device modes input/output (I/O) pad configuration for minimum power
- Variety of operating modes to optimize device performance and wake-up times
- Thermal monitoring and thermal aware performance management

Many of the low power features are fully or partially software controllable and can be configured for the specific requirements of a target system.

Combining these techniques, the system designer may meet tight requirements of low-power standby and operational modes while maintaining high performance for time-critical tasks.

10.4.2.1 Power saving techniques

The table below lists power saving techniques supported by the SoC in their connection to different components of power consumption.

Table 10-4. Power saving design/architecture and power saving techniques

Techniques	Active SoC Power	Standby SoC Power	System Power
Temperature Monitoring, and active frequency throttling	√		
Cortex-A7 Core Platform DVFS	√		
Cortex-A7 Core Platform SRPG (Software)		√	
Cortex-A7 Core Platform Power Gating		√	
Clock gating (automatic dynamic and forced)	√		
Integrated PMU (IR drop, efficiency, accuracy)	√		√
C4 package (IR drop, thermal)	√		
Display Backlight optimization (SW)			√
Architecture: L2 cache	√		
Architecture: USB integration			√
Low Power DDR: LPDDR2, LV-DDR3			√

10.4.2.2 Thermal-aware power management

The temperature sensor block (TEMPMON) implements a temperature sensor/conversion function. The block features an alarm function that can raise an interrupt signal if the temperature is above a specified threshold.

Software may implement temperature aware DVFS for the ARM domain as well as temperature aware frequency scaling for other system components to ensure that both the frequency and voltage is lowered when the die temperature is above the specified limit.

Software may also implement temperature aware task scheduling to ensure that non-critical tasks are suspended when the die temperature is above the specified limit.

See Temperature Monitor (TEMPMON) for further details on temperature monitor functions and programmability options.

10.4.2.3 Peripheral Power management

10.4.2.3.1 Main memory power management

Main system memory, DDR3, and LPDDR2 are some of the most power-hungry system components, but the SoC provides several options to manage DDR power.

Automated power saving modes are supported by the MMDC hardware. This feature allows the DDR memory to automatically enter self-refresh mode when there are no DDR accesses for a configurable time. The default setting is 1024 clock cycles which can be optimized based on the customer use case and application.

See Power Saving and Clock Frequency Change modes for further details on MMDC power saving features and programmability options.

Software may support DDR frequency scaling. Automated frequency changing procedure is supported by MMDC and CCM modules.

NOTE

DDR frequency changes cost extra time and power. Slowing requestors while keeping DDR at full speed may increase total system power. Software may also implement Cooperative Dynamic Frequency Scaling in order to keep the system balanced, (that is, keep the system in balance when DDR throughput is equal or slightly higher than total amount of requests generated by all requestors).

Reducing the DDR frequency while in DLL-ON mode may be not efficient because:

- Reduction in DDR frequency will cause bus duty cycle to increase and thus reduces chance of automatic MMDC power saving (place memory into SR).
- Total amount of read/write operation does not change (power is per-operation).
- The termination is active longer, though, lowering frequency below 396 MHz may enable lowering drive strengths and termination.

When possible at lower performance use cases, software may switch DDR3 to DLL-off mode. This allows it to greatly reduce DDR3 frequency and thus disable or reduce termination and drive strength, which significantly reduces the power consumption of the DDR3 interface.

A good strategy for many types of workloads is to combine most activity in bursts (natively possible, for example, for typical multimedia applications, communication, etc.) and run this segment at maximal speed, then switch to DLL-OFF mode to support background activity (communication, display refresh, housekeeping).

The DRAM Interface power dissipation depends on many variables, however, proper termination and drive strength is key for power and thermal performance. Memory and controllers provide a host of programmable options for the drive strength of the output buffers and for the on-die termination impedance.

The ideal settings for drive strength and ODT also depend on the clock frequency to ensure that inter-symbol interference (ISI) effects are not introduced.

DDR PHY power is proportional to the amount and type of bus activity.

In cases where the DDR is placed into self-refresh, software can configure DDR I/O to be floated or lowered to the minimum drive allowed by JEDEC.

Modifying the DDR drive strength must be done by code that is executing from a memory region other than DDR (for example, IRAM). No access to DDR (including page table walks, cache misses, alternate bus master accesses, etc.) is allowed while the DDR I/O pads are being re-configured.

10.4.2.3.2 Video-Graphics system power management

10.4.2.3.3 IO power reduction

SW configures IO to low power modes:

- PHYs - make sure that all unused PHYs are placed to lowest power state. Please refer relevant chapter for further information about different PHYs
- Digital IOs - Make sure all unnecessary PU/PD are disabled and IO are switched to either minimal drive strength or to input mode (when applicable)
- Set DDR type IO to CMOS mode if possible

10.4.3 Examples of External Power Supply Interface

This section presents the example of external power supply interfacing to the chip.

The scenario based on integrated PMU system is presented below. This scenario minimizes BoM and board design complexity.

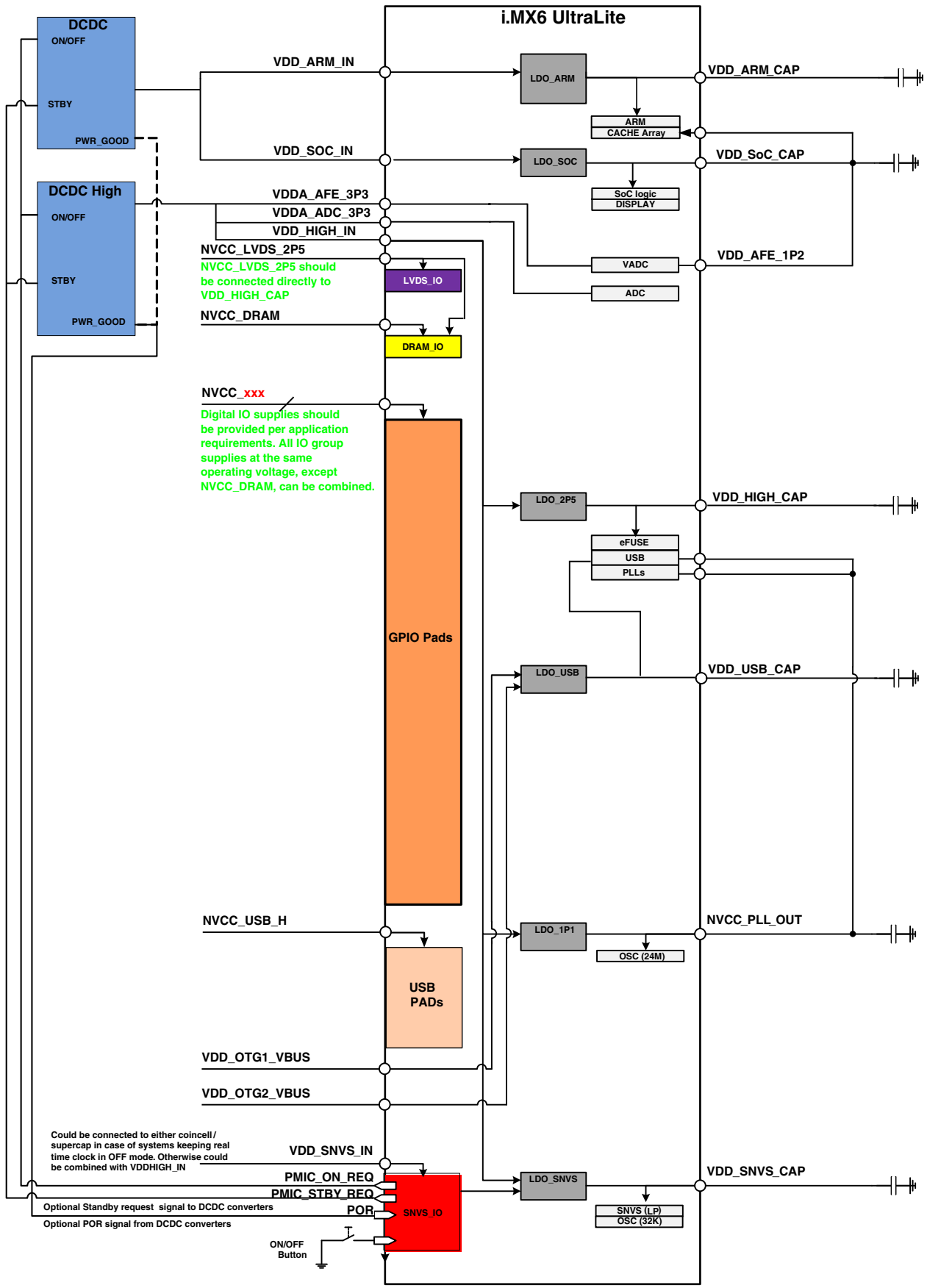


Figure 10-11. Supplying i.MX 6UltraLite power using integrated PMU

10.5 ONOFF (Button)

The chip supports the use of a button input signal to request main SoC power state changes (i.e. On or Off) from the PMU.

The ONOFF logic inside of SNVS_LP allows for connecting directly to a PMIC or other voltage regulator device. The logic takes a button input signal and then outputs a pmic_en_b and set_pwr_off_irq signal. PMIC logic also supports the SNVS_LP tamper logic which will allow waking the system up when a tamper event has happened while in the OFF state. The logic has two different modes of operation (Dumb and Smart mode).

The Dumb PMIC Mode uses pmic_en_b to issue a level signal for on and off. Dumb pmic mode has many different configuration options which include (debounce, off to on time, and max time out).

- Debounce: The debounce configuration supports 0 msec, 50 msec, 100 msec and 500 msec. The debounce is used to generate the set_pwr_off_irq interrupt. While in the ON state and the button is pressed longer than the debounce time the set_pwr_off_irq is generated.
- Off to On Time: The Off to On configuration supports 0 msec, 50 msec, 100 msec, and 500 msec. This configuration supports the time it takes to request power on after the configured button press time has been reached. Once the button is pressed longer than the configuration time, the state machine will transition from the OFF to the ON state.
- Max Timeout: The max timeout configuration supports 5 secs, 10 secs, 15 secs and disable. This configuration supports the time it takes to request power down after the button has been pressed for the defined time.

The Dumb PMIC mode uses a 2 state state machine, as shown below. The output of the pmic_en_b is generated by the state of the state machine.

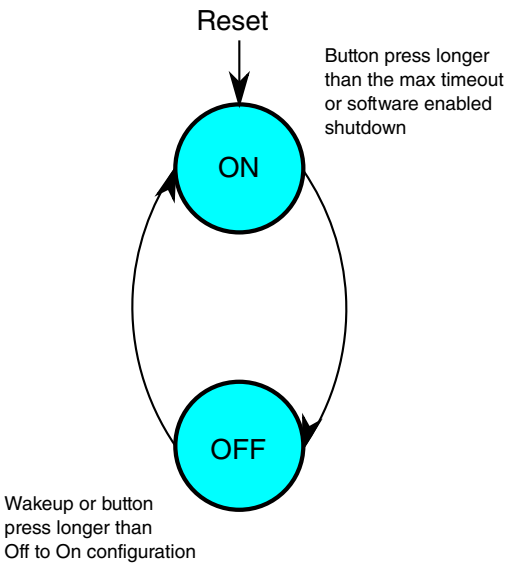


Figure 10-12. Dumb PMIC Mode State Machine

The Smart PMIC mode is meant to connect to another PMIC. The pmic_en_b signal issues a pulse instead of a level signal. The only configuration option available for this mode is the Debounce configuration that is used for the set_pwr_off_irq.

Chapter 11
System Security

11.1 Overview

Security is a common requirement for platforms built using the chip, although the specific needs vary greatly depending on the platform and market. The type and cost of assets to be protected on a portable consumer device are very different from those to be protected on automotive or industrial platforms, and the same applies to the kind of attacks and level of resources threatening those assets. The platform designer must select an appropriate set of counter measures to meet the relevant platform security needs.

For the platform designer to meet the requirements for each market, the chip incorporates a range of security features which can be used individually or in concert to underpin the platform security architecture. Most of the chip security features provide protection against particular kinds of attack and can be configured at various levels according to the required degree of protection. These features are designed to work together and can be integrated with appropriate software to create defensive layers. In addition to protection features, the chip includes a general purpose accelerator to enhance the performance of selected industry standard cryptographic algorithms.

The following is an introduction to the chip security components.

- High Assurance Boot (HAB) feature in the System Boot up to RSA-4096 signature verification
- Secure Non Volatile Storage (SNVS)
- TrustZone (TZ) Architecture in the ARM Cortex A7 Platform, TrustZone aware Interrupt Controller (GIC) and TrustZone Watchdog Timer (WDOG-2)
- TrustZone Address Space Controller (TZC-380) - providing security address region control functions on DDR memory space.
- On-chip RAM (OCRAM) with TrustZone protection using OCRAM controller.
- 64 Kbyte of on-chip Secure RAM
- On chip OTP (OCOTP) with on-chip electrical fuses
- Central Security Unit (CSU)

- Secure JTAG Controller (SJC)
- Locked mode in the Smart Direct Memory Access (SDMA) controller
- DryICE (real-time monitors for frequency, temperature and voltage)
- 10 tamper pins with 5 active tamper detection sources support
- Hardware Cryptographic Accelerators
  - Symmetric: AES-128, AES-192, AES-256, DES, 3DES, and ARC4
  - Hash Message Digest and HMAC: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, and MD-5
  - Public Key RSA (up to 4096 bit) and ECC (up to 1023 bit)
- DPA protection for 3DES engine
- True and Pseudo Random Number Generator
- Bus Encryption Engine (BEE)

11.2 Central Security Unit (CSU)

11.2.1 CSU Overview

The CSU manages the system security policy for peripheral access on the SoC. The CSU allows trusted code to set individual security access privilges on each of the peripherals, using one of eight security access privilege levels. Also, according to programmed policy, the CSU may assign bus master security privileges during bus transactions.

11.2.2 CSU Features

The Central Security Unit (CSU) sets access control policies between bus masters and bus slaves, allowing peripherals to be separated into distinct security domains. This protects against unauthorized access to data e.g. when software programs a DMA bus master to access addresses that the software itself is prohibited from accessing directly. Configuring DMA bus master privileges in the CSU consistent with software privileges defends against such attempted accesses.

CSU has the following security related features:

- Peripheral access policy - Appropriate bus master privileges and identity are required to access each peripheral.
- Masters privilege policy - CSU overrides bus master privilege signals, i.e. user/supervisor secure/non-secure, according to access control policy.

11.2.3 CSU Functional Description

The CSU enables secure software to set bus privilege security policy within the platform.

Security policies may be set, and optionally locked in the CSU registers. These privilege values may originate in the command sequence file (CSF) which is processed by the High Assurance Boot (HAB) itself or by an HAB authenticated image which executes after the initial boot ROM phase.

11.2.3.1 CSU Peripheral Access Policy

According to its programmed policy, the CSU determines the bus master privileges and the masters that are allowed to access each of the slave peripherals.

There are four security modes of operation (i.e. bus privileges) in the system distinguished by security (TrustZone/non-TrustZone) and privilege (Supervisor/User) setting of the module. Below is the list of these security modes from the highest security level to the lowest:

- TrustZone (Secure) Privilege (Supervisor) Mode - Highest Security Level
- TrustZone (Secure) non-Privilege (User) Mode - Medium Security Level
- non-TrustZone (Regular) Privilege (Supervisor) Mode - Medium Security Level
- non-TrustZone (Regular) non-Privilege (User) Mode - Lowest Security Level

This functionality is implemented as follows:

The Configure Slave Level (CSL) Register value for a specified peripheral resource defines the output signal -- csu_sec_level for that peripheral. The value of this signal determines by what master privileges a peripheral is accessible. The relationship between the value of the csu_sec_level signal and security operation mode is shown in the table below. The CSL registers reside in the CSU module. Details, describing CSL register fields and how they are programmed to control access privileges for specific peripherals, can be found in the Security Reference Manual.

Table 11-1. Permission Access Table

CSU_SEC_LEVEL[2:0]	Non-Secure User Mode	Non-Secure Spvr Mode	Secure (TZ) User Mode	Secure (TZ) Spvr Mode	CSL register value
(0) 000	RD+WR	RD+WR	RD+WR	RD+WR	8'b1111_1111
(1) 001	None	RD+WR	RD+WR	RD+WR	8'b1011_1011
(2) 010	RD	RD	RD+WR	RD+WR	8'b0011_1111
(3) 011	None	RD	RD+WR	RD+WR	8'b0011_1011

Table continues on the next page...

Table 11-1. Permission Access Table (continued)

CSU_SEC_LEVEL[2:0]	Non-Secure User Mode	Non-Secure Spvr Mode	Secure (TZ) User Mode	Secure (TZ) Spvr Mode	CSL register value
(4) 100	None	None	RD+WR	RD+WR	8'b0011_0011
(5) 101	None	None	None	RD+WR	8'b0010_0010
(6) 110	None	None	RD	RD	8'b0000_0011
(7) 111	None	None	None	None	Any other value

11.3 Cryptographic Acceleration and Assurance Module (CAAM)

11.3.1 CAAM Overview

CAAM is a cryptographic acceleration device that accelerates block encryption algorithms, stream cipher algorithms, hashing algorithms, and random number generation. It has an integral DMA engine that allows CAAM to fetch its command programs, read input data and write the resulting output.

CAAM works with the SNVS to provide platform assurance features, including support for High Assurance Boot, detection of and response to potential tamper events, and shortterm and long-term protection of secret data such as public keypairs, Digital Rights Management keys and proprietary software.

CAAM provides the following features:

- Cryptographic acceleration of hashing, encryption and decryption
- Offloading of cryptographic functions via programmable command descriptor language
- Two independent queues for commands and results
- Register Bus Interface used for configuration, control and status
- DMA, including scatter/gather support for data
- Automatic short-term encryption/decryption of cryptographic keys
- Long-term protection of secret data via device-specific encryption keys
- Secure Memory for access-controlled protection of critical data, and automatic zeroization in the case of tampering
- Random Number Generation using a true random number generator and a NISTcompliant pseudo random number generator

11.4 Secure Non-Volatile Storage (SNVS)

11.4.1 SNVS Overview

SNVS is a hardware device that includes a security state machine and security violation detection circuits that, together with High Assurance Boot software, determine whether the chip is currently in a secure state.

When the security state machine indicates a secure state, the SNVS allows use of special cryptographic keys to decrypt long-term secrets such as public/private keypairs, Digital Rights Management keys and proprietary software. When the SNVS detects a potential security violation, such as a tamper alert, the SNVS sends an interrupt to alert the Operating System of the event. The SNVS also includes a general purpose real-time counter.

The SNVS includes the following features:

- Security State Machine driven by High Assurance Boot software and tamper detection circuits
- Master Key Control that protects the integrity and secrecy of the Master Key (OTPMK) stored in fuses
- Tamper detection circuits that detect JTAG events, power glitches, Master Key ECC check failure, and software-reported and hardware-reported security violations
- 256-bit Zeroizable Master Key that can be automatically erased in the event of a security breach
- Tamper-protected Secure Realtime Counter that continues running when the chip is powered off
- Non-volatile Monotonic Counter used to protect against “roll-back” attacks
- Non-volatile General Purpose Register can be used to store a 32-bit value across power cycles
- Non-Secure Real Time Counter with programmable alarm and periodic interrupt
- Controls the access to the OTP master secret key used by CAAM to protect confidential data in off-chip storage
- Smart button input in the SNVS_LP logic of the SoC.
- Pulse mode output signal to request power state changes to a ‘Smart’ external PMIC.

11.4.2 Tamper Detection

There are 2 types of tamper detection's supported, external and internal. The internal tamper detection supports voltage, temperature and clock monitors. Voltage Monitor, monitors the Out of Range Battery Voltage. The allowed battery voltage is from 2.5V (low-trip point) to 3.5V (high-trip point). hysteresis is 100mV for each trip-point. Temperature Monitor has the temp low trip-point spec at -30°C and high trip-point spec at 115°C. Hysteresis is 5°C for each trip-point.

External Tamper Detection is a special mechanism provided through a chip pin to signal when the device encounters unauthorized opening or tampering. Inside the chip, the received signal is compared with the desired signal level, once unequal, tamper event is found. When the desired signal is fixed, it is a passive tamper; when the desired signal level is also toggling with time, it is an active tamper. The chip supports at most 10 passive tamper detection pins, or 5 active tamper pairs alternatively. The use of each active tamper pair will reduce the available number of passive tamper pins at most by 2.

An always-ON power supply (coincell battery) should be present in the system. If the tamper detection feature is enabled by software then opening of the tamper contact:

- Switches system power ON with a Tamper Detection alarm interrupt asserted (for software reaction)
- Activate security related hardware (e.g. automatic and immediate erasure of the Zeroizable Master Key and deny access and erase secure memory contents)

11.5 High Assurance Boot (HAB)

HAB, which is the high assurance boot feature in the system boot ROM, detects and prevents execution of unauthorized software (malware) during the boot sequence.

When unauthorized software is permitted to gain control of the boot sequence, it can be exploited for a variety of goals, such as exposing stored secrets; circumventing access controls to sensitive data, services, or networks; or repurposing the platform. Unauthorized software can enter the platform during upgrades or reprovisioning, or when booting from USB connections or removable devices.

HAB protects against unauthorized software by:

- Using digital signatures to recognize authentic software. This allows the user to boot the device to a known initial state, running software signed by the device manufacturer.

11.6 System JTAG Controller (SJC)

The JTAG port provides debug access to hardware blocks, including the ARM processor and the system bus. This allows program control and manipulation as well as visibility to the chip peripherals and memory.

The JTAG port must be accessible during initial platform development, manufacturing tests, and general troubleshooting. Given its capabilities, JTAG manipulation is a known attack vector for accessing sensitive data and gaining control over software execution. System JTAG Controller (SJC) protects against the whole range of attacks based on unauthorized JTAG manipulation. It also provides a JTAG port that conforms to IEEE 1149.1 and IEEE 1149.6 (AC) standards for BSR (boundary scan) testing.

SJC provides the following security levels:

- JTAG Disabled-JTAG use is permanently blocked.
- No-Debug-All security sensitive JTAG features are permanently blocked.
- Secure JTAG-JTAG use is restricted (as in the No-Debug level) unless a secret-key challenge/response protocol is successfully executed.
- JTAG Enabled-JTAG use is unrestricted.

Security levels are selected via e-fuse configuration.

11.7 Bus Encryption Engine (BEE)

The BEE is implemented as an on-the-fly encryption or decryption engine. The main features are following:

- Standard AXI interconnection
- On-the-fly AES-128 encryption/decryption, supporting ECB and CTR mode
- Aliased memory space support and address remapping for up to two individual regions
- Bus access pattern optimization with the aid of local store and forward buffer
- Non-secured access filtering based on security label of the access
- Illegal acess check and filtering

Chapter 12
ARM Cortex A7 Platform (CA7)

12.1 Overview

The Cortex-A7 MPCore platform is a high-performance, low-power processor compliant with the ARMv7-A architecture. The Cortex-A7 MPCore platform has a single ARM®Cortex®-A7 core with a L1 cache subsystem and integrated Generic Interrupt Controller (GIC).

Each Cortex-A7 MPCore includes the following:

- 32 KB L1 Instruction Cache
- 32 KB L1 Data Cache
- Media Processing Engine (MPE) with NEON technology supporting the Advanced Single Instruction Multiple Data version 2 (SIMDv2) architecture
- Floating Point Unit (FPU) with support of the VFPv4-D32 architecture

NOTE

This is a superset of VFPv4-D16

- Support of ARMv7A architecture including:
  - Security extensions for enhanced security
  - Virtualization extensions
  - Large Physical Address (LPA) extension (LPA only supported within Cortex-A7 MPCore platform)

The Cortex-A7 MPCore platform includes the following:

- Integrated Global Interrupt Controller (GIC) configured to support 128 shared peripheral interrupts
- Generic timer
- Snoop control unit (SCU)
- 128 KB unified L2 cache
- Interconnect using a single 128-bit wide bus AMBA AXI bus
- ARMv7.1 ARM debug architecture that complies with the Coresight debug/trace architecture

12.2 External Signals

The following table describes the external signals of ARM:

12.3 Clocks

This section will discuss the Cortex-A7 clocks. For the specification of the maximum Cortex-A7 core frequency, please see the product datasheet. The following table describes the clock sources for ARM. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 12-1. ARM Clocks

Clock name	Clock Root	Description
clk_ahb	ahb_clk_root	AHB Clock
clk_apb_dbg	ahb_clk_root	APB Debug Clock
clk_atb	ahb_clk_root	ATB Clock
ipg_clk	ipg_clk_root	Peripheral Clock
trace_clk_in	ahb_clk_root	Trace Clock

Bus Clocks: The AXI master port of the Cortex-A7 MPCore platform is designed to run at half the speed (1:2 ratio) of the Cortex-A7 core.

Debug Clocks: The APB debug interface is designed to run at half the speed (1:2 ratio) of the Cortex-A7 core.

12.4 Platform Configuration

The revision and configuration of components the Cortex-A7 MPCore platform are detailed below.

Table 12-2. Component Revision

Component	Revision
Cortex-A7 MPCore	MP020-BU-50000-r0p5-00rel0 MP020-BU-50000-r0p5-50rel0
ETM-A7	TM956-BU-50000-r0p0-00rel0

Table 12-3. Cortex-A7 MPCore Global Configuration

Option	Selected Value	Comments
Instruction cache size	32 KB	L1 instruction cache size per core
Data cache size	32 KB	L1 data cache size per core
L2 cache controller	Present	Integrated L2 cache controller
L2 data RAM cycle latency	2 cycles	
Shared Peripheral Interrupts	128	128 shared peripheral interrupts
Number of processors	1	Single Cortex-A7 MPCore
Integrated GIC	True	GIC included

Table 12-4. Cortex-A7 Core-Level Configuration

Option	Selected Value	Comments
NEON and/or FPU	FPU and NEON	FPU and NEON are both included in each processor

12.5 Low-Power and Performance

This section will discuss the low-power and performance features of the Cortex-A7 Core Platform.

The Cortex-A7 MPCore includes the following low-power features:

- Power-efficient processing provided by Cortex-A7 CPU
- 40nm LP process technology
- Flexible power domain partitioning with separate domains for the following blocks:
  - CPU and respective L1 caches
  - SCU and L2 cache controller
  - L2 cache memory
  - Debug including ETM
- Power-efficient timer events using combination of local generic timers and the global system counter
- DVFS to support overdrive and non-overdrive modes of operation

Chapter 13
Analog-to-Digital Converter (ADC)

13.1 Overview

The analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

13.1.1 Features

The features of the ADC are as follows:

- Configuration registers
  - 32-bit, word aligned, byte enabled registers. (byte and halfword access is not supported)
- Linear successive approximation algorithm with up to 12-bit resolution with 10/11 bit accuracy.
- Up to 10 ENOB (dedicated single ended channels)
- Up to 1MS/s sampling rate
- Up to 16 single-ended external analog inputs
- Single or continuous conversion (automatic return to idle after single conversion)
- Output Modes: (in right-justified unsigned format)
  - 12-bit
  - 10-bit
  - 8-bit
- Configurable sample time and conversion speed/power
- Conversion complete and hardware average complete flag and interrupt
- Input clock selectable from up to four sources
- Asynchronous clock source for lower noise operation with option to output the clock
- Selectable asynchronous hardware conversion trigger with hardware channel select
- Selectable voltage reference, internal, external, or alternate
- Automatic compare with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value

- Operation in low power modes for lower noise operation
- Hardware average function
- Self-calibration mode

13.1.2 ADC I/F block diagram

Below is a diagram of the ADC I/F block.

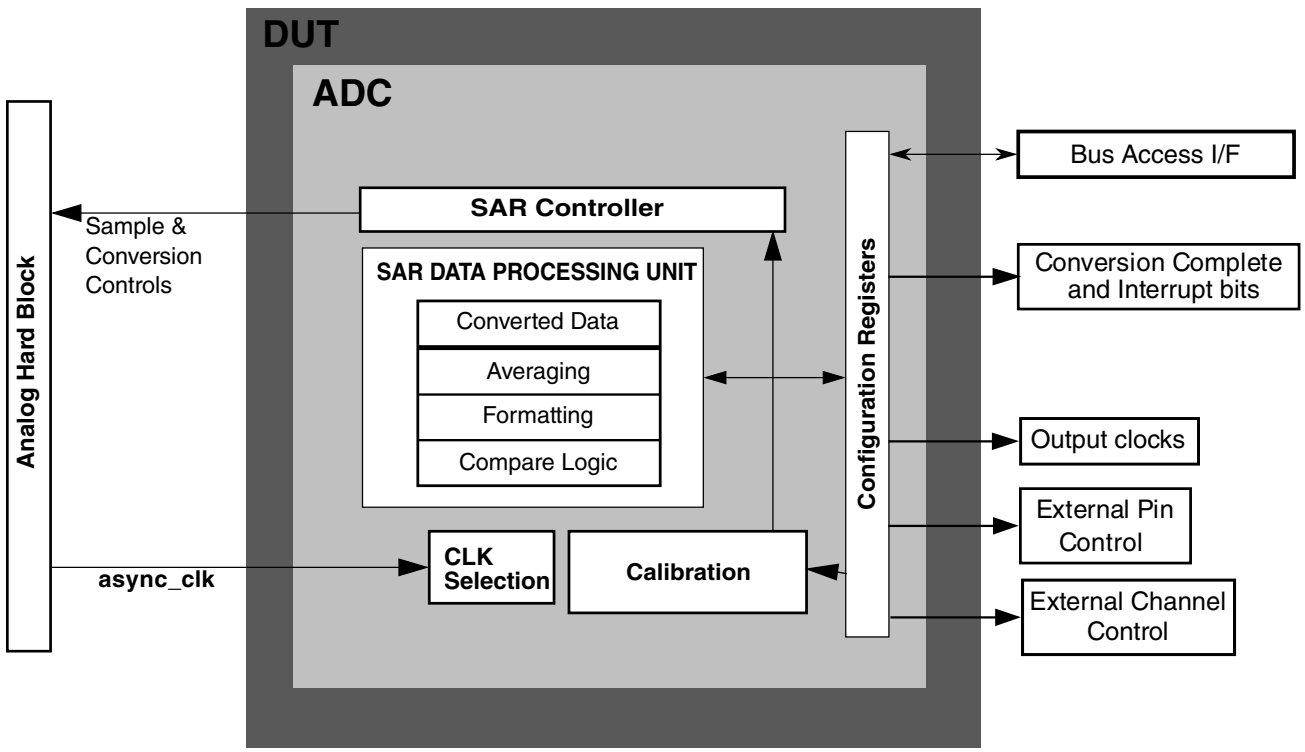


Figure 13-1. ADC-Digital I/F block diagram

13.1.3 ADC block diagram

The following figure shows a top-level block diagram of the ADC.

Figure 13-2. ADC block diagram

NOTE

ADC_VREFL is shorted to ground internally, no dedicate ball for this signal.

13.1.4 ADC module interface

The ADC is connected to many interfaces such as the clocks and reset, access bus, voltage references, interrupt controller, hardware triggers, ADC pin control, and analog I/F as shown in the following figure.

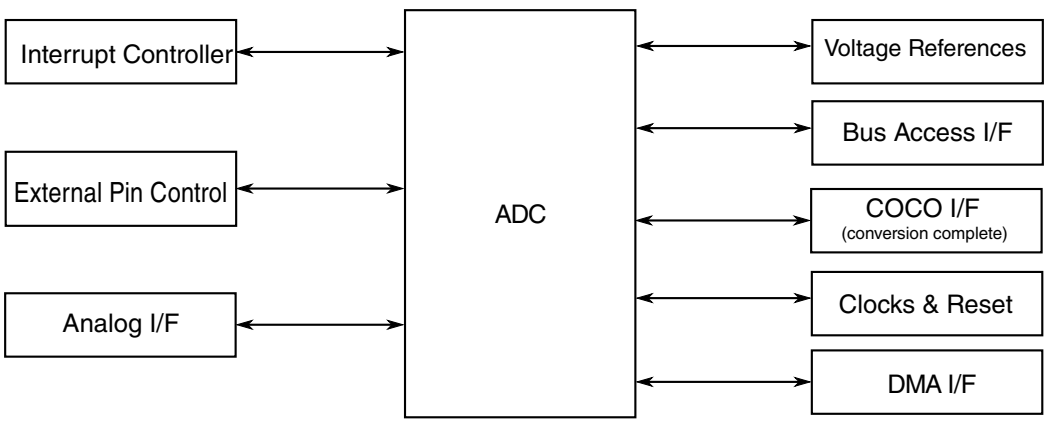


Figure 13-3. ADC module interface

13.1.5 Modes of Operation

By default, the ADC is in disabled mode. In this state, no conversion or other actions occur. All of the ADC control registers are accessible in this state through an access bus interface.To enable the ADC, required configurations should be done by programming the ADC configuration registers.

13.2 External Signals

The following table describes the external signals of ADC:

Table 13-1. ADC External Signals

Signal	Description	Pad	Mode	Direction
ADC_VREFH	Voltage reference high	ADC_VREFH	-	I
ADC1_IN0	Analog channel 1 input 0	GPIO1_IO00	-	I
ADC1_IN1	Analog channel 1 input 1	GPIO1_IO01	-	I

Table continues on the next page...

Table 13-1. ADC External Signals (continued)

Signal	Description	Pad	Mode	Direction
ADC1_IN2	Analog channel 1 input 2	GPIO1_IO02	-	I
ADC1_IN3	Analog channel 1 input 3	GPIO1_IO03	-	I
ADC1_IN4	Analog channel 1 input 4	GPIO1_IO04	-	I
ADC1_IN5	Analog channel 1 input 5	GPIO1_IO05	-	I
ADC1_IN6	Analog channel 1 input 6	GPIO1_IO06	-	I
ADC1_IN7	Analog channel 1 input 7	GPIO1_IO07	-	I
ADC1_IN8	Analog channel 1 input 8	GPIO1_IO08	-	I
ADC1_IN9	Analog channel 1 input 9	GPIO1_IO09	-	I
ADC2_IN0	Analog channel 2 input 0	GPIO1_IO00	-	I
ADC2_IN1	Analog channel 2 input 1	GPIO1_IO01	-	I
ADC2_IN2	Analog channel 2 input 2	GPIO1_IO02	-	I
ADC2_IN3	Analog channel 2 input 3	GPIO1_IO03	-	I
ADC2_IN4	Analog channel 2 input 4	GPIO1_IO04	-	I
ADC2_IN5	Analog channel 2 input 5	GPIO1_IO05	-	I
ADC2_IN6	Analog channel 2 input 6	GPIO1_IO06	-	I
ADC2_IN7	Analog channel 2 input 7	GPIO1_IO07	-	I
ADC2_IN8	Analog channel 2 input 8	GPIO1_IO08	-	I
ADC2_IN9	Analog channel 2 input 9	GPIO1_IO09	-	I

NOTE

The ADC input signals connect to GPIO[0:9]. The GPIO default configuration is enabled for keeper. The keeper causes an undesired jump behavior in ADC. To avoid the problem, disable keeper before starting ADC. For detailed information about keeper, refer to the GPIO block.

13.3 Functional Description

There are three possible states which ADC module can be in:

1. Disabled State
2. Idle state
3. Performing conversions

Disabled State:

The ADC module is disabled during reset, stop mode (if internal clock is not selected as source of clock), or when the ADCH bits of the hardware control (ADC_HC*n*) registers are all high.

Idle State:

The module is idle when a conversion has completed and another conversion has not been initiated. When idle and the asynchronous clock output enable is disabled (ADACKEN = 0), the module is in its lowest power state.

Conversion State:

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm.

To meet accuracy specifications the ADC module must be calibrated using the on chip calibration function. Calibration is recommended to be done after any reset.

When the conversion is completed, the result is placed in the data result registers (ADC_R*n*). The conversion complete flag (COCO*n*) field in the Hardware Status register is/are then set and an interrupt is generated, if the respective conversion complete interrupt has been enabled (ADC_HC*n*[AIEN]=1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of the compare value registers. The compare function is enabled by setting ACFE (ADC Compare Function Enable) in the ADC general control register.

The ADC module has the capability of automatically averaging the result of multiple conversions. The hardware average function is enabled by setting AVGE in the ADC general control register.

13.3.1 Clock Select and Divide Control

The ADC digital module has three clock sources:

- IPG clock
- Alternate Clock (ALTCLK) is connected to the signal as described in the chip-specific ADC information.
- Internal clock (ADACK) is a dedicated clock used only by the ADC.

ADC digital block generates IPG clock/2 by internally dividing the IPG clock. The final clock is chosen from the following clocks.

- IPG clock
- IPG clock divided by 2
- ALTCLK
- ADACK

Out of the above four clocks one is chosen depending on the configuration of ADCLK[1:0] bits of ADC_CFG. This chosen clock is divided depending on the configuration of ADIV[1:0] bits of ADC_CFG. The final generated clock is used as conversion clock for ADC.

ADICLK	Selected Clock Source
00	IPG clock
01	IPG clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

- The IPG clock. This is the default selection following reset.
- The IPG clock divided by two. For higher IPG clock rates, this allows a maximum divide by 16 of the IPG clock with using the ADIV bits.
- ALTCLK, as defined for this chip.
- The asynchronous clock (ADACK). This clock is generated from a clock source within the ADC module. Conversions are possible using ADACK as the input clock source while the MCU is in stop mode.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC may not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

13.3.2 Voltage Reference Selection

The ADC can be configured to use reference pairs as the reference voltages used for conversions (V_{REFSH} and V_{REFSL}). Each pair contains a positive reference which must be between the minimum Ref Voltage High and V_{DDAD}, and a ground reference which must be at the same potential as V_{SSAD}. The pairs can be as follows:

- External (V_{REFH} and V_{REFL})

These voltage references are selected by configuring ADC_CFG[REFSEL].

13.3.3 Hardware Triggering and Channel Selection

The ADC module has a trigger input (known as Alternate Trigger) which provides asynchronous hardware conversion trigger when the ADTRG bit in ADC configuration register (ADC_CFG) is set and any of the external hardware trigger select is high.

To be reliably captured, the Alternate Trigger pulse must be high for sufficient time to satisfy clocking requirement of capturing Flop and the external hardware trigger select event must be set for sufficient time before and after the positive edge of Alternate trigger pulse to meet the setup / hold requirement of capturing flop.

If an external hardware trigger select event gets asserted during a conversion it must stay asserted until end of current conversion and remain set until the receipt of the an Alternate Trigger to initiate a new conversion.

When the Alternate Trigger source is available and hardware triggering is enabled (ADC_CFG[ADTRG]=1), a conversion is initiated on the rising edge of the Alternate Trigger after a external hardware trigger select event has occurred.

If a conversion is in progress when a rising edge of a trigger occurs, the rising edge is ignored. In continuous convertion configuration, only the initial rising edge to launch continuous conversions is observed and until conversion gets aborted the ADC will continue to do conversions on the same ADC Hardware Trigger Control register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The channel selected for the conversion will depend on the settings of active Hardware Trigger register field ADC_HCn[ADCH] of enabled external hardware trigger.

NOTE

Asserting more than one external hardware trigger select signal at the same time will result in unknown results. To avoid this, only select one external hardware trigger select signal prior to the next intended conversion.

When the conversion is completed, the result is placed in the data registers associated with the external hardware trigger received (active trigger selects ADC_Rn). The conversion complete flag associated with the external hardware trigger received (ADC_HS[COCOn]) is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled (ADC_HCn[AIEN]=1).

13.3.4 Conversion Control

Conversions are performed as determined by ADC_CFG[MODE] field.

Conversions can be initiated by either software or hardware triggers. In addition, the ADC can be configured for low power operation, long sample time, continuous conversion, hardware average, and automatic comparison of conversion results with predetermined values.

13.3.4.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADC_HC0 (with ADCHn bits not all 1's) and if software triggered operation is selected (ADTRG=0).
- Following a hardware trigger event if hardware triggered operation is selected (ADTRG=1) and a external hardware trigger select event has occurred. The channel selected will depend on the active trigger select signal active selects ADC_HC1; if neither is active the off condition is selected).

Note

Selecting more than one external hardware trigger select signal (ext_hwts[n]) prior to a conversion completion will generate unknown results. To avoid this, only select one hardware trigger select signal (ext_hwts[n]) prior to a conversion completion.

- Following the transfer of the result to the data registers when continuous conversion is enabled (ADCO=1 in ADC_GC register).

If continuous conversion is enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation (ADTRG=0), continuous conversions begin after ADC_HC0 is written and continue until aborted. In hardware triggered operation(ADTRG=1 and one external hardware trigger select event has occurred), continuous conversions begin after a hardware trigger event and continue until aborted.

If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions is completed. In software triggered operation, conversions begin after ADC_HC0 is written. In hardware triggered operation, conversions begin after a hardware trigger. If continuous conversions is also enabled, a new set of conversions to be averaged are initiated following the last of the selected number of conversions.

13.3.4.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers. (provided the compare function & hardware averaging is disabled), this is indicated by the setting of COCOn. If hardware averaging is enabled, COCOn sets only, if the last of the selected number of conversions is complete. If the compare function is enabled, COCOn sets and conversion result data is transferred only if the compare condition is true. If both hardware averaging and compare functions are enabled, then COCOn sets only if the last of the selected number of conversions is complete and the compare condition is true. An interrupt is generated, if ADC_HCn[AIEN] is high at the time that COCOn is set and if DMAEN is set, DMA request is asserted, if COCOn is set. Both the requests get deasserted when COCOn is low, cleared, which happens when data is read.

In all modes a blocking mechanism prevents a new result from overwriting previous data in ADC_Rn, if the previous data is in the process of being read. When blocking is active (OVWREN=0 in ADC_CFG), the conversion result data transfer is blocked, COCOn is not set, and the new result is lost. In all other cases of operation, when a conversion result data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

Note

If continuous conversions are enabled, the blocking mechanism could result in the loss of data occurring at specific timepoints. To avoid this issue, the data must be read in fewer cycles than an ADC conversion time, accounting for interrupt or software polling loop latency.

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

13.3.4.3 Aborting Conversions

Any conversion in progress is aborted when:

- The MCU enters stop mode with ADACK not enabled.
- In software trigger mode, write to ADC_HC0 register, while ADC_HC0 is actively (already) controlling a conversion, aborts the current conversion. Since, any of the ADC_HC1 - ADC_HCn registers are not used for software trigger operation therefore, write to any of them neither initiate a new conversion nor abort the software triggered active conversion.
- In hardware trigger mode, write to any one of the ADC_HC0 - ADC_HCn register, while that specific ADC_HC0 - ADC_HCn register is actively controlling a conversion, aborts the current conversion.
- A write to any ADC register besides the ADC_HC0: ADC_HCn registers occurs . This indicates a mode of operation change has occurred and the current conversion is therefore invalid.

Note

When a conversion is aborted, the contents of the data result registers, ADCRn are not altered. The data result registers continue to hold the values, transferred after the completion of the last successful conversion. If the conversion is aborted by a reset or stop (not operated with internal ADACK), ADCRn (data result register) return to their reset states.

13.3.4.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source but the asynchronous clock output is disabled (ADACKEN=0), the ADACK clock generator will also remain in its idle state (disabled) until a conversion is initiated. If the asynchronous clock output is enabled (ADACKEN=1), it will remain active regardless of the state of the ADC or the MCU power mode.

Power consumption when the ADC is active can be reduced by setting ADLPC..

13.3.4.5 Sample Time and Total Conversion Time

The total conversion time depends upon the following:

- the sample phase time (as determined by ADLSMP and ADSTS bits in ADC_CFG register),
- the compare phase time (determined by MODE bits)
- the frequency of the conversion clock (fADCK).
- the MCU bus frequency (for Handshaking and selection of clock)

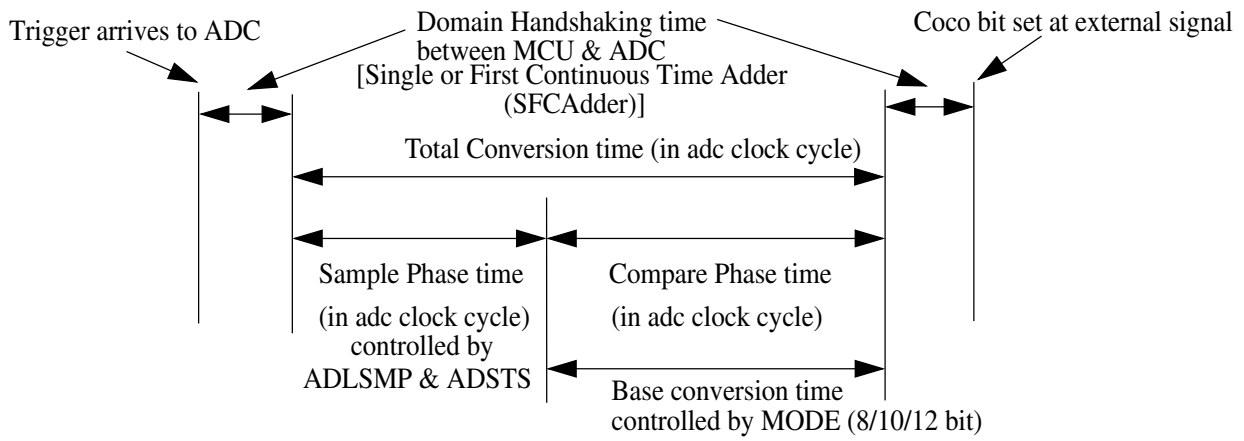


Figure 13-4. ADC conversion time details

After the module becomes active, sampling of the input begins. ADLSMP and ADSTS decide the sample time duration. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADC_Rn upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed .

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits in ADC_CFG register, and the divide ratio is specified by the ADIV bits.

The maximum total conversion time for all configurations is summarized in Equation 1 on page 392. Refer to Table 13-2 through Table 13-5 for the variables referenced in the equation.

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder})$$

Equation 1

Table 13-2. Single or First Continuous Time Adder (SFCAdder)

ADACKEN	ADICLK	Single or First Continuous Time Adder (SFCAdder)
x	0x, 10	3 ADCK cycles (before starting of conversion) + 1ADCK (after end of conversion) + 2 bus clock cycles
1	11	3 ADCK cycles (before starting of conversion) + 1 ADCK (after end of conversion) +

Table continues on the next page...

Table 13-2. Single or First Continuous Time Adder (SFCAdder) (continued)

ADACKEN	ADICLK	Single or First Continuous Time Adder (SFCAdder)
		2 bus clock cycles
0	11	1.5µs + 3 ADCK cycles (before starting of conversion) + 1 ADCK (after end of conversion) + 2 bus clock cycles

Table 13-3. Average Number Factor (AverageNum)

AVGE	AVGS[1:0]	Average Number Factor (AverageNum)
0	xx	1
1	00	4
1	01	8
1	10	16
1	11	32

Table 13-4. Base Conversion Time (BCT) (compare phase duration)

Mode	Base Conversion Time (BCT) (compare phase duration)
8 bit	17 ADCK cycles
10 bit	21 ADCK cycles
12 bit	25 ADCK cycles

Table 13-5. Long Sample Time

ADLSMP	ADSTS	Long Sample Time Adder (LSTAdder)
0	00	3 ADCK cycles
0	01	5 ADCK cycles
0	10	7 ADCK cycles (default)
0	11	9 ADCK cycles
1	00	13 ADCK cycles
1	01	17 ADCK cycles
1	10	21 ADCK cycles
1	11	25 ADCK cycles

Note

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

13.3.4.6 Conversion Time Examples

The following examples uses Equation 1 on page 392 and the information provided in tables Table 13-2 through Table 13-5.

13.3.4.6.1 Typical conversion time configuration

A typical configuration for ADC conversion is: 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 40 MHz, ADLSMP=0,ADLSTS=10 and high speed conversion disabled. The conversion time for a single conversion is calculated by using Equation 1 on page 392 and the information provided in Table 13-6 through Table 13-8. The table below list the variables of Equation 1 on page 392.

Table 13-6. Typical Conversion Time

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
AverageNum	1
BCT	21 ADCK cycles
LSTAdder	7

The resulting conversion time is generated using the parameters listed in Table 13-6. So for Bus clock equal to 40 Mhz and ADCK equal to 40 Mhz the resulting conversion time is 0.95 us.

13.3.4.6.2 Long conversion time configuration

A configuration for long ADC conversion is: 12-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-8 ratio selected, and a bus frequency of 40 MHz, long sample time enabled (ADLSMP=1, ADSTS=11) and configured for longest adder and high speed conversion disabled. Average enabled for 32 conversions (AVGE=1, AVGS=11). The conversion time for this conversion is calculated by using equation on Sample Time and Total Conversion Time and the information provided in Table 13-2 through Table 13-5. The table below lists the variables of equation.

Table 13-7. Typical Conversion Time

Variable	Time
SFCAdder	3 ADCK cycles + 5 bus clock cycles
AverageNum	32
BCT	25 ADCK cycles
LSTAdder	25 ADCK cycles

The resulting conversion time is generated using the parameters listed in Table 13-7. So for Bus clock equal to 40 Mhz and ADCK equal to 5 Mhz the resulting conversion time is 10.0226 us (AverageNum). This results in a total conversion time of 320.725 us.

13.3.4.6.3 Short conversion time configuration

A configuration for short ADC conversion is: 8-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 40 MHz, long sample time disabled(ADLSMP=0, ADSTS=00) and high speed conversion enabled. The conversion time for this conversion is calculated by using the equation and the information provided in Table 13-2 to Table 13-5. The table below list the variables of equation.

Table 13-8. Typical Conversion Time

Variable	Time
SFCAdder	3 ADCK cycles + 5 bus clock cycles
AverageNum	1
BCT	17 ADCK cycles
LSTAdder	3 ADCK cycles

The resulting conversion time is generated using the parameters listed in Table 13-8. So for Bus clock equal to 40Mhz and ADCK equal to 40Mhz the resulting conversion time is 700 ns.

13.3.4.7 Hardware Average Function

The hardware average function can be enabled (AVGE=1) to perform a hardware average of multiple conversions. The number of conversions is determined by the AVGS[1:0] bits, which select 4, 8, 16 or 32 conversions to be averaged. While the hardware average function is in progress the ADACT bit will be set.

After the selected input is sampled and converted, the result is placed in an accumulator from which an average is calculated once the selected number of conversions has been completed. When hardware averaging is selected the completion of a single conversion will not set the COCOn bit.

If the compare function is either disabled or evaluates true, after the selected number of conversions are completed, the average conversion result is transferred into the data result registers, ADC_Rn , and the COCOn bit is set. An ADC interrupt is generated upon the setting of COCOn if the respective ADC interrupt is enabled (AIENn=1).

13.3.5 Automatic Compare Function

The compare function can be configured to check if the result is less than or greater-than-or-equal-to a single compare value, or if the result falls within or outside a range determined by two compare values. The compare mode is determined by ACFGT, ACREN and the values in the compare value register (ADC_CV). After the input is sampled and converted, the compare values (CV1 and CV2) are used as described in the table below. There are six compare modes as shown in the table below.

Table 13-9. Compare Modes

ACFGT	ACREN	CV1 relative to CV2	Function	Compare Mode Description
0	0	-	Less than threshold	Compare true if the result is less than the CV1 registers.
1	0	-	Greater than or equal to threshold	Compare true if the result is greater than or equal to CV1 registers.
0	1	Less than or equal	Outside range, not inclusive	Compare true if the result is less than CV1 **Or** the result is Greater than CV2
0	1	Greater than	Inside range, not inclusive	Compare true if the result is less than CV1 **And** the result is greater than CV2
1	1	Less Than or equal	Inside range, inclusive	Compare true if the result is greater than or equal to CV1 **And** the result is less than or equal to CV2
1	1	Greater than	Outside range, inclusive	Compare true if the result is greater than or equal to CV1 **Or** the result is less than or equal to CV2

With the ADC range enable bit set, ADCREN =1, if compare value 1(CV1 value) is less than or equal to the compare value 2 (CV2 value), setting ACFGT will select a trigger-if-inside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-outside-compare-range, not-inclusive-of-endpoints function.

If CV1 is greater than the CV2, setting ACFGT will select a trigger-if-outside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-inside-compare-range, not-inclusive-of-endpoints function.

If the condition selected evaluates true, COCOn is set.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCOn is not set and the conversion result data will not be transferred to the result register. If the hardware averaging function is enabled, the compare function compares the averaged result to the compare values. The same compare function definitions apply. An ADC interrupt is generated upon the setting of COCOn if the respective ADC interrupt is enabled (ADC_HCn[AIEN]=1).

Note

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

13.3.6 Calibration Function

The ADC contains a self-calibration function that is required to achieve the specified accuracy. Calibration should be run or valid calibration values should be written after power up and system reset (as the calibration register will be reset on reset assertion) with specified settings before any conversion is initiated. The calibration function sets the calibration value at the end of running the full calibration sequence in ADC_CAL register. The user must configure the ADC correctly prior to starting the calibration process, and must allow the process to run the full calibration sequence by checking the status of ADC_GC[CAL] and ADC_GS[CALF] so that the generated calibration value can be loaded.

Prior to calibration, the user must configure the ADC's clock source and frequency, low power configuration, voltage reference selection, sample time, averaging, and the high speed configuration according to the application's clock source availability and needs. If the application uses the ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected, or multiple calibrations can be done for the different configurations. The input channel, conversion mode, continuous function and compare function are all ignored during the calibration process.

To initiate calibration, the user sets the CAL bit and the calibration will automatically begin if the ADTRG bit = 0. If ADTRG = 1, the CAL bit will not get set and the calibration fail flag (CALF) will be set. While calibration is active, no ADC register can be written and no stop mode may be entered or the calibration routine will be aborted causing the CAL bit to clear and the CALF bit to set.

At the end of a calibration sequence the COCO[0] bit of the ADC_HS register will be set. The ADC_HCn[AIEN] bit can be used to allow an interrupt to occur at the end of a calibration sequence. If, at the end of calibration routine, the CALF bit is not set, the automatic calibration routine completed successfully.

To complete calibration, the user must follow the below procedure :

- Configure ADC_CFG with actual operating values for maximum accuracy.
- Configure the ADC_GC values along with CAL bit
- Check the status of CALF bit in ADC_GS and the CAL bit in ADC_GC
- When CAL bit becomes '0' then check the CALF status and COCO[0] bit status

When complete the user may reconfigure and use the ADC as desired.

A second calibration may also be performed if desired by clearing and again setting the CAL bit

Overall the calibration routine may take as many as 14000 ADCK cycles and 100 bus cycles, depending on the results and the clock source chosen.

13.3.7 User Defined Offset Function

The ADC Offset Correction Register (ADC_OFS) contains the user configured offset value. This register is 13 bit wide. The value in MSB (13th bit) is the operation bit, if this bit is ‘0’ then the value in rest 12 bit is added with the converted result value to generate final result to be loaded into ADC_Rn and if this bit is ‘1’ then this field is subtracted from converted value to generate final Result (ADC_Rn). If the Final result is above the maximum or below the minimum result value, it is forced to the appropriate limit for the current mode of operation. Forced to 0x0FFF if over and 0x0000 if lower for 12 bit mode.

The offset value has no effect during calibration and BIST test on final result.

The formatting of the ADC Offset Register is different from the Data Result Registers (ADC_Rn) to preserve the resolution of the value regardless of the conversion mode selected. Lower order bits are ignored in lower resolution modes. For example, in 8b single-ended mode, the bits OFS[11:4] are subtracted from D[7:0] when bit OFS[12] (sign bit) is ‘1’ ; indicates subtraction and bits OFS[4:0] are ignored. For 12b single-ended mode, bits OFS[11:0] are directly subtracted from the conversion result data CDATA[11:0] when OFS[12] (sign bit) is ‘1’. The similar is the addition operation when OFS[12](sign bit) is 0.

ADC_OFS is manually set according to user requirements once the self calibration sequence is done (CAL is cleared). The user have to write ADC_OFS with desired value.

NOTE

There is an effective limit to the values of Offset that can be set by the user. If the magnitude of the offset is too great the results of the conversions will cap off at the limits.

The offset function may be employed by the user to remove application offsets or DC bias values. An offset correction that results in an out-of-range value will be forced to the minimum or maximum value.

For applications which may change the offset repeatedly during operation, it is recommended to store the initial offset value in flash so that it can be recovered and added to any user offset adjustment value and the sum stored in the ADC_OFS registers.

13.3.8 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs (ADC_HC*n*[AIEN] = 010000b = 16d). This channel can be selected through configuring the ADCH*n* bits of any ADC_HC*n* register. Any trigger on this will give a temperature value converted for further processings.

13.3.9 MCU Wait Mode Operation

Wait mode is a **lower power-consumption standby mode** from which **recovery is fast** because **the clock sources remain active**. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for the specific MCU.

A conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from wait mode.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from wait mode if the respective ADC interrupt is enabled (ADC_HCn[AIEN]=1).

If the hardware averaging function is enabled the COCOn will set (and generate an interrupt if enabled) when the selected number of conversions are complete.

If the compare function is enabled the COCOn will set (and generate an interrupt if enabled) only if the compare conditions are met.

If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from wait mode unless a new conversion is initiated by the hardware trigger.

13.3.10 MCU Stop Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled. Stop mode entered when stop indication comes from the MCU.

13.3.10.1 Stop Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of the ADC registers, including ADC_Rn are unaffected by stop mode. After exiting from stop mode, a software or hardware trigger is required to resume conversions.

13.3.10.2 Stop Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop mode.

If a conversion is in progress when the MCU enters stop mode, it continues until completion. Conversions can be initiated while the MCU is in stop mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from stop mode :

Note

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the conversion result data

transfer blocking mechanism (discussed in Completing Conversions) is cleared when entering stop and continuing ADC conversions.

13.4 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. User can configure the module for 8, 10, 12 bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options.

13.4.1 ADC Module Initialization Example

This section describes the initialization sequence along with pseudo-code.

13.4.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

- Calibrate the ADC by following the calibration instructions in Calibration Function
- Update the configuration register (ADC_CFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
- Update Genral control register (ADC_GC) to select whether conversions will be continuous or completed only once (ADCO) and to select whether to perform hardware averaging, etc.
- Update Trigger control register (ADC_HCn) to select the conversion trigger (hardware or software, i.e. configure ADTRG bit) and compare function options, if enabled.

13.4.1.2 Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

ADC_CFG

```
Bit 7    ADLPC   1      Configures for low power (lowers maximum clock speed.
Bit 6:5  ADIV    00     Sets the ADCK to the input clock  ÷  1.
Bit 4    ADLSMP  1      Configures for long sample time.
Bit 3:2  MODE    10     Sets mode at 10-bit conversions.
Bit 1:0  ADICLK  00     Selects bus clock as input clock source.
```

ADC_GC

```
Bit 7    CAL     0      Flag indicates if a conversion is in progress.
Bit 6    ADCO    0      Software trigger selected.
Bit 5    AVGE    0      Compare function disabled.
Bit 4    ACFE    0      Compare function disabled.
Bit 3    ACFGT   0      Not used in this example.
Bit 2    ACREN   0      Not used in this example.
Bit 1    DMAEN   0      Not used in this example.
Bit 0    ADACKEN 0      Not used in this example.
```

ADC_HC0

```
   Bit 7    AIEN   1      Conversion complete interrupt enabled.
Bit 4:0  ADCH   00001  Input channel 1 selected as ADC input channel.
```

ADC_R0

Holds results of conversion. Read high byte (ADCRHA) before low byte (ADCRLA) so that conversion data cannot be overwritten with data from the next conversion.

ADC_CV

Holds compare values when compare function enabled.

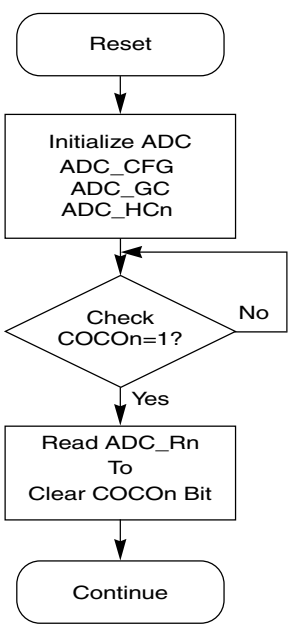


Figure 13-5. Initialization Flowchart for Example

13.5 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

13.5.1 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

13.5.1.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 1.3 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the nominal sample window (6 cycles @ 40 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 4 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP and changing the ADSTS bits (to increase the sample window) or decreasing ADCK frequency to increase sample time.

13.5.1.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDAD} / (2^N * I_{LEAK})$ for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

13.5.1.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 µF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 µF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from V_{DDAD} to V_{SSAD}.
- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
    - For software triggered conversions, immediately follow the write to ADCSC1 with a wait instruction or stop instruction.
    - For stop mode operation, select ADACK as the clock source. Operation in stop reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 µF capacitor (CAS) on the selected input channel to V_{REFL} or V_{SSAD} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

13.5.1.4 Code Width and Quantization Error

Note

This will remain the same as long as the result is rounded for 8 and 10-bit modes. If the result is truncated in 8/10b modes then they will match 12b mode where the quantization error is -1 to 0

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1\text{lsb} = (V_{\text{REFH}} - V_{\text{REFL}}) / 2^N$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is -1 lsb to 0 lsb and the code width of each step is 1 lsb.

13.5.1.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.

- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

13.5.1.6 Code Jitter, Non-Monotonicity, and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Noise-Induced Errors reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

13.6 Memory map and register definition

The ADC-Digital contains 32-bit, word aligned, byte enables registers; byte or half word access are not supported. All configuration registers are accessible via 32-bit access bus Interface. Write access to reserved locations have no impact while read access to reserved locations always return 0.

NOTE

No protection or indication mechanism is available (for example, 32-bit access starting with address offset value 0x01 or 0x02 or 0x03). The ADC does not check for correctness of the programmed values in the registers and the programmer must ensure that correct values are being written.

ADC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_8000	Control register for hardware triggers (ADC1_HC0)	32	R/W	See section	13.6.1/408
219_8004	Control register for hardware triggers (ADC1_HC1)	32	R/W	See section	13.6.2/409
219_8008	Status register for HW triggers (ADC1_HS)	32	R (reads 0)	See section	13.6.3/411
219_800C	Data result register for HW triggers (ADC1_R0)	32	R/W	See section	13.6.4/412
219_8010	Data result register for HW triggers (ADC1_R1)	32	R/W	See section	13.6.5/413
219_8014	Configuration register (ADC1_CFG)	32	R/W	See section	13.6.6/414
219_8018	General control register (ADC1_GC)	32	R/W	See section	13.6.7/416
219_801C	General status register (ADC1_GS)	32	R/W	See section	13.6.8/418
219_8020	Compare value register (ADC1_CV)	32	R/W	See section	13.6.9/419
219_8024	Offset correction value register (ADC1_OFS)	32	R/W	See section	13.6.10/420
219_8028	Calibration value register (ADC1_CAL)	32	R/W	See section	13.6.11/421
219_C000	Control register for hardware triggers (ADC2_HC0)	32	R/W	See section	13.6.1/408
219_C004	Control register for hardware triggers (ADC2_HC1)	32	R/W	See section	13.6.2/409
219_C008	Status register for HW triggers (ADC2_HS)	32	R (reads 0)	See section	13.6.3/411
219_C00C	Data result register for HW triggers (ADC2_R0)	32	R/W	See section	13.6.4/412
219_C010	Data result register for HW triggers (ADC2_R1)	32	R/W	See section	13.6.5/413
219_C014	Configuration register (ADC2_CFG)	32	R/W	See section	13.6.6/414
219_C018	General control register (ADC2_GC)	32	R/W	See section	13.6.7/416
219_C01C	General status register (ADC2_GS)	32	R/W	See section	13.6.8/418
219_C020	Compare value register (ADC2_CV)	32	R/W	See section	13.6.9/419
219_C024	Offset correction value register (ADC2_OFS)	32	R/W	See section	13.6.10/420
219_C028	Calibration value register (ADC2_CAL)	32	R/W	See section	13.6.11/421

13.6.1 Control register for hardware triggers (ADCx_HC0)

ADC_HC0 can be used for both software and hardware trigger mode. Other ADC_HC*n* (*n* = 1...) are for use only in hardware trigger mode. The ADC_HC0 to ADC_HC*n* (*n* = 0,1) registers have identical fields, and are used to control ADC operation. At any one point in time, only one of the ADC_HC0 to ADC_HC*n* (*n* = 0,1) registers is actively controlling ADC conversions. Updating ADC_HC0 while ADC_HC*n* (*n* = 0,1) is actively controlling a conversion is allowed (and vice-versa for any of the ADC_HC*n* (*n* = 0,1) registers). Writing ADC_HC0 while ADC_HC0 is actively controlling a conversion aborts the current conversion. In software trigger mode (ADTRG=0), writes to ADC_HC0 subsequently initiates a new conversion (if the ADCH bits are equal to a value other than all 1s). Similarly, writing any of the ADC_HC*n* (*n* = 0,1) registers while that specific ADC_HC register is actively controlling a conversion aborts the current conversion. ADC_HC1 register is not used for software trigger operation and therefore writes to any of them do not initiate a new conversion.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0															

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

ADCx_HC0 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 AIEN	Conversion Complete Interrupt Enable/Disable Control An interrupt is generated whenever ADC_HS[COCO0]=1 (conversion ADC_HC0 completed), provided the corresponding interrupt is enabled. 1 Conversion complete interrupt enabled 0 Conversion complete interrupt disabled
6–5 Reserved	This field is reserved.
ADCH	Input Channel Select This 5-bit field selects one of the input channels. The successive approximation converter subsystem is turned off when the channel select bits are all set (ADCH = 11111b). This feature allows for explicit

Table continues on the next page...

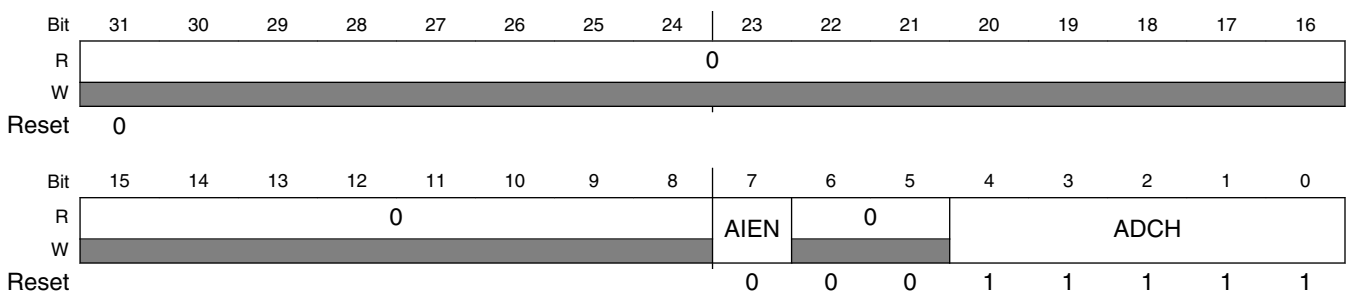
ADCx_HC0 field descriptions (continued)

Field	Description
	disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional single conversion from being performed. 00000-01111 External channels 0 to 15. 10000-10111 Reserved 11000 Reserved. 11001 VREFSH = internal channel, for ADC self-test, hard connected to VRH internally 11010 Reserved. 11011 Reserved. 11100-11110 Reserved. 11111 Conversion Disabled. Hardware Triggers will not initiate any conversion.

13.6.2 Control register for hardware triggers (ADCx_HC1)

ADC_HC1 are for use only in hardware trigger mode. The ADC_HC0 to ADC_HCn registers have identical fields, and are used to control ADC operation. At any one point in time, only one of the ADC_HC0 to ADC_HCn registers is actively controlling ADC conversions. Updating ADC_HC0 while ADC_HCn is actively controlling a conversion is allowed (and vice-versa for any of the ADC_HCn registers). Writing any of the ADC_HCn registers while that specific ADC_HCn register is actively controlling a conversion aborts the current conversion. Any of the ADC_HC1 - ADC_HCn registers are not used for software trigger operation and therefore writes to any of them do not initiate a new conversion.

Address: Base address + 4h offset



ADCx_HC1 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

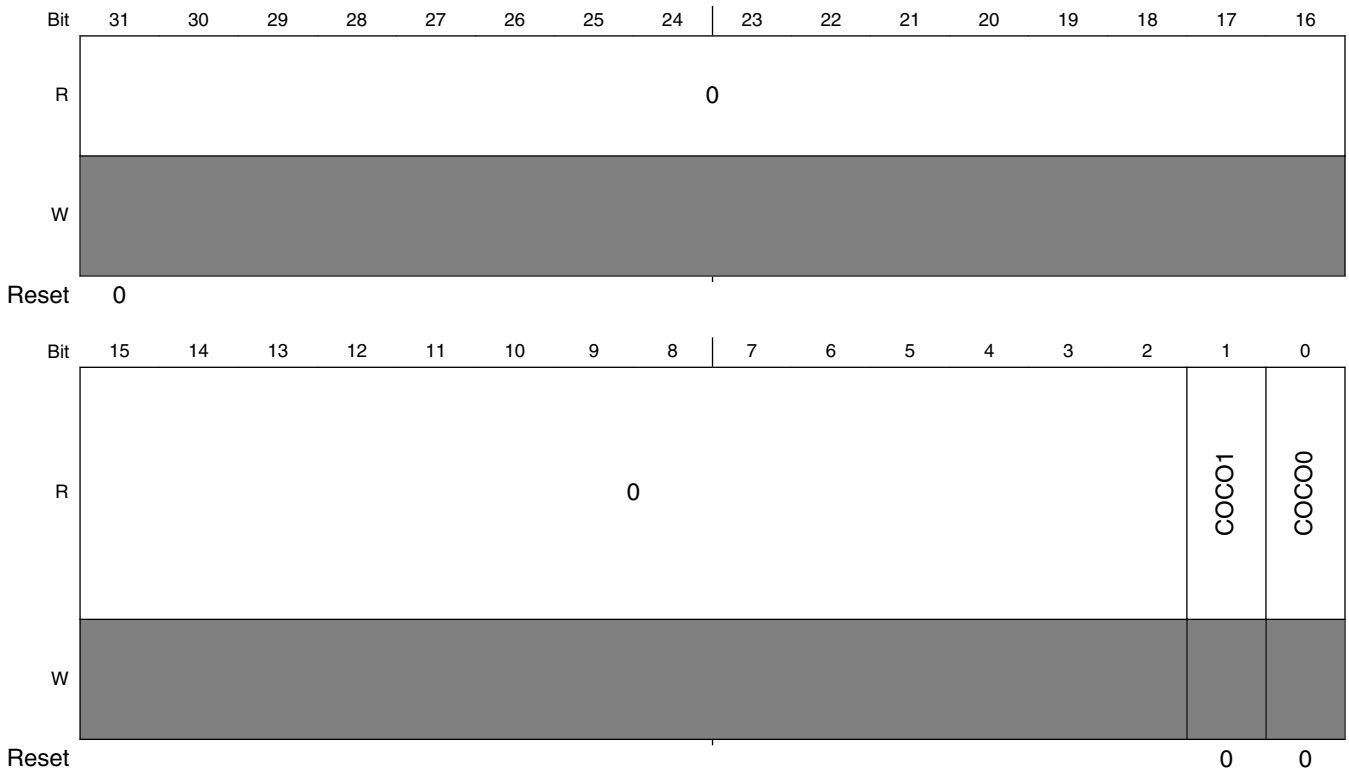
ADCx_HC1 field descriptions (continued)

Field	Description
7 AIEN	Conversion Complete Interrupt Enable/Disable Control An interrupt is generated whenever ADC_HS[COCO0]=1(conversion ADC_HC0 completed), provided the corresponding interrupt is enabled. 1 Conversion complete interrupt enabled 0 Conversion complete interrupt disabled
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ADCH	Input Channel Select This 5-bit field selects one of the input channels. The successive approximation converter subsystem is turned off when the channel select bits are all set (ADCH =11111b). This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional single conversion from being performed. 00000-01111 External channels 0 to 15. 10000-10111 Reserved 11000 Reserved. 11001 VREFSH = internal channel, for ADC self-test, hard connected to VRH internally 11010 Reserved. 11011 Reserved. 11100-11110 Reserved. 11111 Conversion Disabled. Hardware Triggers will not initiate any conversion.

13.6.3 Status register for HW triggers (ADCx_HS)

Bit 0 is used for both software and hardware trigger modes of operation. Bit 1 to bit (n-1) indicate the rest of the HW triggers' statuses similar to bit 0, potentially corresponding to multiple ADC_HC registers (for use only in hardware trigger mode).

Address: Base address + 8h offset



ADCx_HS field descriptions

Field	Description
31–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 COCO1	See description for COCO0.
0 COCO0	Conversion Complete Flag The COCOn flag is a read-only bit that is set each time a conversion is completed when the compare function is disabled (ADC_GC[ACFE]=0) and the hardware average function is disabled (ADC_GC[AVGE]=0). When the compare function is enabled (ADC_GC[ACFE]=1), the COCOn flag is set upon completion of a conversion only if the compare result is true. When the hardware average function is enabled (ADC_GC[AVGE]=1), the COCOn flag is set upon completion of the selected number of conversions (determined by the ADC_CFG[AVGS] field). The COCO0 flag will also set at the completion of a Calibration and Test sequence. A COCOn bit is cleared when the respective ADC_HCn is written or when the respective ADC_Rn is read.

13.6.4 Data result register for HW triggers (ADCx_R0)

Contains the result of an ADC conversion of the channel selected by the respective hardware trigger and channel control register (ADC_HC0:ADC_HCn). For every ADC_HC0:ADC_HCn status and channel control register, there is a respective ADC_R0:ADC_Rn data result register. Unused bits in the ADC_Rn register are cleared in unsigned right justified modes. For example when configured for 10-bit single-ended mode, D[31:10] are cleared. The table below describes the behavior of the data result registers in the different modes of operation.

Table 13-10. Data Result Register Description

Conversion Mode	Data Result Register bits																Format
	D31	D30	.....	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
12b single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
10b single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	unsigned right justified
8b single-ended	0	0	0	0	0	0	0	0	D	D	D	D	D	D	D	D	unsigned right justified

Address: Base address + Ch offset

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

ADCx_R0 field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CDATA	Data (result of an ADC conversion)

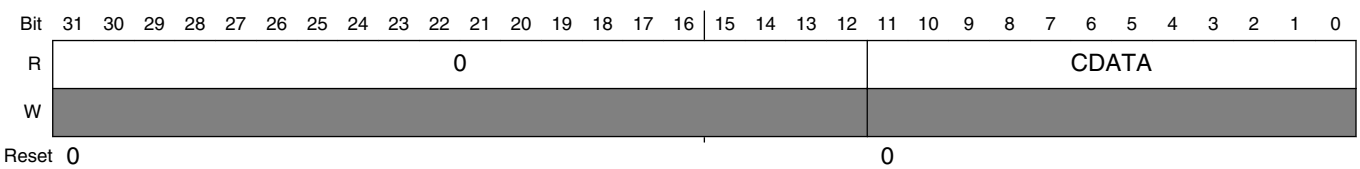
13.6.5 Data result register for HW triggers (ADCx_R1)

Contains the result of an ADC conversion of the channel selected by the respective Hardware Trigger and channel control register (ADC_HC0:ADC_HCn). For every ADC_HC0:ADC_HCn status and channel control register, there is a respective ADC_R0 to ADC_Rn data result register. Unused bits in the ADC_Rn register are cleared in unsigned right justified modes. For example when configured for 10-bit single-ended mode, D[31:10] are cleared. The table below describes the behavior of the data result registers in the different modes of operation.

Table 13-11. Data Result Register Description

Conversion Mode	Data Result Register bits																Format
	D31	D30	.....	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
12b single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
10b single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	unsigned right justified
8b single-ended	0	0	0	0	0	0	0	0	D	D	D	D	D	D	D	D	unsigned right justified

Address: Base address + 10h offset



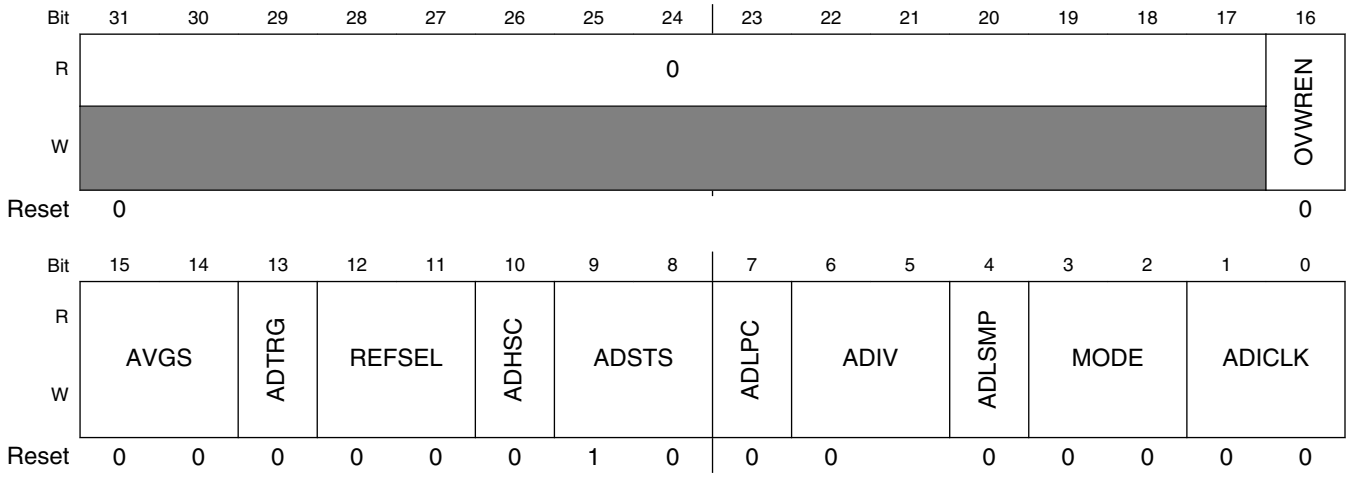
ADCx_R1 field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CDATA	Data (result of an ADC conversion)

13.6.6 Configuration register (ADCx_CFG)

Selects the mode of operation, clock source, clock divide, configure for low power, long sample time, high speed configuration and selects the sample time duration.

Address: Base address + 14h offset



ADCx_CFG field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 OVWREN	Data Overwrite Enable Controls the overwriting of the next converted Data onto the existing (previous) unread data into the Data result register. 1 Enable the overwriting. 0 Disable the overwriting. Existing Data in Data result register will not be overwritten by subsequent converted data.
15–14 AVGS	Hardware Average select Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when ADC_GC[AVGE] = 1. 00 4 samples averaged 01 8 samples averaged 10 16 samples averaged 11 32 samples averaged
13 ADTRG	Conversion Trigger Select Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADC_HC0. When hardware trigger is selected, a conversion is initiated following the assertion of a pulse on Alternate Hardware trigger input along with the assertion of the enable of respective the hardware Triggers input .

Table continues on the next page...

ADCx_CFG field descriptions (continued)

Field	Description
	0 Software trigger selected 1 Hardware trigger selected
12–11 REFSEL	Voltage Reference Selection Selects the voltage reference source used for conversions (see the chip-specific SARADC information for details). 00 Selects VREFH/VREFL as reference voltage. 01 Reserved 10 Reserved 11 Reserved
10 ADHSC	High Speed Configuration This bit configures the ADC for high speed operation. The internal ADC clock is higher than normal. 0 Normal conversion selected. 1 High speed conversion selected.
9–8 ADSTS	Defines the sample time duration. This has two modes, short and long. When long sample time is selected (ADLSMP=1) this works for long sample time otherwise this works for short sample. This allows higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 00 Sample period (ADC clocks) = 2 if ADLSMP=0b Sample period (ADC clocks) = 12 if ADLSMP=1b 01 Sample period (ADC clocks) = 4 if ADLSMP=0b Sample period (ADC clocks) = 16 if ADLSMP=1b 10 Sample period (ADC clocks) = 6 if ADLSMP=0b Sample period (ADC clocks) = 20 if ADLSMP=1b 11 Sample period (ADC clocks) = 8 if ADLSMP=0b Sample period (ADC clocks) = 24 if ADLSMP=1b
7 ADLPC	Low-Power Configuration Puts the ADC hard block into low power mode and reduces the comparator enable period by controlling its timing in the SAR controller block towards the anlong hard block. The signal indicating low power mode to the Analog block is asserted when this bit is set. 0 ADC hard block not in low power mode. 1 ADC hard block in low power mode.
6–5 ADIV	Clock Divide Select Selects the divide ratio used by the ADC to generate the internal clock ADCK. 00 Input clock 01 Input clock / 2 10 Input clock / 4 11 Input clock / 8

Table continues on the next page...

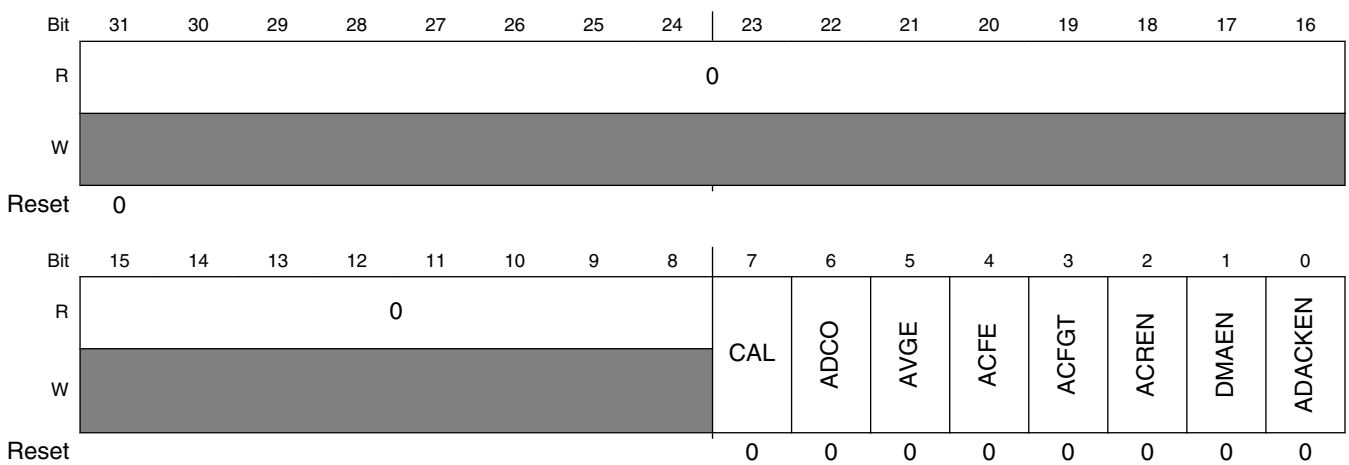
ADCx_CFG field descriptions (continued)

Field	Description
4 ADLSMP	Long Sample Time Configuration Selects between different sample times based on the ADC_CFG[ADSTS] field. This bit adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. If high conversion rates are not required, longer sample times can also be used to lower overall power consumption when continuous conversions are enabled. When ADLSMP=1, the Long Sample Time mode is selected and the time is defined by ADSTS[1:0] of the ADC_CFG register. 0 Short sample mode. 1 Long sample mode.
3–2 MODE	Conversion Mode Selection Used to set the ADC resolution mode. 00 8-bit conversion 01 10-bit conversion 10 12-bit conversion 11 Reserved
ADICLK	Input Clock Select Selects the input clock source to generate the internal clock ADCK. 00 IPG clock 01 IPG clock divided by 2 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

13.6.7 General control register (ADCx_GC)

Controls the calibration, continuous convert, hardware averaging functions, conversion active, hardware/software trigger select, compare function and voltage reference select of the ADC module.

Address: Base address + 18h offset



ADCx_GC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CAL	Calibration CAL begins the calibration sequence when set. This bit stays set while the calibration is in progress and is cleared when the calibration sequence is complete. The ADC_GS[CALF] bit must be checked to determine the result of the calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and the ADC_GS[CALF] bit will set. Setting the CAL bit will abort any current conversion.
6 ADCO	Continuous Conversion Enable Enables continuous conversions. 0 One conversion or one set of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion.
5 AVGE	Hardware average enable Enables the hardware average function of the ADC. 0 Hardware average function disabled 1 Hardware average function enabled
4 ACFE	Compare Function Enable Enables the compare function. 0 Compare function disabled 1 Compare function enabled
3 ACFGT	Compare Function Greater Than Enable Configures the compare function to check the conversion result relative to the compare value register (ADC_CV) based upon the value of ACREN (bit 2 in ADC_GC register). The ACFE bit must be set for ACFGT to have any effect. 0 Configures "Less Than Threshold, Outside Range Not Inclusive and Inside Range Not Inclusive" functionality based on the values placed in the ADC_CV register. 1 Configures "Greater Than Or Equal To Threshold, Outside Range Inclusive and Inside Range Inclusive" functionality based on the values placed in the ADC_CV registers.
2 ACREN	Compare Function Range Enable Configures the compare function to check the conversion result of the input being monitored is either between or outside the range formed by the compare values in register (ADC_CV) determined by the value of ACFGT. The ACFE bit must be set for ACFGT to have any effect. 0 Range function disabled. Only the compare value 1 of ADC_CV register (CV1) is compared. 1 Range function enabled. Both compare values of ADC_CV registers (CV1 and CV2) are compared.
1 DMAEN	DMA Enable Enables the DMA logic.

Table continues on the next page...

ADCx_GC field descriptions (continued)

Field	Description
	0 DMA disabled (default) 1 DMA enabled
0 ADACKEN	Asynchronous clock output enable Enables the ADC's asynchronous clock source and the clock source output regardless of the conversion and input clock select (ADC_CFG[ADICLK]) settings of the ADC. Based on MCU configuration, the asynchronous clock may be used by other modules (see module introduction section). Setting this bit allows the clock to be used even while the ADC is idle or operating from a different clock source. Also, latency of initiating a single or first-continuous conversion with the asynchronous clock selected is reduced since the ADACK clock is already operational. 0 Asynchronous clock output disabled; Asynchronous clock only enabled if selected by ADICLK and a conversion is active. 1 Asynchronous clock and clock output enabled regardless of the state of the ADC

13.6.8 General status register (ADCx_GS)

Controls the calibration, continuous convert, hardware averaging functions, conversion active, hardware/software trigger select, compare function and voltage reference select of the ADC module.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													AWKST	CALF	ADACT
W														w1c	w1c	
Reset														0	0	0

ADCx_GS field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 AWKST	Asynchronous wakeup interrupt status Holds the status of asynchronous interrupt status that occured during stop mode. This bit is set when ipg_stop is deasserted and ipg_clk has started. It is cleared by writing '1' to it. Clearing this bit also deasserts the Asynchronous interrupt to CPU. 1 Asynnchronous wake up interrupt occured in stop mode. 0 No asynchronous interrupt.
1 CALF	Calibration Failed Flag Displays the result of the calibration sequence. The calibration sequence will fail if Hardware Trigger is selected (i.e. ADC_CFG[ADTRG] = 1), or any ADC register is written, or any stop mode is entered before the calibration sequence completes. The CALF bit is cleared by writing a 1 to it. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
0 ADACT	Conversion Active Indicates that a conversion or hardware averaging is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress. 1 Conversion in progress.

13.6.9 Compare value register (ADCx_CV)

Contains compare values used to compare with the conversion result when the compare function is enabled (ADC_GC[ACFE]=1). The compare values are right justified as shown Figure . Therfore, the compare function only uses the compare value register bits that are related to the ADC mode of operation. (e.g. in 8 bit mode, CV1 = ADC_CV[7:0] and CV2 = ADC_CV[23:16], similarly in 10 bit mode, CV1 = ADC_CV[9:0] and CV2 = ADC_CV[25:16] etc.) The compare value 2 in this register is utilized only when the compare range function is enabled (ADC_GC[ACREN]=1).

Address: Base address + 20h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				CV2												0				CV1											
W																																
Reset	0				0												0				0											

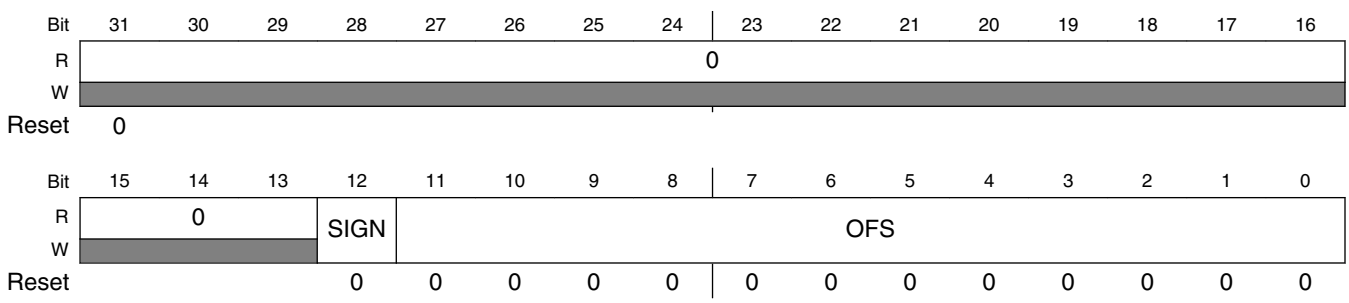
ADCx_CV field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–16 CV2	Compare Value 2 Contains a compare value used to compare with the conversion result when the compare function and compare range function are enabled (ADC_GC[ACFE]=1, ADC_GC[ACREN]=1).
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CV1	Compare Value 1 Contains a compare value used to compare with the conversion result when the compare function ais enabled (ADC_GC[ACFE]=1).

13.6.10 Offset correction value register (ADCx_OFS)

Contains the user-defined offset error correction value. This register is 13 bits wide. The value in the most significant bit (13th bit) is the operation bit. If this bit is ‘0’ then the value in the other 12 bits is added with the converted result value to generate final result to be loaded into ADC_Rn; if this bit is ‘1’ then this field is subtracted from converted value to generate final result (ADC_Rn).

Address: Base address + 24h offset



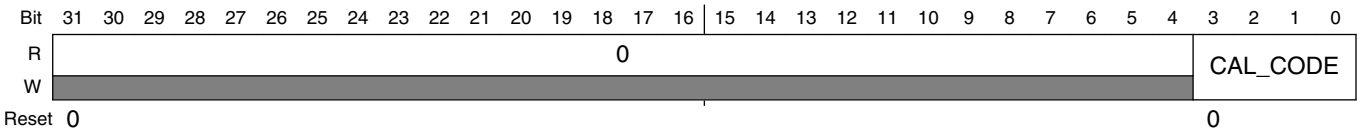
ADCx_OFS field descriptions

Field	Description
31–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SIGN	Sign bit 0 The offset value is added with the raw result 1 The offset value is subtracted from the raw converted value
OFS	Offset value User configurable offset value.

13.6.11 Calibration value register (ADCx_CAL)

Contains calibration information that is generated by the calibration function. This register contains a calibration value of four bits(CAL[3:0]); this is automatically set once the self calibration sequence is done (ADC_SC[CAL] bit is cleared). If this register is written to by the user after calibration, the linearity error specifications may not be met.

Address: Base address + 28h offset



ADCx_CAL field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CAL_CODE	Calibration Result Value This value is automatically loaded and updated at the end of calibration.

Chapter 14
AHB to IP Bridge (AIPSTZ)

14.1 Overview

This section provides an overview of the AHB to IP Bridge (AIPSTZ). This particular peripheral is designed as the bridge between AHB bus and peripherals with the lower bandwidth IP Slave (IPS) buses.

14.1.1 Features

The following list summarizes the key features of the bridge:

- The bridge supports the IPS slave bus signals. This interface is only meant for slave peripherals.
- The bridge supports 8-, 16-, and 32-bit IPS peripherals. (Accesses larger than the size of a peripheral are not supported, except to 32-bit memory.)
- The bridge supports a pair of IPS accesses for 64-bit and certain misaligned AHB transfers to 32-bit memory in 64-bit platforms.
- The bridge directly supports up to 32 16-Kbyte external IPS peripherals, and 2 global external IPS peripheral spaces. The bridge occupies 1 MBytes of total address space.
- The bridge provides configurable per-block and per-master access protections. Access permissions are based on bus master (e.g. DMA or core) privilege levels and resource domain. More details on the protection features and configuration can be found in the Security Reference Manual
- Peripheral read transactions require a minimum of 2 hclk clocks, and unbuffered write transactions require a minimum of 3 hclk clocks.
- The bridge uses one single asynchronous reset and one global clock.

14.2 Clocks

The following table describes the clock sources for AIPSTZ. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 14-1. AIPSTZ Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	Module clock

14.3 Functional Description

The AIPS bridge serves as a protocol translator between the AHB system bus and the IP bus.

Support is provided for generating a pair of 32-bit IP bus accesses when targeted by a 64-bit system bus access, or a misaligned access which crosses a 32-bit boundary. No other bus-sizing access support is provided.

The AHB to IP bridge is the interface between the AHB and on-chip IPS peripherals, which are sub-blocks containing readable/writable control and status registers.

The AHB master reads and writes these registers through the AIPSTZ. The bridge generates block enables, the block address, transfer attributes, byte enables and write data as inputs to the IPS peripherals. The bridge captures read data from the IPS interface and drives it on the AHB.

Each bridge that connects to the IPS (or peripherals) are referred as AIPS. The chip has three separate AIPS modules, and peripherals are grouped and assigned under each AIPS block. The list of peripherals are indicated as n-1, n-2, and n-3 for AIPS-1, AIPS-2, and AIPS-3 respectively.

AIPS occupies a 1-Mbyte portion of the address space. The register maps of the IPS peripherals are located on 16-Kbyte boundaries. Each IPS peripheral is allocated one 16-Kbyte block of the memory map, and is activated by one of the block enables from the bridge. Up to thirty-two 16-Kbyte external IPS peripherals may be implemented, occupying contiguous blocks of 16-Kbytes. Two global external IPS block enables are available for the remaining address space to allow for customization and expansion of addressed peripheral devices. In addition, a single "non-global" block enable is also asserted whenever any of the thirty-two non-global block enables is asserted.

The bridge is responsible for indicating to IPS peripherals if an access is in supervisor or user mode. It may block user mode accesses to certain IPS peripherals or it may allow the individual IPS peripherals to determine if user mode accesses are allowed. In addition, peripherals may be designated as write-protected.

The bridge supports the notion of "trusted" masters for security purposes. Masters may be individually designated as trusted for reads, trusted for writes, or trusted for both reads and writes, as well as being forced to look as though all accesses from a master are in user-mode privilege level. Refer to AIPSTZ Memory Map/Register Definition for more information.

The AIPSTZ prevents access to a peripheral if the transaction originated from a source from a resource domain that has been explicitly omitted. Resource domains are assigned in the RDC submodule. Please refer to the RDC chapter for programming details.

All peripheral devices are expected to only require aligned accesses equal to or smaller in size than the peripheral size. An exception to this rule is supported for 32-bit peripherals to allow memory to be placed on the IPS.

14.4 Access Protections

The AIPSTZ bridge provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user-mode privilege, and allows masters to be designated as trusted or untrusted.

Peripherals may require supervisor privilege level for access, may restrict access to a trusted master only, and may be write-protected. IP bus peripherals are subject to access control policies set in both CSU registers and AIPSTZ registers. An access is blocked if it is denied by either policy.

Masters and peripherals are assigned to one or more resource domains in the RDC submodule (see the RDC chapter for details). Depending on RDC programming, masters transactions through the AIPSTZ may or may not be allowed access to peripherals in different resource domains.

14.5 Access Support

Aligned 64-bit accesses, aligned and misaligned word and half word accesses, as well as byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the IPS.

Peripheral registers must not be misaligned, although no explicit checking is performed by the AIPS bridge. The bridge will perform two IPS transfers for 64-bit accesses, word accesses with byte offsets of 1, 2, or 3, and for half word accesses with a byte offset of 3. All other accesses will be performed with a single IPS transfer.

Only aligned half word and byte accesses are supported for 16-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

Only byte accesses are supported for 8-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

14.6 Initialization Information

The AIPS bridge should be programmed before use.

The following registers should be initialized: The Master Privilege Registers (AIPSTZ_MPRs), the Peripheral Access Control registers (AIPSTZ_PACRs), and the Off-platform Peripheral Access Control registers (AIPSTZ_OPACRs) described in AIPSTZ Memory Map/Register Definition.

14.6.1 Security Block

The AIPSTZ contains a security block that is connected to each off-platform peripheral. This block filters accesses based on write/read, non-secure, and supervisor signals.

Each peripheral can be individually configured to allow or deny each of the following transactions as described in the table below:

Table 14-2. Peripheral Access Configuration options

Config Bit	Write	Non-Secure	Supervisor	Meaning
0	0	0	0	Secure User Read
1	0	0	1	Secure Supervisor Read
2	0	1	0	Non-Secure User Read
3	0	1	1	Non-Secure Supervisor Read
4	1	0	0	Secure User Write
5	1	0	1	Secure Supervisor Write
6	1	1	0	Non-Secure User Write
7	1	1	1	Non-Secure Supervisor Write

Each peripheral has a security configuration (sec_config_X) input for determining whether to allow or deny a given access type. These are 8-bit vectors, with each bit corresponding to one of the transactions above as listed in the Config Bit column of Table 14-2. If the bit is asserted (1'b1), the transaction is allowed. If the bit is negated (1'b0), the transaction is not allowed.

For example, if peripheral 0 is configured as follows:

sec_config_0 [7:0] = 8'b0011_0011

This peripheral can only be accessed by secure transactions. Bits 0, 1, 4, and 5 are asserted and these bits refer to the four types of secure transactions. If an insecure transaction is attempted to this peripheral, it will result in an error.

Eight bits per peripheral across an entire system can result in a large number of configuration bits that must be assigned and controlled, most likely in a series of registers in another block. To reduce the number of register bits required predefined sets of security profiles can be defined and encapsulated in an external security translation block. The table below describes one set of security profiles that has been proposed for use with the AIPSTZ.

Table 14-3. Security Levels

CSU_SEC_LEVEL	Non-Secure User	Non-Secure Supervisor	Secure User	Secure Supervisor
0	RD+WR	RD+WR	RD+WR	RD+WR
1	NOT ALLOWED	RD+WR	RD+WR	RD+WR
2	Read Only	Read Only	RD+WR	RD+WR
3	NOT ALLOWED	Read Only	RD+WR	RD+WR
4	NOT ALLOWED	NOT ALLOWED	RD+WR	RD+WR
5	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED	RD+WR
6	NOT ALLOWED	NOT ALLOWED	Read Only	Read Only
7	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED

Information regarding CSU is provided in the Security Reference Manual. Contact your Freescale representative for information about obtaining this document.

A 3-bit input, 8-bit output translation block can be used such that only three register bits are required to set the security profile and the translation block will drive the correct 8-bit configuration vector. Each peripheral connected to the AIPSTZ would require this translation block. The top level AIPSTZ has this three bit input line `csu_sec_level[2:0]' corresponding to each peripheral X.

14.7 AIPSTZ Memory Map/Register Definition

The memory map for the AIPS SW-visible registers is shown in the table below.

The MPROT and OPACR fields are 4 bits in width. Some bits may be reserved depending on device.

AIPSTZ memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Priviledge Registers (AIPSTZ_MPR)	32	R/W	7700_0000h	14.7.1/428
40	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR)	32	R/W	4444_4444h	14.7.2/431
44	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR1)	32	R/W	4444_4444h	14.7.3/434
48	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR2)	32	R/W	4444_4444h	14.7.4/437
4C	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR3)	32	R/W	4444_4444h	14.7.5/440
50	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR4)	32	R/W	4444_4444h	14.7.6/443

14.7.1 Master Priviledge Registers (AIPSTZ_MPR)

Each AIPSTZ_MPR specifies 16 4-bit fields defining the access privilege level associated with a bus master in the platform, as well as specifying whether write accesses from this master are bufferable shown in Table 14-4

The registers provide one field per bus master, where field 15 corresponds to master 15, field 14 to master 14,... field 0 to master 0 (typically the processor core). The master index allocation is shown in Table 14-5.

Table 14-4. MPROT Field

Bit	Field	Description
3	MBW	**Master Buffer Writes** - This bit determines whether the AIPSTZ is enabled to buffer writes from this master.
2	MTR	**Master Trusted for Reads** - This bit determines whether the master is trusted for read accesses.
1	MTW	**Master Trusted for Writes** - This bit determines whether the master is trusted for write accesses.
0	MPL	**Master Privilege Level** - This bit determines how the privilege level of the master is determined.

NOTE

The reset value is set to 0000_0000_7700_0000, which makes master 0 and master 1 (ARM CORE) the trusted masters. Trusted software can change the settings after reset.

Table 14-5. Master Index Allocation

Master Index	Master Name	Comments
Master 0	All masters excluding ARM core, SDMA and CAAM	Share the same number allocation.
Master 1	ARM A7 CORE	
Master 2	CAAM	
Master 3	SDMA	
Master 4	Reserved	
Master 5	ARM M4 Core	
Master 6-15	Reserved	

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MPROT0				MPROT1				MPROT2				MPROT3				Reserved				MPROT5				Reserved							
Reset	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AIPSTZ_MPR field descriptions

Field	Description
31–28 MPROT0	Master 0 Priviledge, Buffer, Read, Write Control xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
27–24 MPROT1	Master 1 Priviledge, Buffer, Read, Write Control xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses.

Table continues on the next page...

AIPSTZ_MPR field descriptions (continued)

Field	Description
	0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
23–20 MPROT2	Master 2 Priviledge, Buffer, Read, Write Control xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
19–16 MPROT3	Master 3 Priviledge, Buffer, Read, Write Control. xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
15–12 -	This field is reserved. Reserved
11–8 MPROT5	Master 5 Priviledge, Buffer, Read, Write Control. xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
-	This field is reserved. Reserved

14.7.2 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 14-6

Table 14-6. OPAC Field

Bit	Field	Description
3	BW	**Buffer Writes** - This bit determines whether write accesses to this peripheral are allowed to be buffered.[1]
2	SP	**Supervisor Protect** - This bit determines whether the peripheral requires supervisor privilege level for access.
1	WP	**Write Protect** - This bit determines whether the peripheral allows write accesses.
0	TP	**Trusted Protect** - This bit determines whether the peripheral allows accesses from an untrusted master.

1. Buffered writes are not available for AIPSTZ. This bit should be set to '0'.

Address: 0h base + 40h offset = 40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC0				OPAC1				OPAC2				OPAC3				OPAC4				OPAC5				OPAC6				OPAC7			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR field descriptions

Field	Description
31–28 OPAC0	Off-platform Peripheral Access Control 0 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR field descriptions (continued)

Field	Description
27–24 OPAC1	Off-platform Peripheral Access Control 1 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC2	Off-platform Peripheral Access Control 2 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC3	Off-platform Peripheral Access Control 3 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC4	Off-platform Peripheral Access Control 4

Table continues on the next page...

AIPSTZ_OPACR field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC5	Off-platform Peripheral Access Control 5 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC6	Off-platform Peripheral Access Control 6 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC7	Off-platform Peripheral Access Control 7 xxx0 **TP** — Accesses from an untrusted master are allowed.

Table continues on the next page...

AIPSTZ_OPACR field descriptions (continued)

Field	Description
	xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

14.7.3 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR1)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 14-6

Address: 0h base + 44h offset = 44h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC8				OPAC9				OPAC10				OPAC11				OPAC12				OPAC13				OPAC14				OPAC15			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR1 field descriptions

Field	Description
31–28 OPAC8	Off-platform Peripheral Access Control 8 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

Table continues on the next page...

AIPSTZ_OPACR1 field descriptions (continued)

Field	Description
	0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC9	Off-platform Peripheral Access Control 9 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC10	Off-platform Peripheral Access Control 10 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC11	Off-platform Peripheral Access Control 11 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR1 field descriptions (continued)

Field	Description
15–12 OPAC12	Off-platform Peripheral Access Control 12 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC13	Off-platform Peripheral Access Control 13 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC14	Off-platform Peripheral Access Control 14 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC15	Off-platform Peripheral Access Control 15

Table continues on the next page...

AIPSTZ_OPACR1 field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

14.7.4 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR2)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 14-6

Address: 0h base + 48h offset = 48h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC16				OPAC17				OPAC18				OPAC19				OPAC20				OPAC21				OPAC22				OPAC23			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR2 field descriptions

Field	Description
31–28 OPAC16	Off-platform Peripheral Access Control 16 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

Table continues on the next page...

AIPSTZ_OPACR2 field descriptions (continued)

Field	Description
	0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC17	Off-platform Peripheral Access Control 17 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC18	Off-platform Peripheral Access Control 18 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC19	Off-platform Peripheral Access Control 19 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR2 field descriptions (continued)

Field	Description
15–12 OPAC20	Off-platform Peripheral Access Control 20 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC21	Off-platform Peripheral Access Control 21 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC22	Off-platform Peripheral Access Control 22 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC23	Off-platform Peripheral Access Control 23

Table continues on the next page...

AIPSTZ_OPACR2 field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

14.7.5 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR3)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 14-6

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC24				OPAC25				OPAC26				OPAC27				OPAC28				OPAC29				OPAC30				OPAC31			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR3 field descriptions

Field	Description
31–28 OPAC24	Off-platform Peripheral Access Control 24 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

Table continues on the next page...

AIPSTZ_OPACR3 field descriptions (continued)

Field	Description
	0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC25	Off-platform Peripheral Access Control 25 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC26	Off-platform Peripheral Access Control 26 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC27	Off-platform Peripheral Access Control 27 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR3 field descriptions (continued)

Field	Description
15–12 OPAC28	Off-platform Peripheral Access Control 28 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC29	Off-platform Peripheral Access Control 29 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC30	Off-platform Peripheral Access Control 30 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC31	Off-platform Peripheral Access Control 31

Table continues on the next page...

AIPSTZ_OPACR3 field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

14.7.6 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR4)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 14-6

Address: 0h base + 50h offset = 50h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC32				OPAC33				Reserved																							
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR4 field descriptions

Field	Description
31–28 OPAC32	Off-platform Peripheral Access Control 32 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

Table continues on the next page...

AIPSTZ_OPACR4 field descriptions (continued)

Field	Description
	0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC33	Off-platform Peripheral Access Control 33 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
-	This field is reserved. Reserved

Chapter 15
AHB-to-APBH Bridge with DMA (APBH-Bridge-DMA)

15.1 Overview

The AHB-to-APBH bridge provides the chip with an inexpensive peripheral attachment bus running on the AHB's HCLK.

(The H in APBH denotes that the APB*H* is synchronous to HCLK.)

As shown in the figure below, the AHB-to-APBH bridge includes the AHB-to-APB PIO bridge for a memory-mapped I/O to the APB devices, as well as a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM core. Each one of the APB peripherals, including the vectored interrupt controller, is documented in their respective chapters.

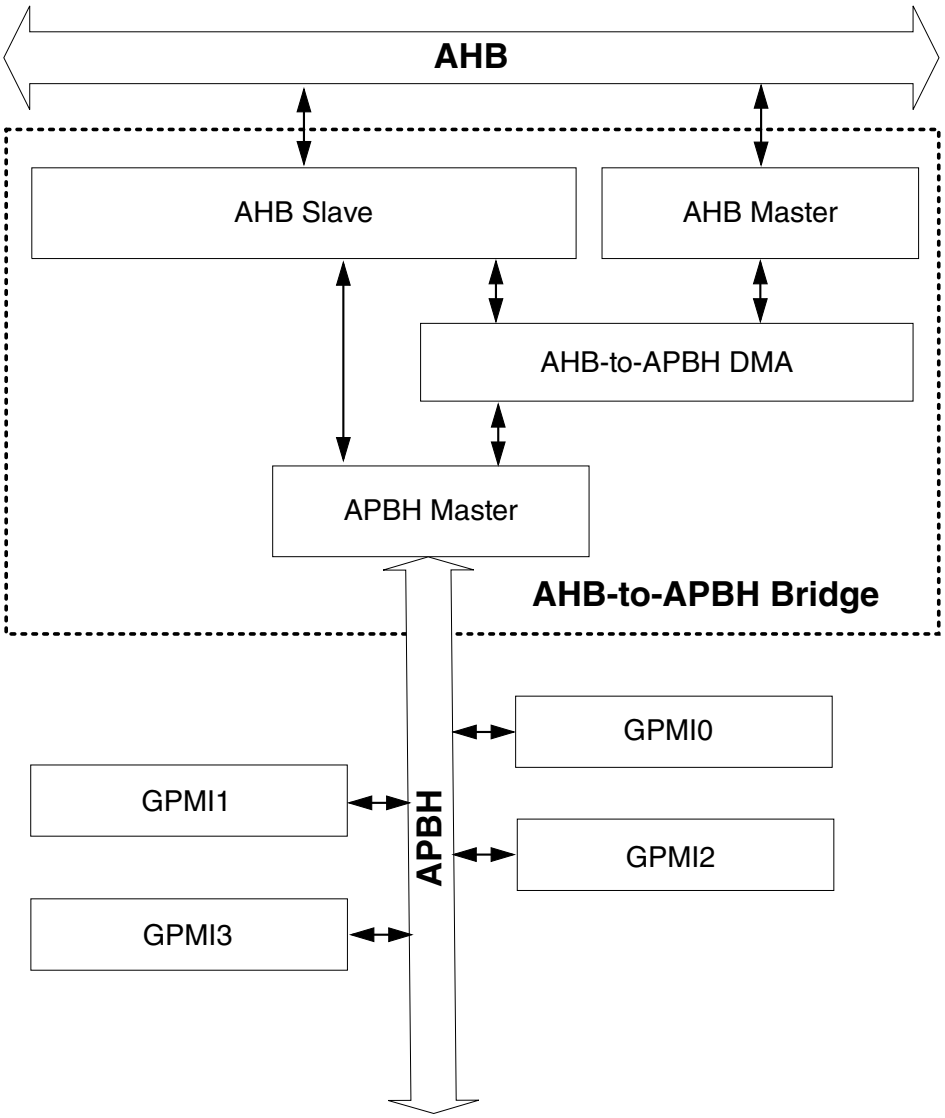


Figure 15-1. AHB-to-APBH Bridge DMA Block Diagram

There is no separate DMA bus for these devices. Contention between the DMA's use of the APBH bus and the AHB-to-APB bridge functions' use of the APBH is mediated by an internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report "not ready" through its HREADY output until the bridge transfer can complete. The arbiter tracks repeated lockouts and inverts the priority, guaranteeing the ARM platform every fourth transfer on the APB.

15.2 Clocks

The table found here describes the clock sources for APBH.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 15-1. APBH Clocks

Clock name	Clock Root	Description
hclk	bch_clk_root	Module clock

15.3 APBH DMA

The DMA supports four channels of DMA services, as shown in the following table. The shared DMA resource allows each independent channel to follow a simple chained command list.

Command chains are built up using the general structure, as shown in Figure 15-2.

Table 15-2. APBH DMA channel assignments

APBH DMA Channel #	Usage
0	GPMI0
1	GPMI1
2	GPMI2
3	GPMI3

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the ARM platform can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA, and have no further concern for the device until the DMA completion interrupt occurs. The goal is to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1 KHz (arrival intervals longer than 1 ms).

A single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and controls that it uses to write DMA data bytes to the device. For example, this allows a chain of operations to be issued to the GPMI controller to send NAND command bytes, address bytes, and data transfers where the command and the address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA. Each DMA structure can have 0–15 PIO words appended to it. The CMDPIOWORDS field, if non-zero, instructs the DMA engine to copy these words to the APB, beginning at the first register address offset for the peripheral and and incrementing the register offset each cycle.

The DMA master generates only normal read/write transfers to the APBH. It does *not* generate set, clear, or toggle (SCT) transfers.

After any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. Table 15-3 shows the four commands implemented by the DMA.

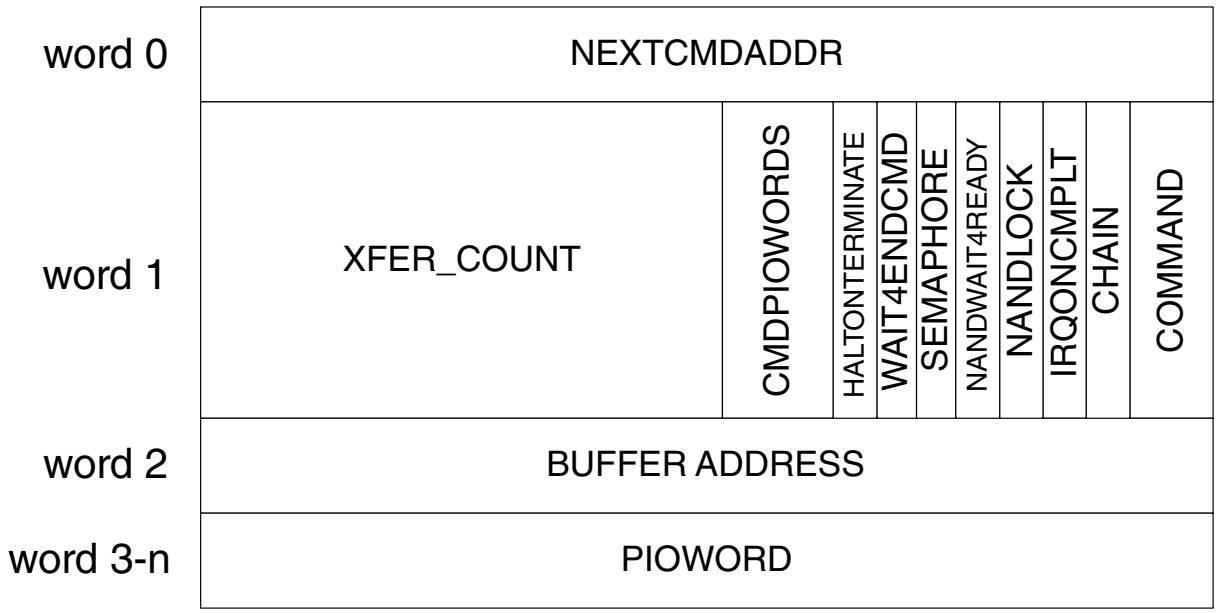


Figure 15-2. AHB-to-APBH Bridge DMA channel command structure

Table 15-3. APBH DMA commands

DMA Command	Usage
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.
11	DMA_SENSE. Perform any requested PIO word transfers, then perform a conditional branch to the next chained device. Follow the NEXTCMD_ADDR pointer if the peripheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true. This command becomes a no-operation for any channel other than a GPMI channel.

DMA_WRITE operations copy data bytes to the system memory (on-chip RAM or SDRAM) from the associated peripheral.

DMA_READ operations copy data bytes to the APB peripheral from the system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or

departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers. This command is useful in such applications as activating the NAND devices CHECKSTATUS operation. The check status command reads a status byte from the NAND device, performs an XOR and MASK against an expected value supplied as part of the PIO transfer. Once the read check completes (see NAND Read Status Polling Example), the NO_DMA_XFER command completes. The result in the peripheral is that its sense line is driven by the results of the comparison. The sense flip-flop is only updated by CHECKSTATUS for the device that is executed. At some future point, the chain contains a DMA command structure with the fourth and final command value, that is, the DMA_SENSE command.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. The DMA_SENSE command uses the DMA buffer pointer word of the command structure to point to an alternate DMA command structure chain or list. The DMA_SENSE command examines the sense line of the associated peripheral. If the sense line is false, then the DMA follows the standard list found whose next command is found from the pointer in the NEXTCMD_ADDR word of the command structure. If the sense line is true, then the DMA follows the alternate list whose next command is found from the pointer in the DMA Buffer Pointer word of the DMA_SENSE command structure (see Figure 15-2). The sense command ignores the CHAIN bit, so that both pointers must be valid when the DMA comes to a sense command.

If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, the DMA channel waits for the device to signal completion of a command by toggling the endcmd signal before proceeding to load and execute the next command structure. Then, if DECREMENT_SEMAPHORE is set, the semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in the following table, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer-count mechanism is duplicated in the associated peripheral, either as an implied or as a specified count in the peripheral.

Table 15-4. DMA channel command word in system memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															

Table continues on the next page...

Table 15-4. DMA channel command word in system memory (continued)

Number DMA Bytes to Transfer	Number PIO Words to Write				HALTONTERMINATE	WAIT4ENDCMD	DECREMENT SEMAPHORE	NANDwAIT4READY	NANDLOCK	IRQ_COMPLETE	CHAIN	COMMAND
DMA Buffer or Alternate CCW												
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBH Bus												

Figure 15-2 also shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1, if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it is not detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt-status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by the software. It can be used to interrupt the ARM platform.

The NAND_LOCK bit is monitored by the DMA channel arbiter. succeeds in the arbiter with its NAND_LOCK bit set, then the arbiter ignores the other NAND channels until a command is completed in which the NAND_LOCK is not set. Notice that the semantic here is that the NAND_LOCK state is to limit scheduling of a non-locked DMA. A DMA channel can go from unlocked to locked in the arbiter at the beginning of a command when the NAND_LOCK bit is set. When the last DMA command of an atomic sequence is completed, the lock should be removed. To accomplish this, the last command does not have the NAND_LOCK bit. It is still locked in the atomic state within the arbiter when the command starts, so that it is the only NAND command that can be executed. At the end, it drops from the atomic state within the arbiter.

The NAND_WAIT4READY bit also has a special use for GPMI channels (from channel to channel), i.e., the NAND device channels. The GPMI peripheral supplies a sample of the ready line from the NAND device. This ready value is used to hold off of a command with this bit set until the ready line is asserted to 1. Once the arbiter sees a command with a wait-for-ready set, it holds off that channel until ready is asserted.

Receiving an IRQ for HALTONTERMINATE (HOT) is a feature in the APBH DMA descriptor that allows GPMI to signal to the DMA engine that an error has occurred. If a commmand is stalled due to an error, a HOT signal is sent from the peripheral to the DMA engine and causes an IRQ after terminating the DMA descriptor being executed.

Therefore, it is recommended that software use this signal as follows:

- Always set HALTONTERMINATE to 1 in a DMA descriptor. That way, if a peripheral signals HOT, the transfer will end, leaving the peripheral block and the DMA engine synchronized (but at the end of a command).
- When an IRQ from an APBH channel is received, and the IRQ is determined to be due to an error (as opposed to an IRQONCOMPLETE interrupt) the software should:
  - Reset the channel.
  - Determine the error from error reporting in the peripheral block, then manage the error in the peripheral that is attached to that channel in whatever appropriate way exists for that device (software recovery, device reset, block reset, etc).

Each channel has an eight-bit counting semaphore that controls whether it is in the idle state. When the semaphore is non-zero, the channel is ready to run, process commands and perform DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the idle state and remains there until the semaphore is incremented by the software. When the semaphore goes to non-zero and the channel is in its idle state, then it uses the value in the APBH_CHn_NXTCMDAR register (next command address register) to fetch a pointer to the next command to process.

NOTE

This is a double indirect case. This method allows the software to append to a running command list under the protection of the counting semaphore.

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the APBH_CHn_NXTCMDAR register, and then writes 1 to the counting semaphore in APBH_CHn_SEMA. The DMA channel loads APBH_CHn_CURCMDAR register and then enters the normal state machine processing for the next command. When the software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of APBH_CHn_CURCMDAR at any time to determine the location of the command structure currently being processed.

15.4 NAND Read Status Polling Example

The following figure shows a more complicated scenario.

This subset of a NAND device workload shows that the first two command structures are used during the data-write phase of an NAND device write operation (CLE and ALE transfers omitted for clarity).

- After writing the data, one must wait until the NAND device status register indicates that the write charge has been transferred. This is built into the workload using a check status command in the NAND in a loop created from the next two DMA command structures.
- The NO_DMA_TRANSFER command is shown here performing the read check, followed by a DMA_SENSE command to branch the DMA command structure list, based on the status of a bit in the external NAND device.

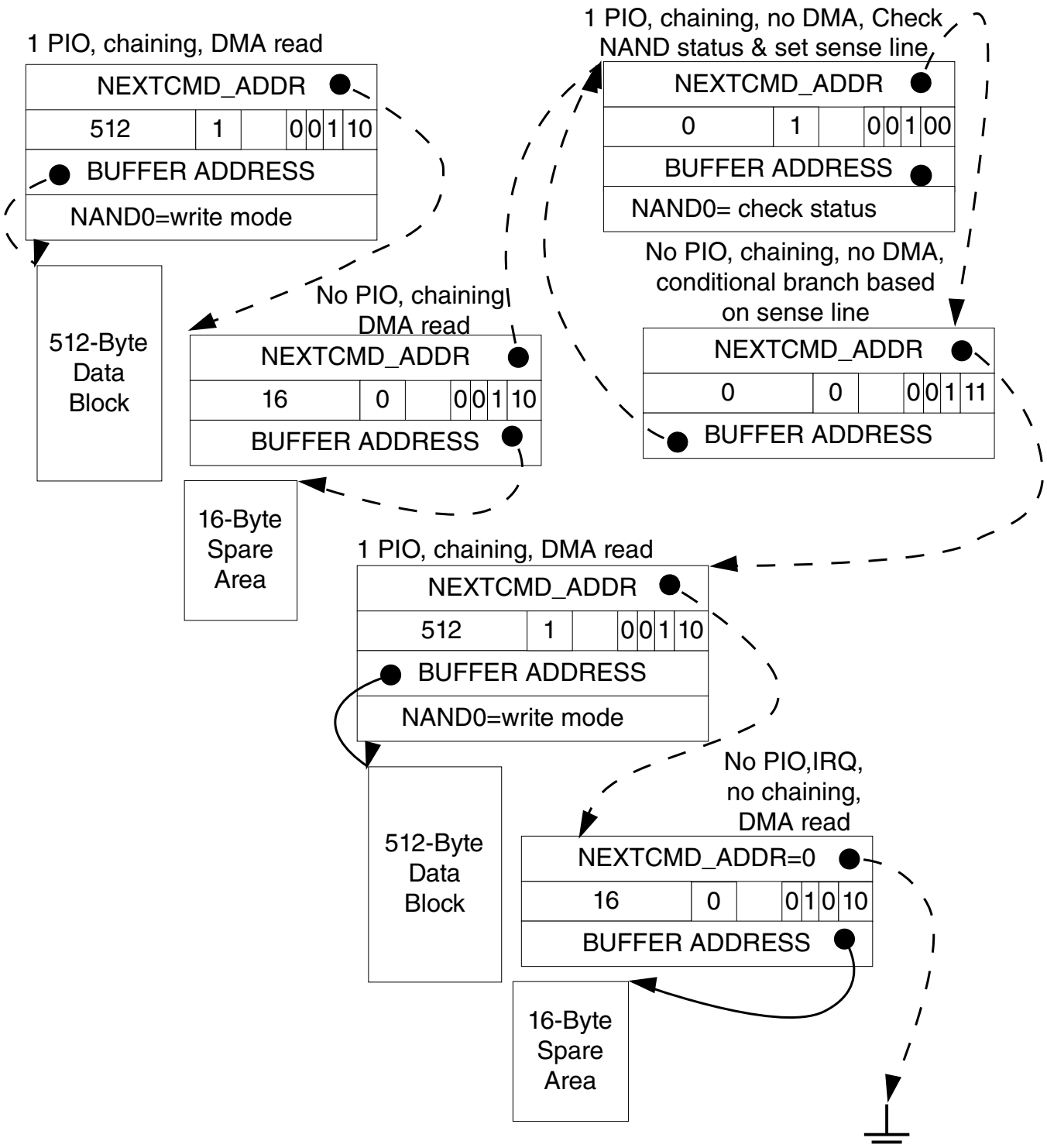


Figure 15-3. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command

The example in the above figure shows the workload continuing immediately to the next NAND page transfer. However, one could perform a second sense operation to see if an error has occurred after the write. One could then point the sense command alternate branch at a NO_DMA_XFER command with the interrupt bit set. If the CHAIN bit is not

set on this failure branch, then the ARM platform is interrupted immediately, and the channel process is also immediately terminated in the presence of a workload-detected NAND error bit.

Note that each word of the three-word DMA command structure corresponds to a PIO register of the DMA that is accessible on the APBH bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register.

To start DMA processing for the first command, initialize the PIO registers of the desired channel, as follows:

- First, load the next command address register with a pointer to the first command to be loaded.
- Then, write 1 to the counting semaphore register. This causes the DMA to schedule the targeted channel for the DMA command structure load, just as if it had finished its previous command.

15.5 APBH Memory Map/Register Definition

APBH Hardware Register Format Summary

APBH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_4000	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0)	32	R/W	E000_0000h	15.5.1/460
180_4004	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_SET)	32	R/W	E000_0000h	15.5.1/460
180_4008	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_CLR)	32	R/W	E000_0000h	15.5.1/460
180_400C	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_TOG)	32	R/W	E000_0000h	15.5.1/460
180_4010	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1)	32	R/W	0000_0000h	15.5.2/461
180_4014	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_SET)	32	R/W	0000_0000h	15.5.2/461
180_4018	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_CLR)	32	R/W	0000_0000h	15.5.2/461

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_401C	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_TOG)	32	R/W	0000_0000h	15.5.2/461
180_4020	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2)	32	R/W	0000_0000h	15.5.3/465
180_4024	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_SET)	32	R/W	0000_0000h	15.5.3/465
180_4028	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_CLR)	32	R/W	0000_0000h	15.5.3/465
180_402C	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_TOG)	32	R/W	0000_0000h	15.5.3/465
180_4030	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL)	32	R/W	0000_0000h	15.5.4/470
180_4034	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_SET)	32	R/W	0000_0000h	15.5.4/470
180_4038	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_CLR)	32	R/W	0000_0000h	15.5.4/470
180_403C	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_TOG)	32	R/W	0000_0000h	15.5.4/470
180_4040	AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)	32	R/W	0000_0000h	15.5.5/471
180_4050	AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)	32	R/W	0055_5555h	15.5.6/472
180_4060	AHB to APBH DMA Debug Register (APBH_DEBUG)	32	R/W	0000_0000h	15.5.7/473
180_4100	APBH DMA Channel n Current Command Address Register (APBH_CH0_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4110	APBH DMA Channel n Next Command Address Register (APBH_CH0_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4120	APBH DMA Channel n Command Register (APBH_CH0_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4130	APBH DMA Channel n Buffer Address Register (APBH_CH0_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4140	APBH DMA Channel n Semaphore Register (APBH_CH0_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4150	AHB to APBH DMA Channel n Debug Information (APBH_CH0_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4160	AHB to APBH DMA Channel n Debug Information (APBH_CH0_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4170	APBH DMA Channel n Current Command Address Register (APBH_CH1_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4180	APBH DMA Channel n Next Command Address Register (APBH_CH1_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4190	APBH DMA Channel n Command Register (APBH_CH1_CMD)	32	R/W	0000_0000h	15.5.10/475
180_41A0	APBH DMA Channel n Buffer Address Register (APBH_CH1_BAR)	32	R/W	0000_0000h	15.5.11/477

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_41B0	APBH DMA Channel n Semaphore Register (APBH_CH1_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_41C0	AHB to APBH DMA Channel n Debug Information (APBH_CH1_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_41D0	AHB to APBH DMA Channel n Debug Information (APBH_CH1_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_41E0	APBH DMA Channel n Current Command Address Register (APBH_CH2_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_41F0	APBH DMA Channel n Next Command Address Register (APBH_CH2_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4200	APBH DMA Channel n Command Register (APBH_CH2_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4210	APBH DMA Channel n Buffer Address Register (APBH_CH2_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4220	APBH DMA Channel n Semaphore Register (APBH_CH2_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4230	AHB to APBH DMA Channel n Debug Information (APBH_CH2_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4240	AHB to APBH DMA Channel n Debug Information (APBH_CH2_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4250	APBH DMA Channel n Current Command Address Register (APBH_CH3_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4260	APBH DMA Channel n Next Command Address Register (APBH_CH3_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4270	APBH DMA Channel n Command Register (APBH_CH3_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4280	APBH DMA Channel n Buffer Address Register (APBH_CH3_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4290	APBH DMA Channel n Semaphore Register (APBH_CH3_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_42A0	AHB to APBH DMA Channel n Debug Information (APBH_CH3_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_42B0	AHB to APBH DMA Channel n Debug Information (APBH_CH3_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_42C0	APBH DMA Channel n Current Command Address Register (APBH_CH4_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_42D0	APBH DMA Channel n Next Command Address Register (APBH_CH4_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_42E0	APBH DMA Channel n Command Register (APBH_CH4_CMD)	32	R/W	0000_0000h	15.5.10/475
180_42F0	APBH DMA Channel n Buffer Address Register (APBH_CH4_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4300	APBH DMA Channel n Semaphore Register (APBH_CH4_SEMA)	32	R/W	0000_0000h	15.5.12/478

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_4310	AHB to APBH DMA Channel n Debug Information (APBH_CH4_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4320	AHB to APBH DMA Channel n Debug Information (APBH_CH4_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4330	APBH DMA Channel n Current Command Address Register (APBH_CH5_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4340	APBH DMA Channel n Next Command Address Register (APBH_CH5_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4350	APBH DMA Channel n Command Register (APBH_CH5_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4360	APBH DMA Channel n Buffer Address Register (APBH_CH5_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4370	APBH DMA Channel n Semaphore Register (APBH_CH5_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4380	AHB to APBH DMA Channel n Debug Information (APBH_CH5_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4390	AHB to APBH DMA Channel n Debug Information (APBH_CH5_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_43A0	APBH DMA Channel n Current Command Address Register (APBH_CH6_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_43B0	APBH DMA Channel n Next Command Address Register (APBH_CH6_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_43C0	APBH DMA Channel n Command Register (APBH_CH6_CMD)	32	R/W	0000_0000h	15.5.10/475
180_43D0	APBH DMA Channel n Buffer Address Register (APBH_CH6_BAR)	32	R/W	0000_0000h	15.5.11/477
180_43E0	APBH DMA Channel n Semaphore Register (APBH_CH6_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_43F0	AHB to APBH DMA Channel n Debug Information (APBH_CH6_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4400	AHB to APBH DMA Channel n Debug Information (APBH_CH6_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4410	APBH DMA Channel n Current Command Address Register (APBH_CH7_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4420	APBH DMA Channel n Next Command Address Register (APBH_CH7_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4430	APBH DMA Channel n Command Register (APBH_CH7_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4440	APBH DMA Channel n Buffer Address Register (APBH_CH7_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4450	APBH DMA Channel n Semaphore Register (APBH_CH7_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4460	AHB to APBH DMA Channel n Debug Information (APBH_CH7_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_4470	AHB to APBH DMA Channel n Debug Information (APBH_CH7_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4480	APBH DMA Channel n Current Command Address Register (APBH_CH8_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4490	APBH DMA Channel n Next Command Address Register (APBH_CH8_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_44A0	APBH DMA Channel n Command Register (APBH_CH8_CMD)	32	R/W	0000_0000h	15.5.10/475
180_44B0	APBH DMA Channel n Buffer Address Register (APBH_CH8_BAR)	32	R/W	0000_0000h	15.5.11/477
180_44C0	APBH DMA Channel n Semaphore Register (APBH_CH8_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_44D0	AHB to APBH DMA Channel n Debug Information (APBH_CH8_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_44E0	AHB to APBH DMA Channel n Debug Information (APBH_CH8_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_44F0	APBH DMA Channel n Current Command Address Register (APBH_CH9_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4500	APBH DMA Channel n Next Command Address Register (APBH_CH9_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4510	APBH DMA Channel n Command Register (APBH_CH9_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4520	APBH DMA Channel n Buffer Address Register (APBH_CH9_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4530	APBH DMA Channel n Semaphore Register (APBH_CH9_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4540	AHB to APBH DMA Channel n Debug Information (APBH_CH9_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4550	AHB to APBH DMA Channel n Debug Information (APBH_CH9_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4560	APBH DMA Channel n Current Command Address Register (APBH_CH10_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4570	APBH DMA Channel n Next Command Address Register (APBH_CH10_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4580	APBH DMA Channel n Command Register (APBH_CH10_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4590	APBH DMA Channel n Buffer Address Register (APBH_CH10_BAR)	32	R/W	0000_0000h	15.5.11/477
180_45A0	APBH DMA Channel n Semaphore Register (APBH_CH10_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_45B0	AHB to APBH DMA Channel n Debug Information (APBH_CH10_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_45C0	AHB to APBH DMA Channel n Debug Information (APBH_CH10_DEBUG2)	32	R/W	0000_0000h	15.5.14/482

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_45D0	APBH DMA Channel n Current Command Address Register (APBH_CH11_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_45E0	APBH DMA Channel n Next Command Address Register (APBH_CH11_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_45F0	APBH DMA Channel n Command Register (APBH_CH11_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4600	APBH DMA Channel n Buffer Address Register (APBH_CH11_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4610	APBH DMA Channel n Semaphore Register (APBH_CH11_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4620	AHB to APBH DMA Channel n Debug Information (APBH_CH11_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4630	AHB to APBH DMA Channel n Debug Information (APBH_CH11_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4640	APBH DMA Channel n Current Command Address Register (APBH_CH12_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_4650	APBH DMA Channel n Next Command Address Register (APBH_CH12_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4660	APBH DMA Channel n Command Register (APBH_CH12_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4670	APBH DMA Channel n Buffer Address Register (APBH_CH12_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4680	APBH DMA Channel n Semaphore Register (APBH_CH12_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4690	AHB to APBH DMA Channel n Debug Information (APBH_CH12_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_46A0	AHB to APBH DMA Channel n Debug Information (APBH_CH12_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_46B0	APBH DMA Channel n Current Command Address Register (APBH_CH13_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_46C0	APBH DMA Channel n Next Command Address Register (APBH_CH13_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_46D0	APBH DMA Channel n Command Register (APBH_CH13_CMD)	32	R/W	0000_0000h	15.5.10/475
180_46E0	APBH DMA Channel n Buffer Address Register (APBH_CH13_BAR)	32	R/W	0000_0000h	15.5.11/477
180_46F0	APBH DMA Channel n Semaphore Register (APBH_CH13_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4700	AHB to APBH DMA Channel n Debug Information (APBH_CH13_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4710	AHB to APBH DMA Channel n Debug Information (APBH_CH13_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4720	APBH DMA Channel n Current Command Address Register (APBH_CH14_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_4730	APBH DMA Channel n Next Command Address Register (APBH_CH14_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_4740	APBH DMA Channel n Command Register (APBH_CH14_CMD)	32	R/W	0000_0000h	15.5.10/475
180_4750	APBH DMA Channel n Buffer Address Register (APBH_CH14_BAR)	32	R/W	0000_0000h	15.5.11/477
180_4760	APBH DMA Channel n Semaphore Register (APBH_CH14_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_4770	AHB to APBH DMA Channel n Debug Information (APBH_CH14_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_4780	AHB to APBH DMA Channel n Debug Information (APBH_CH14_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4790	APBH DMA Channel n Current Command Address Register (APBH_CH15_CURCMDAR)	32	R/W	0000_0000h	15.5.8/474
180_47A0	APBH DMA Channel n Next Command Address Register (APBH_CH15_NXTCMDAR)	32	R/W	0000_0000h	15.5.9/475
180_47B0	APBH DMA Channel n Command Register (APBH_CH15_CMD)	32	R/W	0000_0000h	15.5.10/475
180_47C0	APBH DMA Channel n Buffer Address Register (APBH_CH15_BAR)	32	R/W	0000_0000h	15.5.11/477
180_47D0	APBH DMA Channel n Semaphore Register (APBH_CH15_SEMA)	32	R/W	0000_0000h	15.5.12/478
180_47E0	AHB to APBH DMA Channel n Debug Information (APBH_CH15_DEBUG1)	32	R/W	00A0_0000h	15.5.13/479
180_47F0	AHB to APBH DMA Channel n Debug Information (APBH_CH15_DEBUG2)	32	R/W	0000_0000h	15.5.14/482
180_4800	APBH Bridge Version Register (APBH_VERSION)	32	R/W	0301_0000h	15.5.15/483

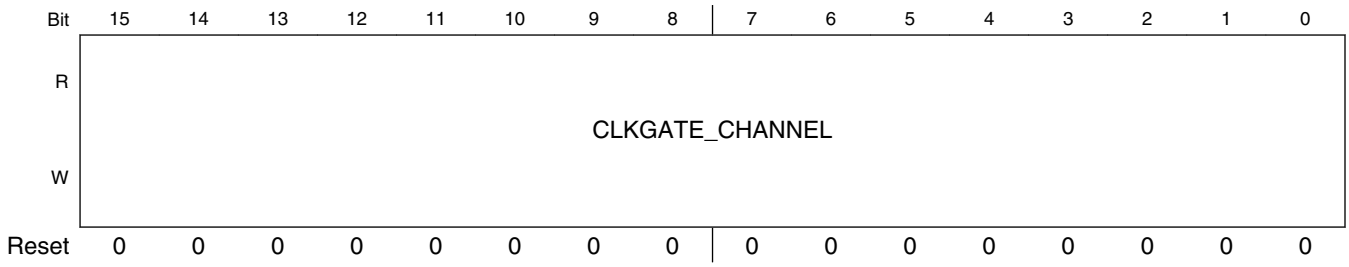
15.5.1 AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0*n*)

The APBH CTRL 0 provides overall control of the AHB to APBH bridge and DMA.

This register contains module softreset, clock gating, channel clock gating/freeze bits.

Address: 180_4000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	AHB_BURST8_EN	APB_BURST_EN	Reserved											
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0



APBH_CTRL0*n* field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal APBH DMA operation. Set this bit to one (default) to disable clocking with the APBH DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBH DMA block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 AHB_BURST8_EN	Set this bit to one (default) to enable AHB 8-beat burst. Set to zero to disable 8-beat burst on AHB interface.
28 APB_BURST_EN	Set this bit to one to enable apb master do a continous transfers when a device request a burst dma. Set to zero will treat a burst dma request as 4/8 individual requests.
27–16 RSVD0	This field is reserved. Reserved, always set to zero.
CLKGATE_CHANNEL	These bits must be set to zero for normal operation of each channel. When set to one they gate off the individual clocks to the channels. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** — 0x0080 **NAND7** — 0x0100 **SSP** —

15.5.2 AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1*n*)

The APBH CTRL one provides overall control of the interrupts generated by the AHB to APBH DMA. This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
```

```
                    BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register
struct's bitfield
```

Address: 180_4000h base + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CH15_CMDCMPLT_IRQ_EN	CH14_CMDCMPLT_IRQ_EN	CH13_CMDCMPLT_IRQ_EN	CH12_CMDCMPLT_IRQ_EN	CH11_CMDCMPLT_IRQ_EN	CH10_CMDCMPLT_IRQ_EN	CH9_CMDCMPLT_IRQ_EN	CH8_CMDCMPLT_IRQ_EN	CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15_CMDCMPLT_IRQ	CH14_CMDCMPLT_IRQ	CH13_CMDCMPLT_IRQ	CH12_CMDCMPLT_IRQ	CH11_CMDCMPLT_IRQ	CH10_CMDCMPLT_IRQ	CH9_CMDCMPLT_IRQ	CH8_CMDCMPLT_IRQ	CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CTRL1*n* field descriptions

Field	Description
31 CH15_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 15.
30 CH14_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 14.
29 CH13_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 13.
28 CH12_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 12.
27 CH11_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 11.
26 CH10_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 10.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
25 CH9_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 9.
24 CH8_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 8.
23 CH7_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 7.
22 CH6_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 6.
21 CH5_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 5.
20 CH4_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 4.
19 CH3_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 3.
18 CH2_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 2.
17 CH1_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 1.
16 CH0_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 0.
15 CH15_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
14 CH14_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
13 CH13_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
12 CH12_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
11 CH11_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
10 CH10_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
9 CH9_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
8 CH8_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
7 CH7_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6 CH6_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5 CH5_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4 CH4_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3 CH3_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2 CH2_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
1 CH1_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0 CH0_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

15.5.3 AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2*n*)

The APBH CTRL 2 provides channel error interrupts generated by the AHB to APBH DMA. This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
                    BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
                    BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register
struct's bitfield
```

Address: 180_4000h base + 20h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CH15_ERROR_STATUS	CH14_ERROR_STATUS	CH13_ERROR_STATUS	CH12_ERROR_STATUS	CH11_ERROR_STATUS	CH10_ERROR_STATUS	CH9_ERROR_STATUS	CH8_ERROR_STATUS	CH7_ERROR_STATUS	CH6_ERROR_STATUS	CH5_ERROR_STATUS	CH4_ERROR_STATUS	CH3_ERROR_STATUS	CH2_ERROR_STATUS	CH1_ERROR_STATUS	CH0_ERROR_STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15_ERROR_IRQ	CH14_ERROR_IRQ	CH13_ERROR_IRQ	CH12_ERROR_IRQ	CH11_ERROR_IRQ	CH10_ERROR_IRQ	CH9_ERROR_IRQ	CH8_ERROR_IRQ	CH7_ERROR_IRQ	CH6_ERROR_IRQ	CH5_ERROR_IRQ	CH4_ERROR_IRQ	CH3_ERROR_IRQ	CH2_ERROR_IRQ	CH1_ERROR_IRQ	CH0_ERROR_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CTRL2*n* field descriptions

Field	Description
31 CH15_ERROR_STATUS	Error status bit for APBH DMA Channel 15. Valid when corresponding Error IRQ is set. 1 - AHB bus error

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
	0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
30 CH14_ERROR_ STATUS	Error status bit for APBH DMA Channel 14. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
29 CH13_ERROR_ STATUS	Error status bit for APBH DMA Channel 13. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
28 CH12_ERROR_ STATUS	Error status bit for APBH DMA Channel 12. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
27 CH11_ERROR_ STATUS	Error status bit for APBH DMA Channel 11. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
26 CH10_ERROR_ STATUS	Error status bit for APBH DMA Channel 10. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
25 CH9_ERROR_ STATUS	Error status bit for APBH DMA Channel 9. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
24 CH8_ERROR_ STATUS	Error status bit for APBH DMA Channel 8. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
	0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
23 CH7_ERROR_STATUS	Error status bit for APBX DMA Channel 7. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
22 CH6_ERROR_STATUS	Error status bit for APBX DMA Channel 6. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
21 CH5_ERROR_STATUS	Error status bit for APBX DMA Channel 5. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
20 CH4_ERROR_STATUS	Error status bit for APBX DMA Channel 4. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
19 CH3_ERROR_STATUS	Error status bit for APBX DMA Channel 3. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
18 CH2_ERROR_STATUS	Error status bit for APBX DMA Channel 2. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
17 CH1_ERROR_STATUS	Error status bit for APBX DMA Channel 1. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
16 CH0_ERROR_STATUS	Error status bit for APBX DMA Channel 0. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
15 CH15_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
14 CH14_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
13 CH13_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
12 CH12_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
11 CH11_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
10 CH10_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
9 CH9_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
8 CH8_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
7 CH7_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
6 CH6_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
5 CH5_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
4 CH4_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
3 CH3_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
2 CH2_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
1 CH1_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
0 CH0_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

15.5.4 AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL*n*)

The APBH CHANNEL CTRL provides reset/freeze control of each DMA channel. This register contains individual channel reset/freeze bits.

Address: 180_4000h base + 30h offset + (4d × i), where i=0d to 3d

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

APBH_CHANNEL_CTRL*n* field descriptions

Field	Description
31–16 RESET_ CHANNEL	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** — 0x0080 **NAND7** — 0x0100 **SSP** —
FREEZE_ CHANNEL	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is deined access to the central DMA resources. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** —

Table continues on the next page...

APBH_CHANNEL_CTRL*n* field descriptions (continued)

Field	Description
	0x0080 **NAND7** — 0x0100 **SSP** —

15.5.5 AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)

This register allows reassignment of the APBH device connected to the DMA Channels.

In this chip, APBH DMA channel resource is enough for high speed peripherals, so this register is of no use and reserved.

Address: 180_4000h base + 40h offset = 180_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

APBH_DEVSEL field descriptions

Field	Description
31–30 CH15	This field is reserved. Reserved.
29–28 CH14	This field is reserved. Reserved.
27–26 CH13	This field is reserved. Reserved.
25–24 CH12	This field is reserved. Reserved.
23–22 CH11	This field is reserved. Reserved.
21–20 CH10	This field is reserved. Reserved.
19–18 CH9	This field is reserved. Reserved.
17–16 CH8	This field is reserved. Reserved.
15–14 CH7	This field is reserved. Reserved.
13–12 CH6	This field is reserved. Reserved.

Table continues on the next page...

APBH_DEVSEL field descriptions (continued)

Field	Description
11–10 CH5	This field is reserved. Reserved.
9–8 CH4	This field is reserved. Reserved.
7–6 CH3	This field is reserved. Reserved.
5–4 CH2	This field is reserved. Reserved.
3–2 CH1	This field is reserved. Reserved.
CH0	This field is reserved. Reserved.

15.5.6 AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)

This register programs the apbh burst size of the APBH DMA devices when a DMA burst request is issued.

This register provides a mechanism for assigning the device.

Address: 180_4000h base + 50h offset = 180_4050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		CH8	
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

APBH_DMA_BURST_SIZE field descriptions

Field	Description
31–30 CH15	This field is reserved. Reserved.
29–28 CH14	This field is reserved. Reserved.
27–26 CH13	This field is reserved. Reserved.
25–24 CH12	This field is reserved. Reserved.
23–22 CH11	This field is reserved. Reserved.

Table continues on the next page...

APBH_DMA_BURST_SIZE field descriptions (continued)

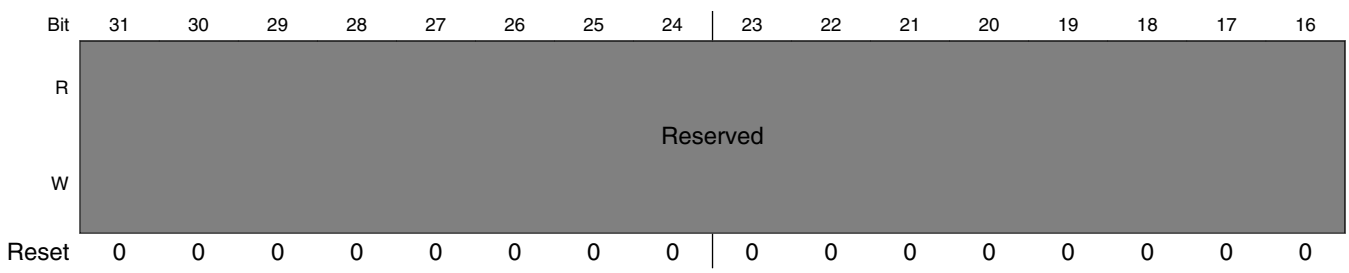
Field	Description
21–20 CH10	This field is reserved. Reserved.
19–18 CH9	This field is reserved. Reserved.
17–16 CH8	DMA burst size for SSP. 0x0 **BURST0** — 0x1 **BURST4** — 0x2 **BURST8** —
15–14 CH7	DMA burst size for GPMI channel 7. Do not change. GPMI only support burst size 4.
13–12 CH6	DMA burst size for GPMI channel 6. Do not change. GPMI only support burst size 4.
11–10 CH5	DMA burst size for GPMI channel 5. Do not change. GPMI only support burst size 4.
9–8 CH4	DMA burst size for GPMI channel 4. Do not change. GPMI only support burst size 4.
7–6 CH3	DMA burst size for GPMI channel 3. Do not change. GPMI only support burst size 4.
5–4 CH2	DMA burst size for GPMI channel 2. Do not change. GPMI only support burst size 4.
3–2 CH1	DMA burst size for GPMI channel 1. Do not change. GPMI only support burst size 4.
CH0	DMA burst size for GPMI channel 0. Do not change. GPMI only support burst size 4.

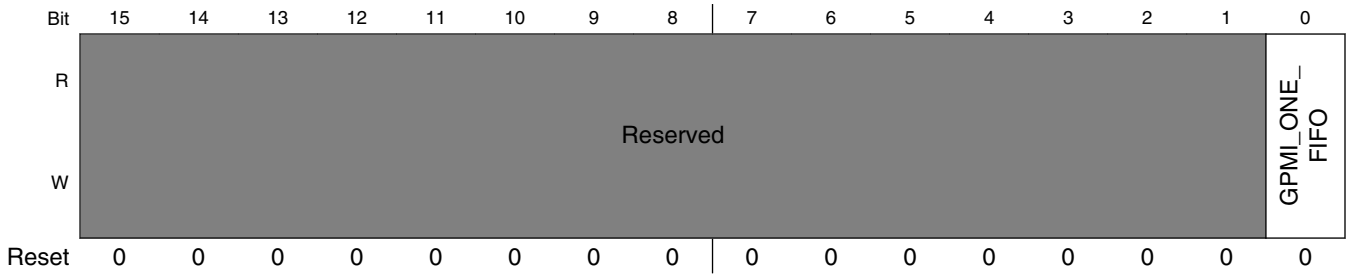
15.5.7 AHB to APBH DMA Debug Register (APBH_DEBUG)

This register is for debug purpose.

The debug register is for internal use only. Not recommend for customer useage.

Address: 180_4000h base + 60h offset = 180_4060h





APBH_DEBUG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved, always set to zero.
0 GPMI_ONE_FIFO	Set to 0ne and the 8 GPMI channels will share the DMA FIFO, and when set to zero, the 8 GPMI channels will use its own DMA FIFO.

15.5.8 APBH DMA Channel n Current Command Address Register (APBH_CH*n*_CURCMDAR)

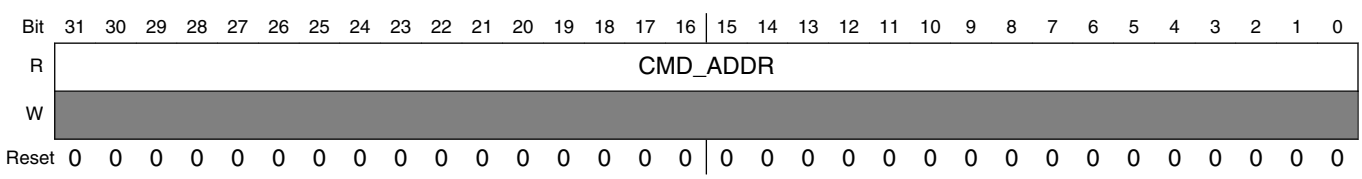
The APBH DMA channel n current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel n is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE

```
pCurCmd = (apbh_chn_cmd_t *) APBH_CHn_CURCMDAR_RD(0);              // read the whole
register, since there is only one field
pCurCmd = (apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 0, CMD_ADDR);  // or, use multi-
register bitfield read macro
pCurCmd = (apbh_chn_cmd_t *) APBH_CHn_CURCMDAR(0).CMD_ADDR;        // or, assign from
bitfield of indexed register's struct
```

Address: 180_4000h base + 100h offset + (112d × i), where i=0d to 15d



APBH_CH*n*_CURCMDAR field descriptions

Field	Description
CMD_ADDR	Pointer to command structure currently being processed for channel n.

15.5.9 APBH DMA Channel n Next Command Address Register (APBH_CH*n*_NXTCMDAR)

The APBH DMA Channel n Next Command Address register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

APBH DMA Channel n is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel n semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE

```
APBH_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure);         // write the entire
register, since there is only one field
BF_WRn(APBH_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure);       // or, use multi-
register bitfield write macro
APBH_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure;  // or, assign to bitfield
of indexed register's struct
```

Address: 180_4000h base + 110h offset + (112d × i), where i=0d to 15d

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

APBH_CH*n*_NXTCMDAR field descriptions

Field	Description
CMD_ADDR	Pointer to next command structure for channel n.

15.5.10 APBH DMA Channel n Command Register (APBH_CH*n*_CMD)

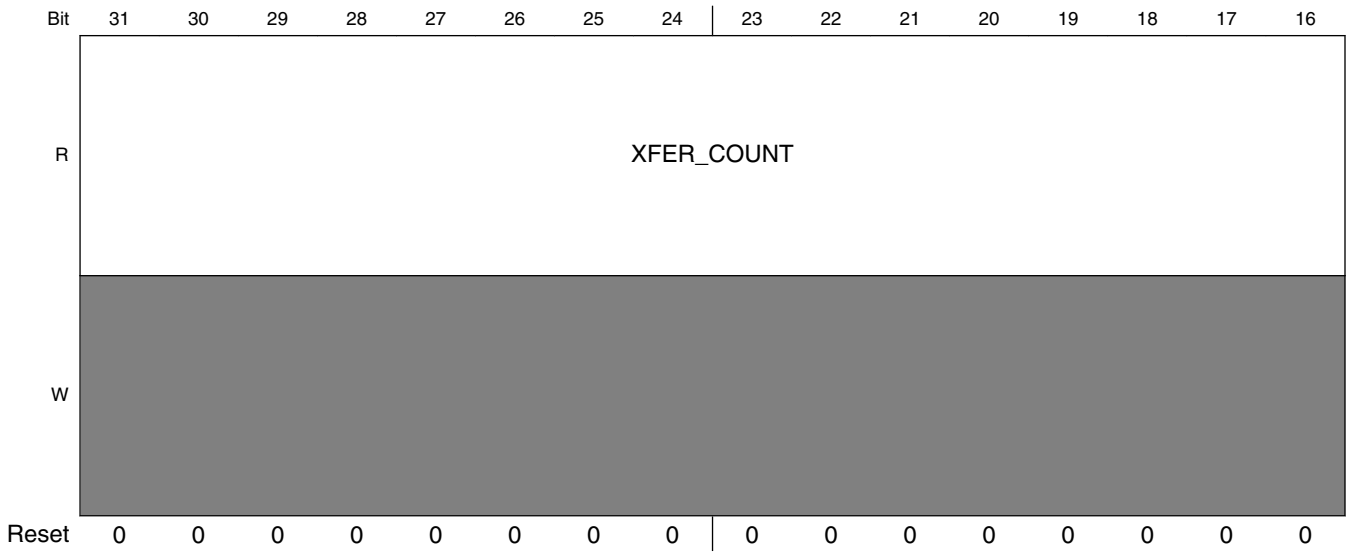
The APBH DMA Channel n command register specifies the DMA transaction to perform for the current command chain item.

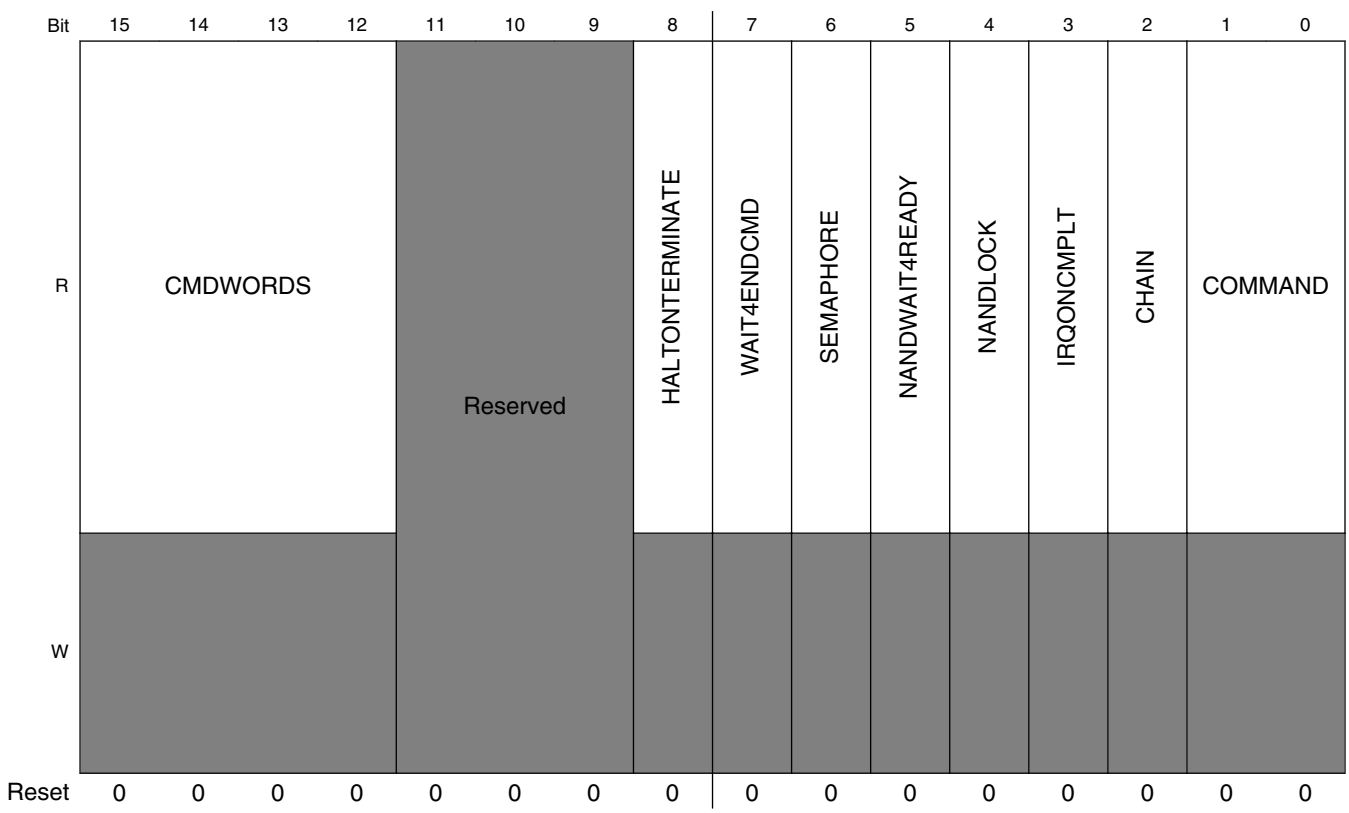
The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE

```
apbh_chn_cmd_t  dma_cmd;
dma_cmd.XFER_COUNT = 512;                                    // transfer 512 bytes
dma_cmd.COMMAND = BV_APBH_CHn_CMD_COMMAND__DMA_WRITE;  // transfer to system memory from
peripheral device
dma_cmd.CHAIN = 1;                                           // chain an additional command
structure on to the list
dma_cmd.IRQONCMPLT = 1;                                      // generate an interrupt on
completion of this command structure
```

Address: 180_4000h base + 120h offset + (112d × i), where i=0d to 15d





APBH_CH*n*_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI0 device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI0, starting with the base PIO address of the GPMI0 control register and incrementing from there. Zero means transfer NO command words
11–9 -	This field is reserved. Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel will will wait until the NAND device reports "ready" before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel will remain "locked" in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in APBH_CHn_CMDAR to find the next command.
COMMAND	This bitfield indicates the type of current command: 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

15.5.11 APBH DMA Channel n Buffer Address Register (APBH_CH*n*_BAR)

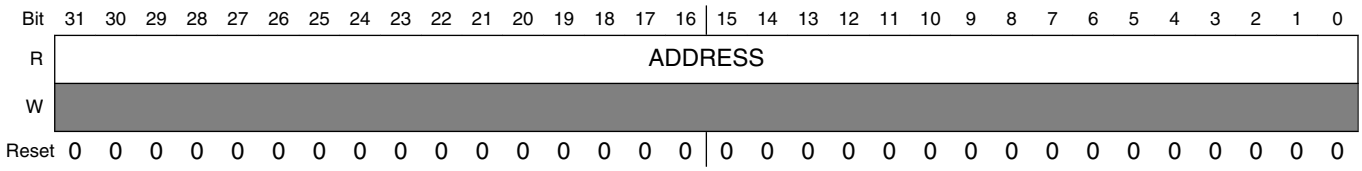
The APBH DMA Channel n buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE

```
apbh_chn_bar_t  dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

Address: 180_4000h base + 130h offset + (112d × i), where i=0d to 15d



APBH_CH*n*_BAR field descriptions

Field	Description
ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

15.5.12 APBH DMA Channel n Semaphore Register (APBH_CH*n*_SEMA)

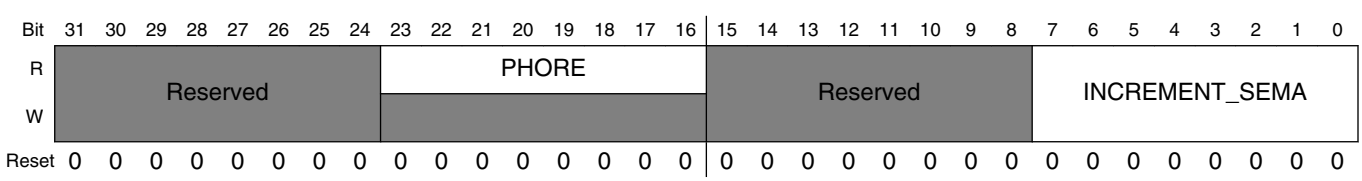
The APBH DMA Channel n semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBH_CHn_SEMA, 0, INCREMENT_SEMA, 2);     // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 0, PHORE);  // get instantaneous value
```

Address: 180_4000h base + 140h offset + (112d × i), where i=0d to 15d



APBH_CH*n*_SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero.
INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

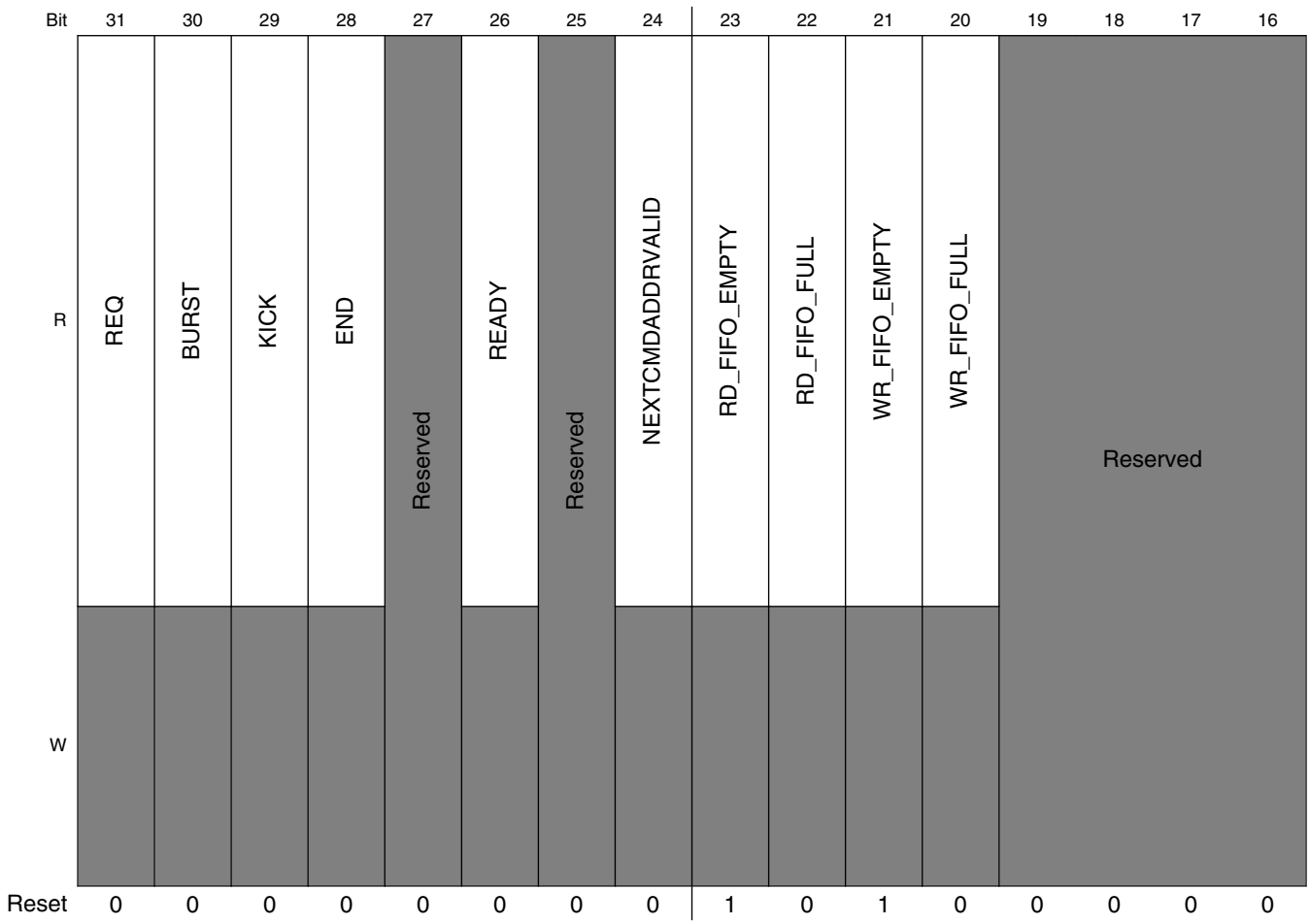
15.5.13 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG1)

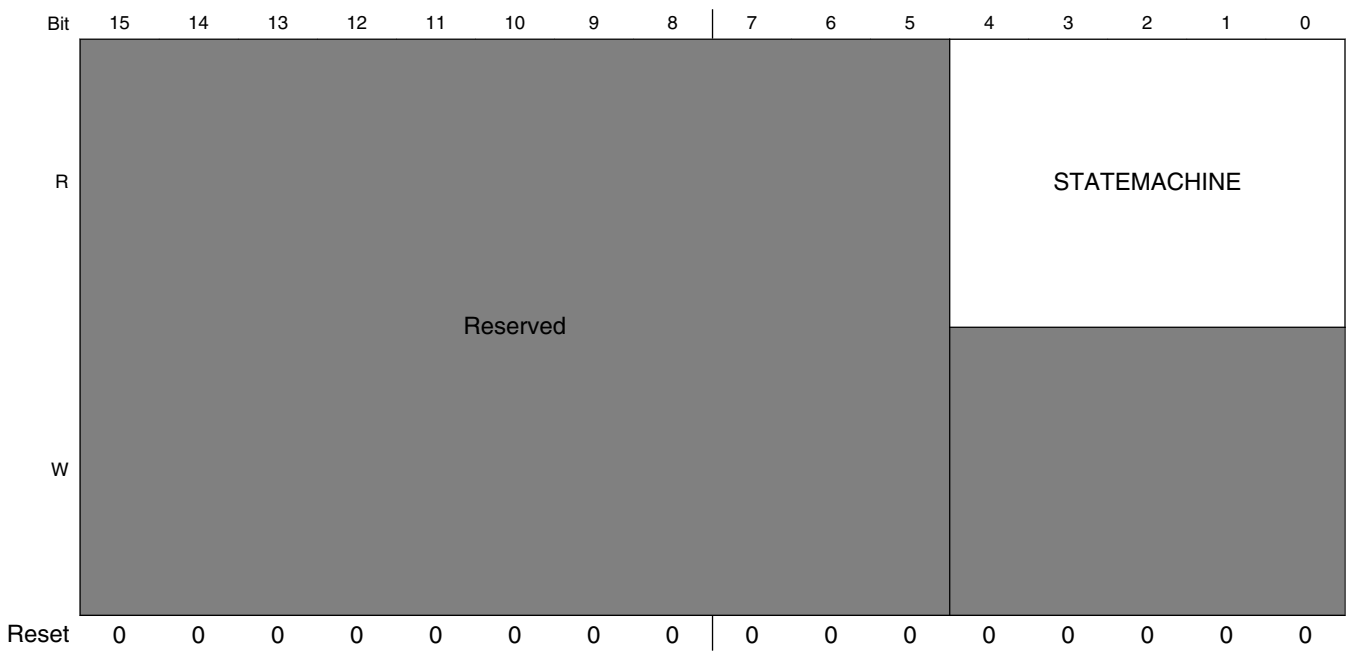
This register gives debug visibility into the APBH DMA Channel n state machine and controls.

This register allows debug visibility of the APBH DMA Channel n.

APBH Memory Map/Register Definition

Address: 180_4000h base + 150h offset + (112d × i), where i=0d to 15d





APBH_CH*n*_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This field is reserved. This bit is reserved for this DMA Channel and always reads 0.
26 READY	This bit is reserved for this DMA Channel and always reads 0.
25 LOCK	This field is reserved. This bit is reserved for this Channel and always reads 0.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	This field is reserved. Reserved
STATEMACHINE	PIO Display of the DMA Channel n state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.

Table continues on the next page...

APBH_CH*n*_DEBUG1 field descriptions (continued)

Field	Description
	0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

15.5.14 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel n.

This register allows debug visibility of the APBH DMA Channel n.

Address: 180_4000h base + 160h offset + (112d × i), where i=0d to 15d

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

APBH_CH*n*_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

15.5.15 APBH Bridge Version Register (APBH_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (APBH_VERSION.B.MAJOR != 3)
   Error();
```

Address: 180_4000h base + 800h offset = 180_4800h

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

APBH_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 16
Asynchronous Sample Rate Converter (ASRC)

16.1 Overview

The Asynchronous Sample Rate Converter (ASRC) converts the sampling rate of a signal associated with an input clock into a signal associated with a different output clock.

The ASRC supports concurrent sample rate conversion of up to 10 channels of about -120dB THD+N. The ASRC supports up to three sampling rate pairs.

The incoming audio data to this chip may be received from various sources at different sampling rates. The outgoing audio data of this chip may have different sampling rates and it can also be associated with output clocks that are asynchronous to the input clocks.

The ASRC is implemented as a co-processor in hardware, with minimal ARM Platform intervention required.

Figure 16-1 is a system view of the connection between the ASRC block and other blocks.

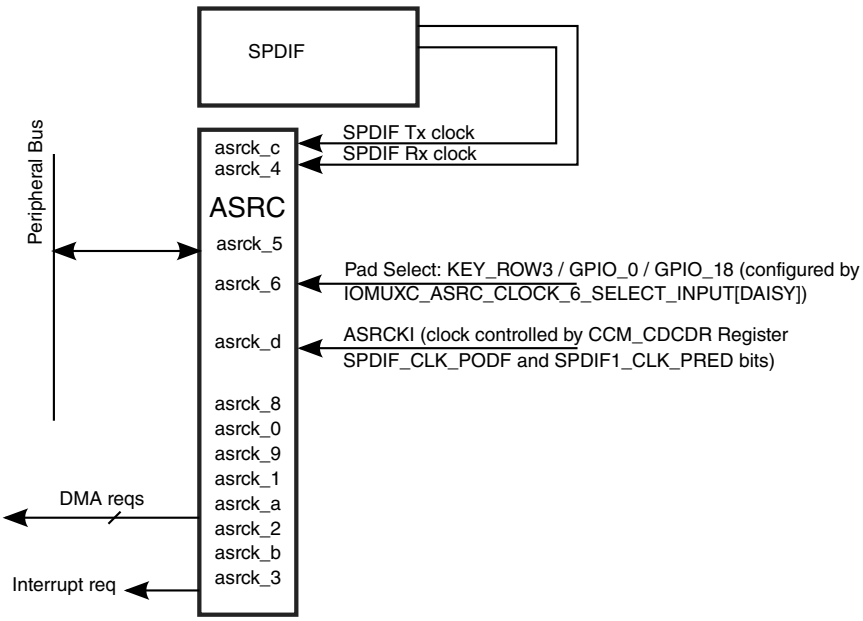


Figure 16-1. General System Overview

Figure 16-2 is the ASRC block diagram.

The red dotted line designates the ASRC block. Objects outside the dotted line represent SoC-level resources.

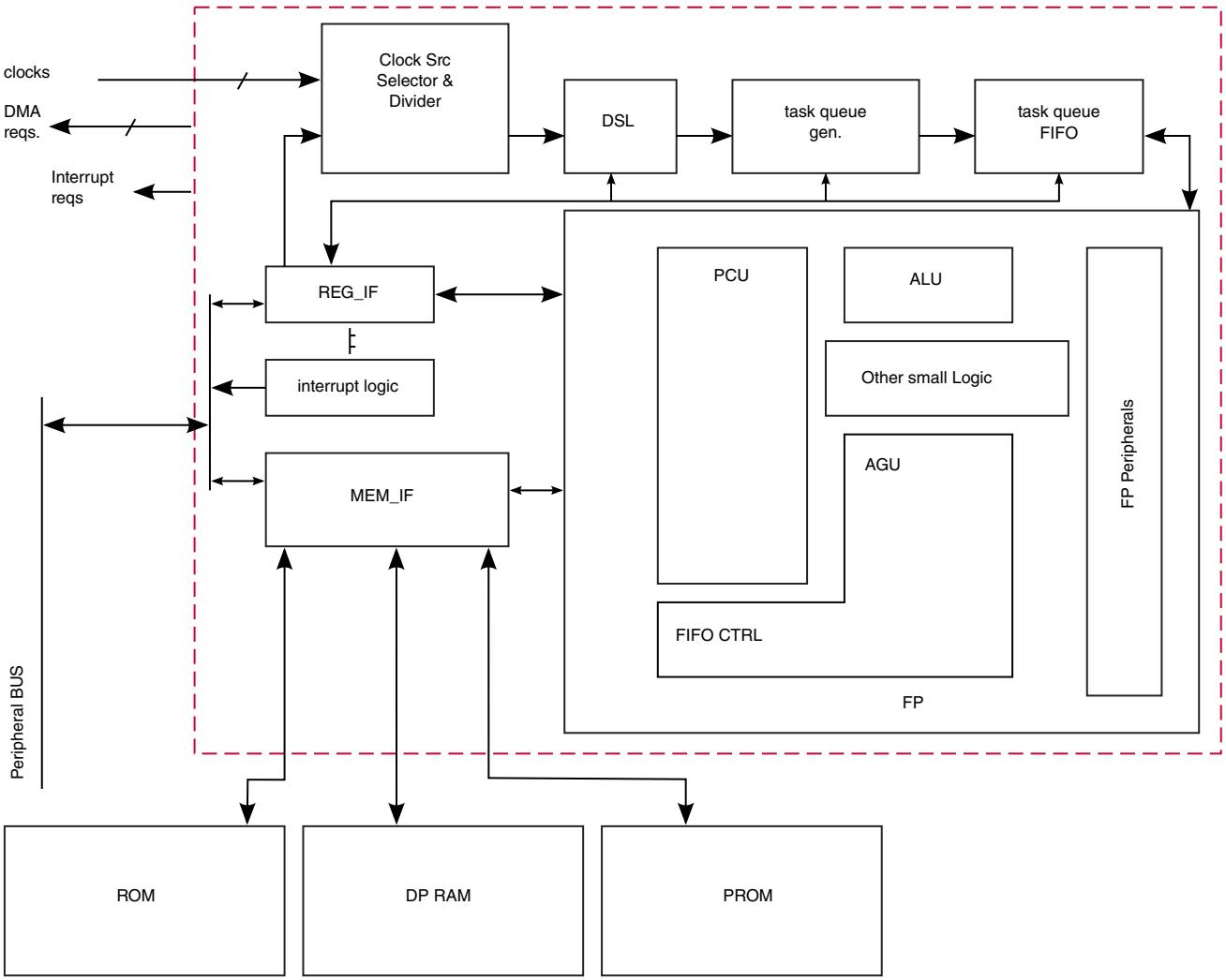


Figure 16-2. ASRC block diagram

16.1.1 Features

Table 16-1. ASRC Specifications

Parameters	Test Conditions	Minimum	Typical	Maximum	Unit
Channels Supported		0 [1]		10	
Pairs of Rate Conversion		1	-	3	
THD+N	120 MHz < Fs_{ASRC} [2] <160 MHz		-120		dB
Dynamic Range				144	dB
Settling Time			40		ms
Comment:					

1. When a pair has zero channels, the pair will be disabled, although the pair enable bit may be set in ASRCTR register.
2. Fs_{ASRC} is the processing clock of ASRC block.

Other Features:

- Support user-programmable threshold for the input/output FIFOs.
- Support flexible 8/16/24-bit width of input data, and 16/24-bit width of output data.
- Designed for rate conversion between 44.1 kHz, 32 kHz, 48 kHz, 96 kHz, and 192 kHz. The useful signal bandwidth is below 24 kHz.
- Other input sampling rates in the range of 8 kHz to 200 kHz is also supported, but possibly with less desirable bandwidth.
- Other output sampling rates in the range of 30 kHz to 200 kHz is also supported, but possibly with less desirable bandwidth.
- Automatic accommodation to slow variations in the incoming and outgoing sampling rates.
- Linear phase
- Tolerant to sample clock jitter

Clock/Data Connections

- The sampling rate clocks are directly connected to the ASRC block, the ratio estimation of the input clocks with output clocks are done in ASRC hardware.
- The clock signals come from the following blocks, for example:
  - Receiving bit clock and transmitting bit clock
  - SPDIF, receiving bit clock and transmitting bit clock
  - other audio peripherals etc.
- The exchange of audio data is done by the processor accessing ASRC block through registers defined on shared peripheral bus.

16.1.2 Modes of Operation

See ASRC Memory Map/Register Definition for a definition of the registers and parameters used in ASRC.

16.1.2.1 Data Transfer Schemes

16.1.2.1.1 Data Input Modes

The input mode for each of the three channel sets may be set independently. Three modes of supplying data to the ASRC input FIFOs are available:

- Polling
- Interrupt
- DMA

In all input-data transfer schemes, the ASRC fetches data from each enabled FIFO and processes the data sample-by-sample after each rising edge of the associated input sampling clock until the FIFO level reaches a threshold.

After the threshold is reached, the ASRC requests data. The FIFO size for each channel set is 64 samples and the threshold is set at 32 samples.The threshold can be defined by interface registers ASRMCRx, x=A, B or C.

If the ASRC attempts to fetch data from an empty FIFO, an error is generated and the ASRSTR_AOLE bit is set. If the ASRC overload interrupt is enabled (ASRIER_AOLIE bit is set), an interrupt is generated.

When writing data to an input FIFO, you must ensure that it is in a predefined sequence. For example, when writing to an input FIFO, the sequence should be: channel_0, channel_1, channel_2,..., channel_n, channel_0, channel_1, channel_2, etc. Here channel_n stands for the data intended for the n-th channel. The hardware will re-allocate each data to its corresponding channel FIFO. The channel being re-allocated is shown by ASRCCR_ACIx, x=A,B or C.

Mode 1 (Polling Mode)

Polling mode is the default mode following power-on or individual reset, and is selected by clearing the associated channel set A, B, or C data-input interrupt enable bit (ASRIER_ADIEx, where x=A, B or C). In this mode, data-input interrupts are disabled. When the FIFO level is below the threshold, the associated status bit (ASRSTR_AIDEx, where x=A, B, or C) is set. To clear the status bit, the FIFO must be written with enough data to raise the level above the threshold.

Mode 2 (Interrupt Mode)

The ASRC input FIFOs can also be serviced by interrupts. To enable interrupts, the corresponding data-input interrupt enable bits (ASRIER_ADIEx, where x=A, B, or C) should be set. An interrupt is automatically generated any time the input FIFO level is below the threshold. The interrupt is cleared when enough data is written to the FIFO to raise the level above the threshold.

Mode 3 (DMA Mode)

The ASRC input FIFOs can also be filled using DMA. In this mode, the data-input interrupt-enable bits (ASRIER_ADIEx, where x=A, B, or C) should be cleared and the DMA controller should be configured to use the ASRC as a request source.

16.1.2.1.2 Data Output Modes

The output mode for each of the 3 channel sets (A, B, and C) may be set independently.

Three modes of retrieving data from the ASRC output FIFOs are available:

- Polling
- Interrupt
- DMA

In all output-data transfer schemes, the ASRC places a processed sample into the associated output FIFO. After a threshold is reached, the ASRC requests that data be transferred out of the FIFO.

The FIFO size for each channel set is 64 samples and the threshold is set at 32 samples. The threshold can be defined by interface registers ASRMCRx, x=A, B or C.

If the ASRC attempts to place data into a FIFO that is already full, an error is generated and the ASRSTR_AOLE bit is set. If the ASRC overload interrupt is enabled (ASRIER_AOLIE bit is set), an interrupt is generated.

Each output FIFO is organized in the same channel order in which the associated input FIFO was written.

Three transfer modes are supported by Interface Block.

Mode 1 (Polling Mode)

The ASRC output FIFOs can be serviced by polling. In this mode, ensure the associated output-data interrupt enable bit (ASRIER_ADOEx, where x=A, B, or C) is cleared. In this mode, all output-data interrupts are disabled. Any time the output FIFO exceeds the threshold the associated status bit (ASRSTR_AODFx, where x=A, B, or C) is set. To clear the status bit, enough data must be read from the associated output FIFO to lower the level below the threshold.

Mode 2 (Interrupt Mode)

The ASRC output FIFOs may also be serviced using interrupts. To enable this mode, the corresponding output-data interrupt-enable bits (ASRIER_ADOEx, where x=A, B, or C) should be set. Any time the output FIFO level exceeds the threshold, an interrupt is automatically generated. The interrupt is cleared when enough data is read from the FIFO to lower the level below the threshold.

Mode 3 (DMA Mode)

The ASRC output FIFOs can also be read using DMA. In this mode, the output-data interrupt-enable bits (ASRIER_ADOEx, where x=A, B, or C) should be cleared and the DMA controller should be configured to use the ASRC as a request source.

16.1.2.2 Word Alignment Supported

16.1.2.2.1 Input Data Alignment Modes

The position and length of input data word to the input data FIFOs ASRDIA, ASRDIB, ASRDIC are programmable. The control bits are defined in ASRMCR1x {x=A, B, or C}. It supports the following modes.

Table 16-2. Input Data Alignment

Format	Bit Number																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-bit LSB Aligned																									7	6	5	4	3	2	1	0
8-bit MSB Aligned																	7	6	5	4	3	2	1	0								
16-bit LSB Aligned																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit MSB Aligned	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
24-bit LSB Aligned									23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit MSB Aligned	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

16.1.2.2.2 Output Data Alignment Modes

The position and length of output data word from the output data FIFOs ASRDOA, ASRDOB, ASRDOC are programmable. The control bits are defined in ASRMCR1x {x=A, B, or C}. It supports the following modes.

Table 16-3. Output Data Alignment

Format	Bit Number																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit LSB Aligned																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit LSB Aligned with Sign Extension	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit MSB Aligned	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
24-bit LSB Aligned									23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

Table 16-3. Output Data Alignment (continued)

Format	Bit Number																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit LSB Aligned with Sign Extension	23	23	23	23	23	23	23	23	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit MSB Aligned	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

16.2 Clocks

The table found here describes the clock sources for ASRC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 16-4. ASRC Clocks

Clock name	Clock Root	Description
asrck_clock_d	spdif1_clk_root	ASRC module clock (SPDIF clock)
ipg_clk	ahb_clk_root	Peripheral clock
mem_clk	ahb_clk_root	Peripheral access clock

16.3 Interrupts

ASRC has several interrupts events.

The priorities are shown as Table 16-5 .

Table 16-5. Interrupt Priorities/Vector

Priority	Description
lowest	ASRC Pair A input data needed
	ASRC Pair B input data needed
	ASRC Pair C input data needed
	ASRC Pair A output data ready
	ASRC Pair B output data ready
	ASRC Pair C output data ready
	ASRC Overload

16.4 DMA requests

ASRC has six DMA requests. They are directly connected to the lowest six status bits in the ASRSTR register.

Table 16-6. DMA requests

Type	Description
0	ASRC Pair A input data needed
1	ASRC Pair B input data needed
2	ASRC Pair C input data needed
3	ASRC Pair A output data ready
4	ASRC Pair B output data ready
5	ASRC Pair C output data ready

16.5 Functional Description

This section provides a complete functional description of the block.

16.5.1 Algorithm Description

16.5.1.1 Signal processing flow

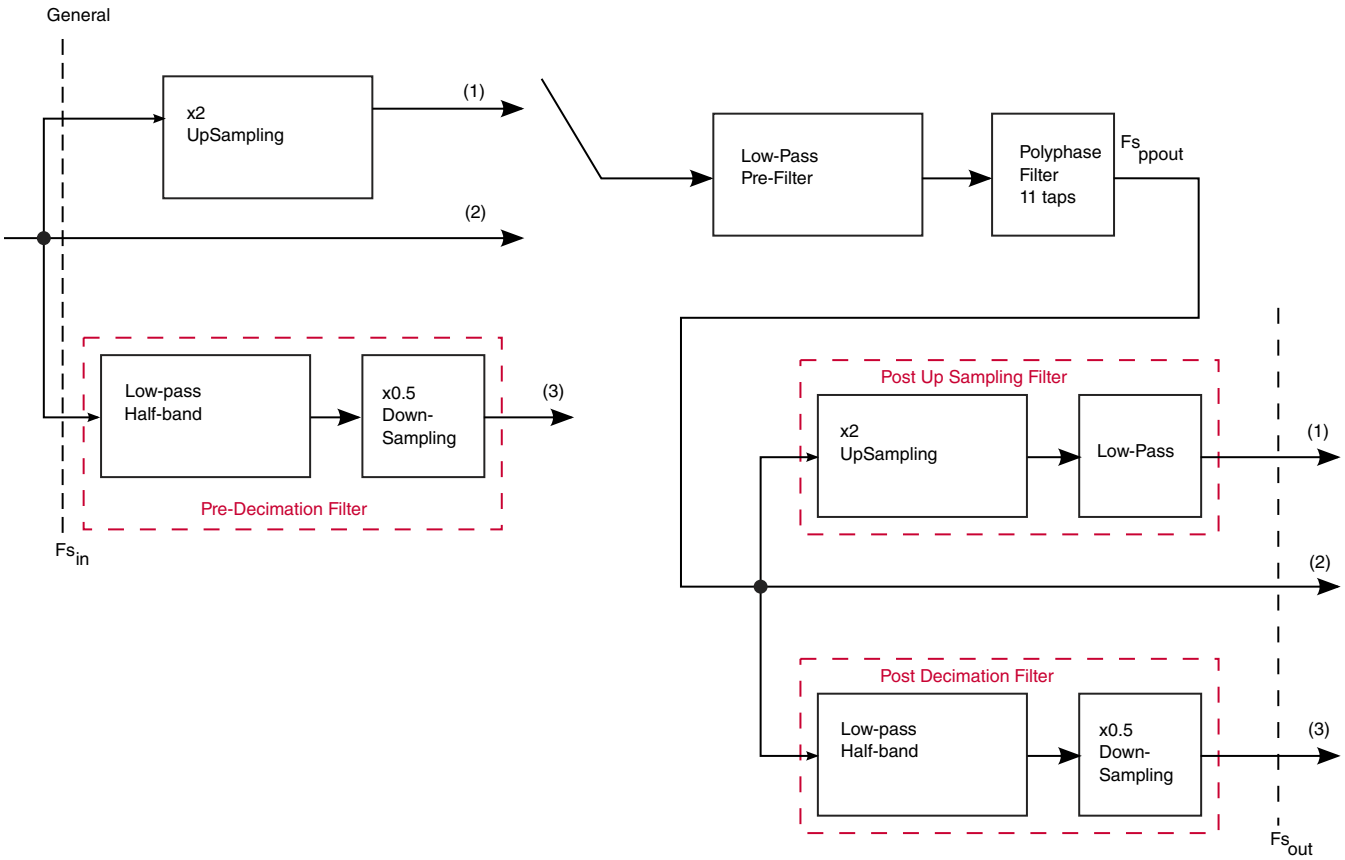


Figure 16-3. Signal processing configurations

The figure above shows the possible configurations of the ASRC. Each configuration consists of 2 to 4 stages.

- x2 up-sampling rate expander (zero insertion only) (input branch 1), direct connection (input branch 2), or low-pass pre decimation filter (consisting of a low-pass half-band FIR filter with x0.5 downsampling rate decimator) (input branch 3),
- low-pass pre-filter, the low-pass bandwidth is at most 0.25 x Fs, where Fs is the sampling rate of the input signal to this low-pass pre-filter,
- polyphase filter,
- x2 post upsampling filter (consisting of a x2 up-sampling rate expander (zero insertion only) with low-pass half-band FIR filter) (output branch 1), direct connection (output branch 2), or low-pass post decimation filter (consisting of a low-pass half-band FIR filter with x0.5 downsampling rate decimator) (output branch 3).

By flowing through different processing branches and different setups of the pre-filter, this ASRC scheme can be used to handle different rate conversion requirements.

- Configuration (a): Input Branch 1+Output Branch 1:

The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/2$.
The signal sampling rate of the polyphase filter output is $Fs_{ppout} = Fs_{out}/2$.

- Configuration (b): Input Branch 1+Output Branch 2:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/2$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = Fs_{out}$.

- Configuration (c): Input Branch 1+Output Branch 3:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/2$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = 2Fs_{out}$.

- Configuration (d): Input Branch 2+Output Branch 1:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/4$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = Fs_{out}/2$.

- Configuration (e): Input Branch 2+Output Branch 2:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/4$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = Fs_{out}$.

- Configuration (f): Input Branch 2+Output Branch 3:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/4$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = 2Fs_{out}$.

- Configuration (g): Input Branch 3+Output Branch 1:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/8$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = Fs_{out}/2$.

- Configuration (h): Input Branch 3+Output Branch 2:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/8$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = Fs_{out}$.

- Configuration (i): Input Branch 3+Output Branch 3:

  The signal bandwidth observed before the polyphase filter is at most $BW_{in} = Fs_{in}/8$.
  The signal sampling rate of the polyphase filter output is $Fs_{ppout} = 2Fs_{out}$.

Table 16-7. Pre-processing, post-processing options

{Pre_Proc, Post_Proc}		Fsout (KHz)											
		8	12	16	24	32	44.1	48	64	88.2	96	128	192
Fsin (KHz)	8	{0,1}	{0,1}	{0,1}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}
	12	{0,2}	{0,1}	{0,1}	{0,1}	{0,1}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}
	16	{1,2}	{0,2}	{0,1}	{0,1}	{0,1}	{0,1}	{0,1}	{0,0}	{0,0}	{0,0}	{0,0}	{0,0}
	24	{1,2}	{1,2}	{0,2}	{0,1}	{0,1}	{0,1}	{0,1}	{0,1}	{0,0}	{0,0}	{0,0}	{0,0}
	32	{1,2}	{1,2}	{1,2}	{0,2}	{0,1}	{0,1}	{0,1}	{0,1}	{0,1}	{0,0}	{0,0}	{0,0}
	44.1	{2,2}	{1,2}	{1,2}	{0,2}	{0,2}	{0,1}	{0,1}	{0,1}	{0,1}	{0,1}	{0,0}	{0,0}
	48	{2,2}	{1,2}	{1,2}	{1,2}	{0,2}	{0,2}	{0,1}	{0,1}	{0,1}	{0,1}	{0,1}	{0,0}
	64	{2,2}	{2,2}	{1,2}	{1,2}	{0,2}	{0,2}	{0,2}	{0,1}	{0,1}	{0,1}	{0,1}	{0,0}
	88.2	NA	{2,2}	{2,2}	{1,2}	{1,2}	{1,2}	{1,2}	{1,1}	{1,1}	{1,1}	{1,1}	{1,1}
	96	NA	{2,2}	{2,2}	{1,2}	{1,2}	{1,2}	{1,2}	{1,1}	{1,1}	{1,1}	{1,1}	{1,1}
	128	NA	NA	{2,2}	{2,2}	{1,2}	{1,2}	{1,2}	{1,1}	{1,1}	{1,1}	{1,1}	{1,1}
	192	NA	NA	NA	{2,2}	{2,2}	{2,2}	{2,2}	{2,1}	{2,1}	{2,1}	{2,1}	{2,1}

NOTE: In the {Pre_Proc, Post_Proc} pair, the meaning of the values are:

Pre_Proc:

- 0 --- Pre-processing Branch 1 as shown in Figure 16-3
- 1 --- Pre-processing Branch 2 as shown in Figure 16-3
- 2 --- Pre-processing Branch 3 as shown in Figure 16-3, decimation-by-2

Post_Proc:

- 0 --- Post-processing Branch 1 as shown in Figure 16-3
- 1 --- Post-processing Branch 2 as shown in Figure 16-3
- 2 --- Post-processing Branch 3 as shown in Figure 16-3

The latencies of the different option can be roughly calculated as follows:

- For PreProc = 0, PostProc = 1 : min latency = constant_A / input-sample-rate + constant_B / output-sample-rate
- For PreProc = 0, PostProc = 0 : min latency = constant_A / input-sample-rate + constant_C / output-sample-rate
- For PreProc = 1, PostProc = 1 : min latency = constant_D / input-sample-rate + constant_B / output-sample-rate

The constants above (e.g., constant_A means the Constant for Preproc = 0, constant_B means the Constant for Postproc = 1, ...) are only influenced by the PreProc/PostProc and (input/output) sampling rate to which they are connected. Input latencies have no relationship with the output latencies, but both elements add together to form the total latencies.

For a rough estimation, the constants can be set as:

- Constant for Preproc = 0: 39

- Constant for Preproc = 1: 78.5
- Constant for Preproc = 2: 235
- Constant for Postproc = 0: 42.5
- Constant for Postproc = 1: 8.5
- Constant for Postproc = 2: 172

The max latency can be derived from this value by using the following formula (where 32 means the input/output FIFO depth that will arouse data transfer):

- max latency = min latency + 32 / input-sample-rate + 32 / output-sample-rate

16.5.1.2 Operation of the Filter

16.5.1.2.1 Support of Physical Clocks

This design supports physical sampling clocks. The clocks can be provided by Sony/ Phillips digital interface (SPDIF), Serial Audio Interface (SAI)media local bus controller (MLB) , Core master clock derivative as ASRCK1 .

Figure 16-4. Clock Source Selector & Divider

Software can set the ASRC Clock Source Register (ASRCSR) and the Clock Divider Register to select the desired clock source and divide it to the needed sample rate clock for use by the ASRC. The clocks have the following restriction. If the prescaler is set to 1, the clock divider can only be set to 1 and the clock source must have a 50% duty cycle.

16.6 Startup Procedure

The following example shows the normal setup procedure for the ASRC block.

```
#include "asrc_common.h"

#include "stdio.h"

#include "soc_api.h"

int incnt=0;

int outcnt=0;

#include "wy_ideal_ratio_dataini_part.h"

WORD   IdealRatio_High=0x04; //

WORD   IdealRatio_Low=0x0; //
```

```
void asrc_config_alloc(WORD ASRCTR_VAL, WORD ASRIER_VAL, WORD ASRCNCR_VAL,
                           WORD ASRCFG_VAL, WORD ASRCDR1_VAL, WORD ASRCDR2_VAL,
                           WORD ASRCSR_VAL)

{  // Disable ASRC

    reg32_write(ASRC_ASRCTR,0x0);

    reg32_write(ASRC_ASRCTR,ASRCTR_VAL);

    reg32_write(ASRC_ASRIER,ASRIER_VAL);

    reg32_write(ASRC_ASRIEM,0x0);

    reg32_write(ASRC_ASRCNCR,ASRCNCR_VAL);

    reg32_write(ASRC_ASRCFG,ASRCFG_VAL);

    reg32_write(ASRC_ASRCDR1,ASRCDR1_VAL);

    reg32_write(ASRC_ASRCDR2,ASRCDR2_VAL);

    reg32_write(ASRC_ASRCSR, ASRCSR_VAL);

    reg32_write(ASRC_ASRPM1, 0x7fffff);

    reg32_write(ASRC_ASRPM2, 0x255555);

    reg32_write(ASRC_ASRPM3, 0xff7280);

    reg32_write(ASRC_ASRPM4, 0xff7280);

    reg32_write(ASRC_ASRPM5, 0xff7280);

    reg32_write(ASRC_ASRQFIFO1,0x001f00);

    reg32_write(ASRC_ASRMCRA,0x001f00);

    reg32_write(ASRC_ASRMCRB,0x001f00);

    reg32_write(ASRC_ASRMCRC,0x001f00);

}

 void sim_ideal_ratio()

  {

    WORD tmp32bit;

    #define ASRSTR_AIDEA_MASK  0x1

    #define ASRSTR_AODFA_MASK  0x1 <<3

    #define ASRSTR_AOLE_MASK   0x1<<6

    #define ASRCTR_DBG_EN             1<<23

    #define ASRCTR_IDRA               1<<13

    #define ASRCTR_USRA               1<<14

    #define ASRC_CLK_PRED_RSTRICTED 0<<28

    #define ASRC_CLK_PRED_DFLT      1<<28 // default: 596MHz div by 2
```

```
    #define ASRC_CLK_PRED_DIV3      2<<28 // 596MHz div by 3

    #define ASRC_CLK_PRED_DIV4      3<<28 // 596MHz div by 4

    #define ASRC_CLK_PRED_DIV5      4<<28 // 596MHz div by 5

    #define ASRC_CLK_PRED_DIV6      5<<28 // 596MHz div by 6

    #define ASRC_CLK_PRED_DIV7      6<<28 // 596MHz div by 7

    #define ASRC_CLK_PRED_DIV8      7<<28 // 596MHz div by 8

    #define ECSPI_CLK_PRED_DFLT     1<<25

    #define ECSPI_CLK_PODF_DFLT     1<<19

    #define ASRC_CLK_PODF_DIV1      0<<9  // pred output divide by 1 again

    #define ASRC_CLK_PODF_DIV2      1<<9  // pred output divide by 2 again

    #define ASRC_CLK_PODF_DIV3      2<<9  // pred output divide by 3 again

    #define ASRC_CLK_PODF_DIV4      3<<9  // pred output divide by 4 again

    #define ASRC_CLK_PODF_DFLT      4<<9  // default: pred output divide by 5 again

    #define ASRC_CLK_PODF_DIV6      5<<9  // pred output divide by 6 again

    #define ASRC_CLK_PODF_DIV7      6<<9  // pred output divide by 7 again

    #define ASRC_CLK_PODF_DIV25     24<<9  // pred output divide by 7 again

    #define IEEE_CLK_PRED_DFLT      1<<6  //

    #define IEEE_CLK_PODF_DFLT      4     //

    #define ASR_HFA_HFB         0

    #define ASR_PREMODA_UP2         0<<6

    #define ASR_PREMODA_DIR         1<<6

    #define ASR_PREMODA_DN2         2<<6

    #define ASR_PREMODA_PAS         3<<6

    #define ASR_POSTMODA_UP2         0<<8

    #define ASR_POSTMODA_DIR         1<<8

    #define ASR_POSTMODA_DN2         2<<8

     reg32_write(CCM_CSCDR2, ASRC_CLK_PRED_DIV8|ECSPI_CLK_PRED_DFLT|ECSPI_CLK_PODF_DFLT|
ASRC_CLK_PODF_DIV25|IEEE_CLK_PRED_DFLT|IEEE_CLK_PODF_DFLT);

     // Disable the ASRC

     reg32_write(ASRC_ASRCTR, 0x0);

     // program AHB clocks

     tmp32bit = reg32_read(CCM_CBCDR);

     tmp32bit = tmp32bit & (~0x00001C00);
```

```
      //tmp32bit = tmp32bit | (0x00000C00); // AHB 100MHz // divided-by-4

      //tmp32bit = tmp32bit | (0x00001000); // AHB 80MHz // divided-by-5

      //tmp32bit = tmp32bit | (0x00001400); // AHB 66MHz // divided-by-6

      //tmp32bit = tmp32bit | (0x00001800); // AHB 57MHz // divided-by-7

      tmp32bit = tmp32bit | (0x00001C00); // AHB 50MHz // divided-by-8

      reg32_write(CCM_CBCDR, tmp32bit); // enable all perihperal clocks during all modes,
except stop mode

      while ( (reg32_read(CCM_CDHIPR) & 0x00008) != 0);

             asrc_config_alloc ( 0x002 | ASRCTR_IDRA | ASRCTR_USRA,    // ASRCTR_VAL, Use
Ratio input, use ideal ratio, Enable Pair A,

             0x0, //0x09,                                              // ASRIER_VAL, Open
PairA input and output interrupt

             0x002,     // ASRCNCR_VAL, assign 2 channels to Pair A

             ASR_PREMODA_DIR | ASR_POSTMODA_DIR | ASR_HFA_HFB,         // ASRCFG_VAL,
POSTMODA=downsampling by 2 ; PREMODA=downsampling by 2

             0x03b03b , // ASRCDR1_VAL, AOCPA=3(FoutA/(2^3)); AICPA=3(FinA/(2^3));
AOCDA=7(div 8); AICDA=7(div 8);

             0x0  ,     // ASRCDR2_VAL,

             0x00d00d   // ASRCSR_VAL,  AOCSA=d: bit clock d: ASRCK1 clk from CCM; AICSA=d:
bit clock d: ASRCK1 clock from CCM;

             );

      reg32_write(ASRC_ASRIDRHA, 0x04); //

      reg32_write(ASRC_ASRIDRLA, 0x0); // Ideal Ratio is set to be 1.

      #define OUTFIFO_THRESH_0  8<<12

      #define INFIFO_THRESH_1   32

      reg32_write(ASRC_ASRMCRA, OUTFIFO_THRESH_0 | INFIFO_THRESH_1);

      reg32clrbit(ASRC_ASRMCRA, 23);   // zeroize Pair A buffers

      reg32setbit(ASRC_ASRMCRA, 21);   // stall conversion in case of near full/near empty
condition

      reg32clrbit(ASRC_ASRMCRA, 20); // Do not bypass polyA filter

      // Set ASRC Interrupt

      //CAPTURE_INTERRUPT(ASRC_INT_ROUTINE, asrc_handler);

      //enable_hdler(ASRC_INT_NUM);

      disable_hdler(ASRC_INT_NUM);

      incnt=0;

      outcnt=0;

      reg32setbit(ASRC_ASRCTR,0); // enable ASRC
```

```
      #define ASRCFG_INIA_FINISH  0x1<<21
      while ( (reg32_read(ASRC_ASRCFG) & ASRCFG_INIA_FINISH) == 0); // wait for ini finished.
      // Polling
          while (outcnt < 100) <
      {
              int ii;
              if ( (reg32_read(ASRC_ASRSTR) & ASRSTR_AIDEA_MASK) != 0 )
               {
                 for (ii=0;ii<2;ii++)</codeblock
                   {
                      reg32_write(ASRC_ASRDIA,asrc_input_array[incnt]);     // feed in input
data
                      reg32_write(ASRC_ASRDIA,asrc_input_array[incnt]);     // feed in input
data
                      incnt=(incnt+1)%128;
                   }
                }
              if ( (reg32_read(ASRC_ASRSTR) & ASRSTR_AODFA_MASK) != 0 )
               {
                   for (ii=0;ii<2;ii++)<
                     {
                       WORD TempRdOut;
                       TempRdOut=reg32_read(ASRC_ASRDOA);  // get output data
                       TempRdOut=reg32_read(ASRC_ASRDOA);  // get output data
                       outcnt=outcnt+1;
                     }
                 }
              if ( (reg32_read(ASRC_ASRSTR) & ASRSTR_AOLE_MASK) != 0 )
               {
                   reg32_write(ASRC_ASRSTR,ASRSTR_AOLE_MASK);      // clear overloading
errors
                }
             }
         reg32clrbit(ASRC_ASRCTR,0); // disable ASRC
   }
```

16.7 ASRC Memory Map/Register Definition

All useful registers are listed in the memory map below. The access of undefined registers will behave as normal registers.

All the interface registers are LSB aligned except the input FIFOs and the output FIFOs, and each register has only 24 effective bits.

The input FIFO and output FIFO word alignment can be defined using ASRMCR1{A,B,C} registers in 32-bit interface system.

ASRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
203_4000	ASRC Control Register (ASRC_ASRCTR)	32	R/W	0000_0000h	16.7.1/504
203_4004	ASRC Interrupt Enable Register (ASRC_ASRIER)	32	R/W	0000_0000h	16.7.2/507
203_400C	ASRC Channel Number Configuration Register (ASRC_ASRCNCR)	32	R/W	0000_0000h	16.7.3/508
203_4010	ASRC Filter Configuration Status Register (ASRC_ASRCFG)	32	R/W	0000_0000h	16.7.4/510
203_4014	ASRC Clock Source Register (ASRC_ASRCSR)	32	R/W	0000_0000h	16.7.5/512
203_4018	ASRC Clock Divider Register 1 (ASRC_ASRCDR1)	32	R/W	0000_0000h	16.7.6/515
203_401C	ASRC Clock Divider Register 2 (ASRC_ASRCDR2)	32	R/W	0000_0000h	16.7.7/516
203_4020	ASRC Status Register (ASRC_ASRSTR)	32	R	0000_0000h	16.7.8/518
203_4040	ASRC Parameter Register n (ASRC_ASRPMn1)	32	R/W	0000_0000h	16.7.9/521
203_4044	ASRC Parameter Register n (ASRC_ASRPMn2)	32	R/W	0000_0000h	16.7.9/521
203_4048	ASRC Parameter Register n (ASRC_ASRPMn3)	32	R/W	0000_0000h	16.7.9/521
203_404C	ASRC Parameter Register n (ASRC_ASRPMn4)	32	R/W	0000_0000h	16.7.9/521
203_4050	ASRC Parameter Register n (ASRC_ASRPMn5)	32	R/W	0000_0000h	16.7.9/521
203_4054	ASRC ASRC Task Queue FIFO Register 1 (ASRC_ASRTFR1)	32	R/W	0000_0000h	16.7.10/ 522
203_405C	ASRC Channel Counter Register (ASRC_ASRCCR)	32	R/W	0000_0000h	16.7.11/ 523
203_4060	ASRC Data Input Register for Pair x (ASRC_ASRDIA)	32	W	0000_0000h	16.7.12/ 524
203_4064	ASRC Data Output Register for Pair x (ASRC_ASRDOA)	32	R	0000_0000h	16.7.13/ 524
203_4068	ASRC Data Input Register for Pair x (ASRC_ASRDIB)	32	W	0000_0000h	16.7.12/ 524
203_406C	ASRC Data Output Register for Pair x (ASRC_ASRDOB)	32	R	0000_0000h	16.7.13/ 524

Table continues on the next page...

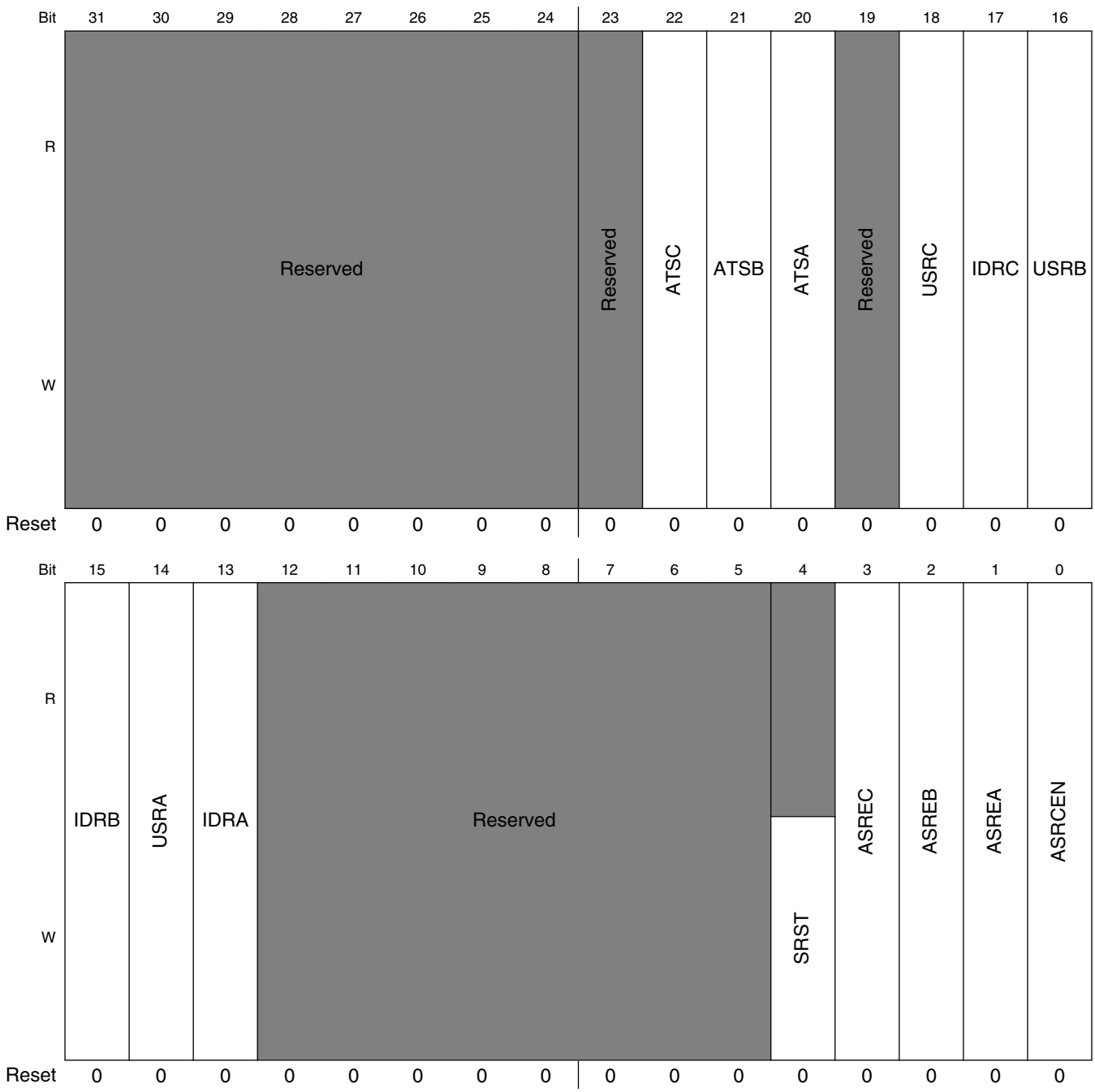
ASRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
203_4070	ASRC Data Input Register for Pair x (ASRC_ASRDIC)	32	W	0000_0000h	16.7.12/ 524
203_4074	ASRC Data Output Register for Pair x (ASRC_ASRDOC)	32	R	0000_0000h	16.7.13/ 524
203_4080	ASRC Ideal Ratio for Pair A-High Part (ASRC_ASRIDRHA)	32	R/W	0000_0000h	16.7.14/ 525
203_4084	ASRC Ideal Ratio for Pair A -Low Part (ASRC_ASRIDRLA)	32	R/W	0000_0000h	16.7.15/ 526
203_4088	ASRC Ideal Ratio for Pair B-High Part (ASRC_ASRIDRHB)	32	R/W	0000_0000h	16.7.16/ 526
203_408C	ASRC Ideal Ratio for Pair B-Low Part (ASRC_ASRIDRLB)	32	R/W	0000_0000h	16.7.17/ 527
203_4090	ASRC Ideal Ratio for Pair C-High Part (ASRC_ASRIDRHC)	32	R/W	0000_0000h	16.7.18/ 527
203_4094	ASRC Ideal Ratio for Pair C-Low Part (ASRC_ASRIDRLC)	32	R/W	0000_0000h	16.7.19/ 528
203_4098	ASRC 76 kHz Period in terms of ASRC processing clock (ASRC_ASR76K)	32	R/W	0000_0A47h	16.7.20/ 529
203_409C	ASRC 56 kHz Period in terms of ASRC processing clock (ASRC_ASR56K)	32	R/W	0000_0DF3h	16.7.21/ 530
203_40A0	ASRC Misc Control Register for Pair A (ASRC_ASRMCRA)	32	R/W	0000_0000h	16.7.22/ 531
203_40A4	ASRC FIFO Status Register for Pair A (ASRC_ASRFSTA)	32	R	0000_0000h	16.7.23/ 533
203_40A8	ASRC Misc Control Register for Pair B (ASRC_ASRMCRB)	32	R/W	0000_0000h	16.7.24/ 534
203_40AC	ASRC FIFO Status Register for Pair B (ASRC_ASRFSTB)	32	R	0000_0000h	16.7.25/ 536
203_40B0	ASRC Misc Control Register for Pair C (ASRC_ASRMCRC)	32	R/W	0000_0000h	16.7.26/ 537
203_40B4	ASRC FIFO Status Register for Pair C (ASRC_ASRFSTC)	32	R	0000_0000h	16.7.27/ 539
203_40C0	ASRC Misc Control Register 1 for Pair X (ASRC_ASRMCR1A)	32	R/W	0000_0000h	16.7.28/ 540
203_40C4	ASRC Misc Control Register 1 for Pair X (ASRC_ASRMCR1B)	32	R/W	0000_0000h	16.7.28/ 540
203_40C8	ASRC Misc Control Register 1 for Pair X (ASRC_ASRMCR1C)	32	R/W	0000_0000h	16.7.28/ 540

16.7.1 ASRC Control Register (ASRC_ASRCTR)

The ASRC control register (ASRCTR) is a 24-bit read/write register that controls the ASRC operations.

Address: 203_4000h base + 0h offset = 203_4000h



ASRC_ASRCTR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 -	This field is reserved. Reserved. Should be written as zero for compatibility.
22 ATSC	ASRC Pair C Automatic Selection For Processing Options When this bit is 1, pair C will automatic update its pre-processing and post-processing options (ASRCFG: PREMODC, ASRCFG:POSTMODC see ASRC Misc Control Register 1 for Pair C) based on the frequencies it detected. To use this option, the two parameter registers(ASR76K and ASR56K) should be set correctly (see ASRC Misc Control Register 1 for Pair C and ASRC Misc Control Register 1 for Pair C). When this bit is 0, the user is responsible for choosing the proper processing options for pair C. This bit should be disabled when {USRC, IDRC}={1,1}.
21 ATSB	ASRC Pair B Automatic Selection For Processing Options When this bit is 1, pair B will automatic update its pre-processing and post-processing options (ASRCFG: PREMODB, ASRCFG:POSTMODB see ASRC Misc Control Register 1 for Pair C) based on the frequencies it detected. To use this option, the two parameter registers(ASR76K and ASR56K) should be set correctly (see ASRC Misc Control Register 1 for Pair C and ASRC Misc Control Register 1 for Pair C). When this bit is 0, the user is responsible for choosing the proper processing options for pair B. This bit should be disabled when {USRB, IDRB}={1,1}.
20 ATSA	ASRC Pair A Automatic Selection For Processing Options When this bit is 1, pair A will automatic update its pre-processing and post-processing options (ASRCFG: PREMODA, ASRCFG:POSTMODA see ASRC Misc Control Register 1 for Pair C) based on the frequencies it detected. To use this option, the two parameter registers(ASR76K and ASR56K) should be set correctly (see ASRC Misc Control Register 1 for Pair C and ASRC Misc Control Register 1 for Pair C). When this bit is 0, the user is responsible for choosing the proper processing options for pair A. This bit should be disabled when {USRA, IDRA}={1,1}.
19 -	This field is reserved. Reserved. Should be written as zero for compatibility.
18 USRC	Use Ratio for Pair C Use ratio as the input to ASRC. This bit is used in conjunction with IDRC control bit.
17 IDRC	Use Ideal Ratio for Pair C When USRC=0, this bit has no usage. When USRC=1 and IDRC=0, ASRC internal measured ratio will be used. When USRC=1 and IDRC=1, the idea ratio from the interface register ASRIDRHC, ASRIDRLC will be used. It is suggested to manually set ASRCFG:POSTMODC, ASRCFG:PREMODC according to Table 16-7 in this case.
16 USRB	Use Ratio for Pair B Use ratio as the input to ASRC. This bit is used in conjunction with IDRB control bit.
15 IDRB	Use Ideal Ratio for Pair B When USRB=0, this bit has no usage. When USRB=1 and IDRB=0, ASRC internal measured ratio will be used.

Table continues on the next page...

ASRC_ASRCTR field descriptions (continued)

Field	Description
	When USRB=1 and IDRB=1, the idea ratio from the interface register ASRIDRHB, ASRIDRLB will be used.It is suggested to manually set ASRCFG:POSTMODB, ASRCFG:PREMODB according to Table 16-7 in this case.
14 USRA	Use Ratio for Pair A Use ratio as the input to ASRC. This bit is used in conjunction with IDRA control bit.
13 IDRA	Use Ideal Ratio for Pair A When USRA=0, this bit has no usage. When USRA=1 and IDRA=0, ASRC internal measured ratio will be used. When USRA=1 and IDRA=1, the idea ratio from the interface register ASRIDRHA, ASRIDRLA will be used. It is suggested to manually set ASRCFG:POSTMODA, ASRCFG:PREMODA according to Table 16-7 in this case.
12–5 -	This field is reserved. Reserved. Should be written as zero for compatibility.
4 SRST	Software Reset This bit is self-clear bit. Once it is been written as 1, it will generate a software reset signal inside ASRC. After 9 cycles of the ASRC processing clock, this reset process will stop, and this bit will be cleared automatically.
3 ASREC	ASRC Enable C Enable the operation of the conversion C of ASRC. When ASREC is cleared, operation of conversion C is disabled.
2 ASREB	ASRC Enable B Enable the operation of the conversion B of ASRC. When ASREB is cleared, operation of conversion B is disabled.
1 ASREA	ASRC Enable A Enable the operation of the conversion A of ASRC. When ASREA is cleared, operation of conversion A is disabled.
0 ASRCEN	ASRC Enable Enable the operation of ASRC.

16.7.2 ASRC Interrupt Enable Register (ASRC_ASRIER)

Address: 203_4000h base + 4h offset = 203_4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								AFPWE	AOLIE	ADOEC	ADOEB	ADOEA	ADIEC	ADIEB	ADIEA
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ASRC_ASRIER field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–8 -	This field is reserved. Reserved. Should be written as zero for compatibility.
7 AFPWE	FP in Wait State Interrupt Enable Enables the FP in wait state interrupt. 1 interrupt enabled 0 interrupt disabled
6 AOLIE	Overload Interrupt Enable Enables the overload interrupt. 1 interrupt enabled 0 interrupt disabled
5 ADOEC	Data Output C Interrupt Enable Enables the data output C interrupt. 1 interrupt enabled 0 interrupt disabled
4 ADOEB	Data Output B Interrupt Enable Enables the data output B interrupt.

Table continues on the next page...

ASRC_ASRIER field descriptions (continued)

Field	Description
	1 interrupt enabled 0 interrupt disabled
3 ADOEA	Data Output A Interrupt Enable Enables the data output A interrupt. 1 interrupt enabled 0 interrupt disabled
2 ADIEC	Data Input C Interrupt Enable Enables the data input C interrupt. 1 interrupt enabled 0 interrupt disabled
1 ADIEB	Data Input B Interrupt Enable Enables the data input B interrupt. 1 interrupt enabled 0 interrupt disabled
0 ADIEA	Data Input A Interrupt Enable Enables the data input A Interrupt. 1 interrupt enabled 0 interrupt disabled

16.7.3 ASRC Channel Number Configuration Register (ASRC_ASRCNCR)

The ASRC channel number configuration register (ASRCNCR) is a 24-bit read/write register that sets the number of channels used by each ASRC conversion pair.

There are 10 channels available for distribution among 3 conversion pairs, they are ordered as 0,1,...,9. The bottom [0, ANCA-1] channels are used for pair A, the top [10-ANCC, 9] channels are used for pair C, and the [ANCA, ANCA+ANCB-1] channels are allocated for pair B. In case that ANCA=0, then the [0, ANCB-1] channels are assigned for pair B.

Address: 203_4000h base + Ch offset = 203_400Ch

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

ASRC_ASRCNCR field descriptions

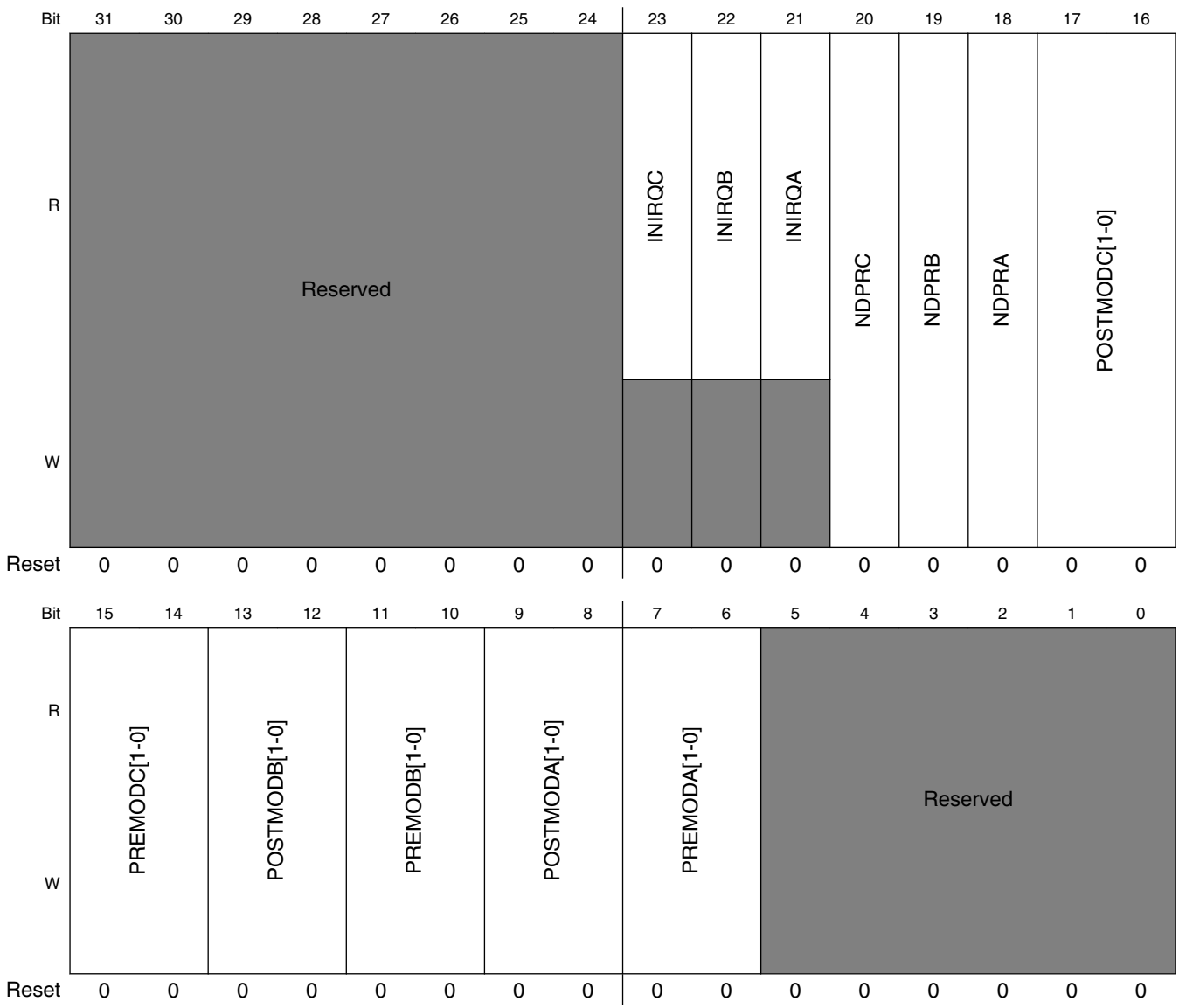
Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–12 -	This field is reserved. Reserved. Should be written as zero for compatibility.
11–8 ANCC	Number of C Channels[1] 0000 0 channels in C (Pair C is disabled) 0001 1 channel in C 0010 2 channels in C 0011 3 channels in C 0100 4 channels in C 0101 5 channels in C 0110 6 channels in C 0111 7 channels in C 1000 8 channels in C 1001 9 channels in C 1010 10 channels in C 1011-1111 Should not be used.
7–4 ANCB	Number of B Channels 0000 0 channels in B (Pair B is disabled) 0001 1 channel in B 0010 2 channels in B 0011 3 channels in B 0100 4 channels in B 0101 5 channels in B 0110 6 channels in B 0111 7 channels in B 1000 8 channels in B 1001 9 channels in B 1010 10 channels in B 1011-1111 Should not be used.
ANCA	Number of A Channels 0000 0 channels in A (Pair A is disabled) 0001 1 channel in A 0010 2 channels in A 0011 3 channels in A 0100 4 channels in A 0101 5 channels in A 0110 6 channels in A 0111 7 channels in A 1000 8 channels in A 1001 9 channels in A 1010 10 channels in A 1011-1111 Should not be used.

1. ANCC+ANCB+ANCA<=10. Hardware is not checking the constraint. Programmer should take the responsibility to ensure the constraint is satisfied.

16.7.4 ASRC Filter Configuration Status Register (ASRC_ASRCFG)

The ASRC configuration status register (ASRCFG) is a 24-bit read/write register that sets and/or automatically senses the ASRC operations.

Address: 203_4000h base + 10h offset = 203_4010h



ASRC_ASRCFG field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 INIRQC	Initialization for Conversion Pair C is served When this bit is 1, it means the initialization for conversion pair C is served. This bit is cleared by disabling the ASRC conversion pair (ASRCTR:ASREC=0 or ASRCTR:ASRCEN=0).
22 INIRQB	Initialization for Conversion Pair B is served When this bit is 1, it means the initialization for conversion pair B is served. This bit is cleared by disabling the ASRC conversion pair (ASRCTR:ASREB=0 or ASRCTR:ASRCEN=0).
21 INIRQA	Initialization for Conversion Pair A is served When this bit is 1, it means the initialization for conversion pair A is served. This bit is cleared by disabling the ASRC conversion pair (ASRCTR:ASREA=0 or ASRCTR:ASRCEN=0).
20 NDPRC	Not Use Default Parameters for RAM-stored Parameters For Conversion Pair C 0 Use default parameters for RAM-stored parameters. Override any parameters already in RAM. 1 Don't use default parameters for RAM-stored parameters. Use the parameters already stored in RAM.
19 NDPRB	Not Use Default Parameters for RAM-stored Parameters For Conversion Pair B 0 Use default parameters for RAM-stored parameters. Override any parameters already in RAM. 1 Don't use default parameters for RAM-stored parameter. Use the parameters already stored in RAM.
18 NDPRA	Not Use Default Parameters for RAM-stored Parameters For Conversion Pair A 0 Use default parameters for RAM-stored parameters. Override any parameters already in RAM. 1 Don't use default parameters for RAM-stored parameters. Use the parameters already stored in RAM.
17–16 POSTMODC[1-0]	Post-Processing Configuration for Conversion Pair C These bits will be read/write by user if ASRCTR:ATSC=0, and can also be automatically updated by the ASRC internal logic if ASRCTR:ATSC=1 (see ASRC Misc Control Register 1 for Pair C). These bits set the selection of the post-processing configuration. 00 Select Upsampling-by-2 as defined in Signal Processing Flow. 01 Select Direct-Connection as defined in Signal Processing Flow. 10 Select Downsampling-by-2 as defined in Signal Processing Flow.
15–14 PREMODC[1-0]	Pre-Processing Configuration for Conversion Pair C These bits will be read/write by user if ASRCTR:ATSC=0, and can also be automatically updated by the ASRC internal logic if ASRCTR:ATSC=1 (see ASRC Misc Control Register 1 for Pair C). These bits set the selection of the pre-processing configuration. 00 Select Upsampling-by-2 as defined in Signal processing flow 01 Select Direct-Connection as defined in Signal processing flow 10 Select Downsampling-by-2 as defined in Signal processing flow 11 Select passthrough mode. In this case, POSTMODC[1-0] have no use.
13–12 POSTMODB[1-0]	Post-Processing Configuration for Conversion Pair B These bits will be read/write by user if ASRCTR:ATSB=0, and can also be automatically updated by the ASRC internal logic if ASRCTR:ATSB=1 (see ASRC Misc Control Register 1 for Pair C). These bits set the selection of the post-processing configuration.

Table continues on the next page...

ASRC_ASRCFG field descriptions (continued)

Field	Description
	00 Select Upsampling-by-2 as defined in Signal processing flow 01 Select Direct-Connection as defined in Signal processing flow 10 Select Downsampling-by-2 as defined in Signal processing flow
11–10 PREMODB[1-0]	Pre-Processing Configuration for Conversion Pair B These bits will be read/write by user if ASRCTR:ATSB=0, and can also be automatically updated by the ASRC internal logic if ASRCTR:ATSB=1 (see ASRC Misc Control Register 1 for Pair C). These bits set the selection of the pre-processing configuration. 00 Select Upsampling-by-2 as defined in Signal processing flow 01 Select Direct-Connection as defined in Signal processing flow 10 Select Downsampling-by-2 as defined in Signal processing flow 11 Select passthrough mode. In this case, POSTMODB[1-0] have no use.
9–8 POSTMODA[1-0]	Post-Processing Configuration for Conversion Pair A These bits will be read/write by user if ASRCTR:ATSA=0, and can also be automatically updated by the ASRC internal logic if ASRCTR:ATSA=1 (see ASRC Misc Control Register 1 for Pair C). These bits set the selection of the post-processing configuration. 00 Select Upsampling-by-2 as defined in Signal processing flow 01 Select Direct-Connection as defined in Signal processing flow 10 Select Downsampling-by-2 as defined in Signal processing flow
7–6 PREMODA[1-0]	Pre-Processing Configuration for Conversion Pair A These bits will be read/write by user if ASRCTR:ATSA=0, and can also be automatically updated by the ASRC internal logic if ASRCTR:ATSA=1 (see ASRC Misc Control Register 1 for Pair C). These bits set the selection of the pre-processing configuration. 00 Select Upsampling-by-2 as defined in Signal processing flow 01 Select Direct-Connection as defined in Signal processing flow 10 Select Downsampling-by-2 as defined in Signal processing flow 11 Select passthrough mode. In this case, POSTMODA[1-0] have no use.
-	This field is reserved. Reserved. Should be written as zero for compatibility.

16.7.5 ASRC Clock Source Register (ASRC_ASRCSR)

The ASRC clock source register (ASRCSR) is a 24-bit read/write register that controls the sources of the input and output clocks of the ASRC.

The clock connections are shown in Figure 1 :

Address: 203_4000h base + 14h offset = 203_4014h

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

ASRC_ASRCSR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–20 AOCSC	**Output Clock Source C** 0000 bit clock 0 0001 bit clock 1 0010 bit clock 2 0011 bit clock 3 0100 bit clock 4 0101 bit clock 5 0110 bit clock 6 0111 bit clock 7 1000 bit clock 8 1001 bit clock 9 1010 bit clock A 1011 bit clock B 1100 bit clock C 1101 bit clock D 1110 bit clock E 1111 clock disabled, connected to zero any other value bit clock 0
19–16 AOCSB	**Output Clock Source B** 0000 bit clock 0 0001 bit clock 1 0010 bit clock 2 0011 bit clock 3 0100 bit clock 4 0101 bit clock 5 0110 bit clock 6 0111 bit clock 7 1000 bit clock 8 1001 bit clock 9 1010 bit clock A 1011 bit clock B 1100 bit clock C 1101 bit clock D 1110 bit clock E 1111 clock disabled, connected to zero any other value bit clock 0
15–12 AOCSA	**Output Clock Source A** 0000 bit clock 0 0001 bit clock 1 0010 bit clock 2 0011 bit clock 3 0100 bit clock 4

Table continues on the next page...

ASRC_ASRCSR field descriptions (continued)

Field	Description
	0101 bit clock 5 0110 bit clock 6 0111 bit clock 7 1000 bit clock 8 1001 bit clock 9 1010 bit clock A 1011 bit clock B 1100 bit clock C 1101 bit clock D 1110 bit clock E 1111 clock disabled, connected to zero any other value bit clock 0
11–8 AICSC	**Input Clock Source C** 0000 bit clock 0 0001 bit clock 1 0010 bit clock 2 0011 bit clock 3 0100 bit clock 4 0101 bit clock 5 0110 bit clock 6 0111 bit clock 7 1000 bit clock 8 1001 bit clock 9 1010 bit clock A 1011 bit clock B 1100 bit clock C 1101 bit clock D 1110 bit clock E 1111 clock disabled, connected to zero any other value bit clock 0
7–4 AICSB	**Input Clock Source B** 0000 bit clock 0 0001 bit clock 1 0010 bit clock 2 0011 bit clock 3 0100 bit clock 4 0101 bit clock 5 0110 bit clock 6 0111 bit clock 7 1000 bit clock 8 1001 bit clock 9 1010 bit clock A 1011 bit clock B 1100 bit clock C 1101 bit clock D

Table continues on the next page...

ASRC_ASRCSR field descriptions (continued)

Field	Description
	1110 bit clock E 1111 clock disabled, connected to zero any other value bit clock 0
AICSA	**Input Clock Source A** 0000 bit clock 0 0001 bit clock 1 0010 bit clock 2 0011 bit clock 3 0100 bit clock 4 0101 bit clock 5 0110 bit clock 6 0111 bit clock 7 1000 bit clock 8 1001 bit clock 9 1010 bit clock A 1011 bit clock B 1100 bit clock C 1101 bit clock D 1110 bit clock E 1111 clock disabled, connected to zero any other value bit clock 0

16.7.6 ASRC Clock Divider Register 1 (ASRC_ASRCDR1)

The ASRC clock divider register (ASRCDR1) is a 24-bit read/write register that controls the division factors of the ASRC input and output clock sources.

Address: 203_4000h base + 18h offset = 203_4018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								AOCDB			AOCPB			AOCDA		AOCPA				AICDB				AICPB				AICDA				AICPA			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ASRC_ASRCDR1 field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–21 AOCDB	Output Clock Divider B Specify the divide ratio of the output clock divider B. The divide ratio may range from 1 to 8 (AOCDB[2:0] = 000 to 111).

Table continues on the next page...

ASRC_ASRCDR1 field descriptions (continued)

Field	Description
20–18 AOCPB	Output Clock Prescaler B Specify the prescaling factor of the output prescaler B. The prescaling ratio may be any power of 2 from 1 to 128.
17–15 AOCDA	Output Clock Divider A Specify the divide ratio of the output clock divider A. The divide ratio may range from 1 to 8 (AOCDA[2:0] = 000 to 111).
14–12 AOCPA	Output Clock Prescaler A Specify the prescaling factor of the output prescaler A. The prescaling ratio may be any power of 2 from 1 to 128.
11–9 AICDB	Input Clock Divider B Specify the divide ratio of the input clock divider B. The divide ratio may range from 1 to 8 (AICDB[2:0] = 000 to 111).
8–6 AICPB	Input Clock Prescaler B Specify the prescaling factor of the input prescaler B. The prescaling ratio may be any power of 2 from 1 to 128.
5–3 AICDA	Input Clock Divider A Specify the divide ratio of the input clock divider A. The divide ratio may range from 1 to 8 (AICDA[2:0] = 000 to 111).
AICPA	Input Clock Prescaler A Specify the prescaling factor of the input prescaler A. The prescaling ratio may be any power of 2 from 1 to 128.

16.7.7 ASRC Clock Divider Register 2 (ASRC_ASRCDR2)

The ASRC clock divider register (ASRCDR2) is a 24-bit read/write register that controls the division factors of the ASRC input and output clock sources.

Address: 203_4000h base + 1Ch offset = 203_401Ch

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

ASRC_ASRCDR2 field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–12 -	This field is reserved. Reserved. Should be written as zero for compatibility.

Table continues on the next page...

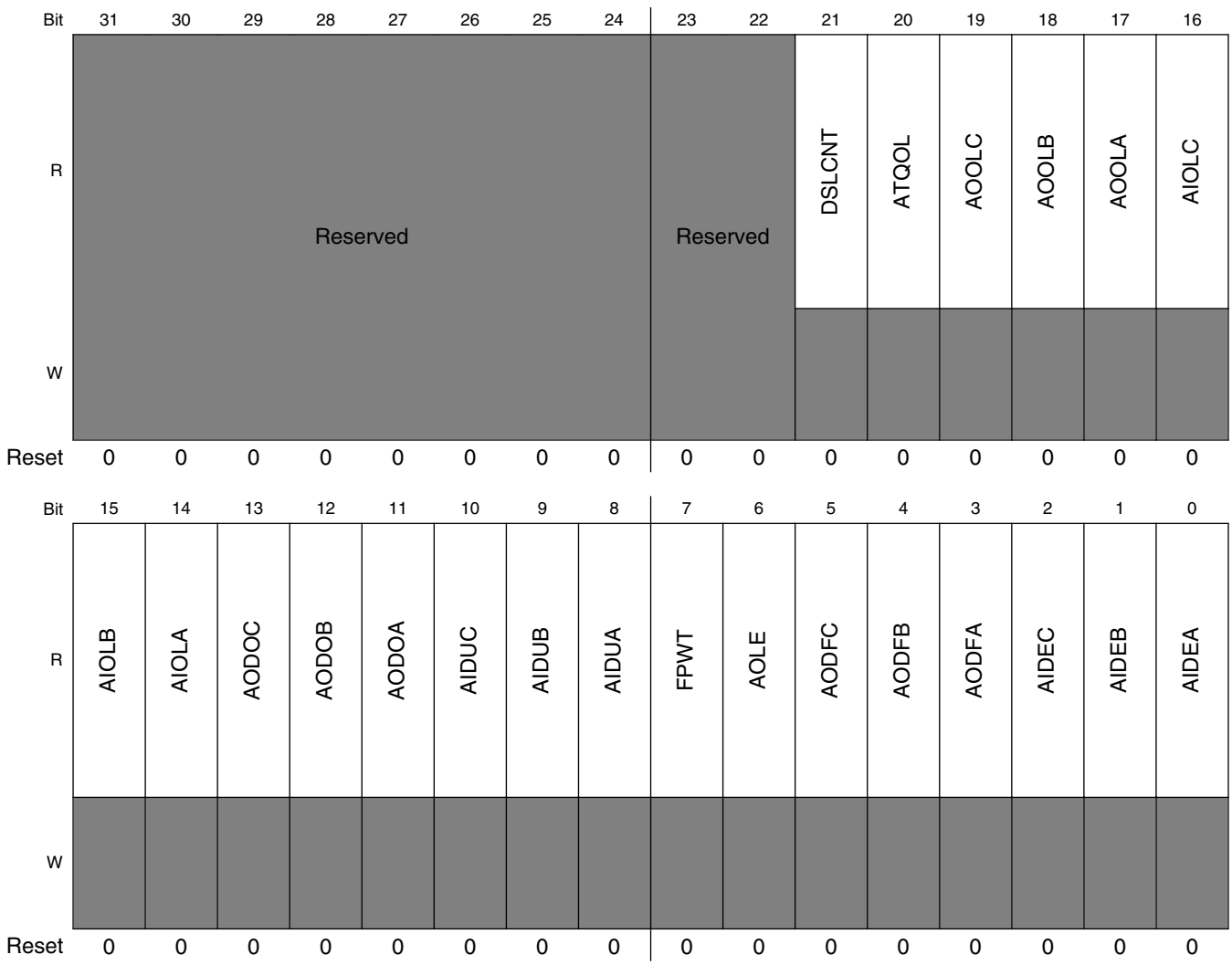
ASRC_ASRCDR2 field descriptions (continued)

Field	Description
11–9 AOCDC	Output Clock Divider C Specify the divide ratio of the output clock divider C. The divide ratio may range from 1 to 8 (AOCDC[2:0] = 000 to 111).
8–6 AOCPC	Output Clock Prescaler C Specify the prescaling factor of the output prescaler C. The prescaling ratio may be any power of 2 from 1 to 128.
5–3 AICDC	Input Clock Divider C Specify the divide ratio of the input clock divider C. The divide ratio may range from 1 to 8 (AICDC[2:0] = 000 to 111).
AICPC	Input Clock Prescaler C Specify the prescaling factor of the input prescaler C. The prescaling ratio may be any power of 2 from 1 to 128.

16.7.8 ASRC Status Register (ASRC_ASRSTR)

The ASRC status register (ASRSTR) is a 24-bit read-write register used by the processor core to examine the status of the ASRC block and clear the overload interrupt request and AOLE flag bit. Read the status register will return the current state of ASRC.

Address: 203_4000h base + 20h offset = 203_4020h



ASRC_ASRSTR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.

Table continues on the next page...

ASRC_ASRSTR field descriptions (continued)

Field	Description
23–22 -	This field is reserved. Reserved. Should be written as zero for compatibility.
21 DSLCNT	DSL Counter Input to FIFO ready When set, this bit indicates that new DSL counter information is stored in the internal ASRC FIFO. When clear, this bit indicates that new DSL counter information is in the process of storage into the internal ASRC FIFO. When ASRIER:AFPWE=1, the rising edge of this signal will propose an interrupt request. Writing any value with this bit set will clear the interrupt request proposed by the rising edge of this bit.
20 ATQOL	Task Queue FIFO overload When set, this bit indicates that task queue FIFO logic is oveloaded. This may help to check the reason why overload interrupt happens. The bit is cleared when writing ASRSTR:AOLE as 1.
19 AOOLC	Pair C Output Task Overload When set, this bit indicates that pair C output task is oveloaded. This may help to check the reason why overload interrupt happens. The bit is cleared when writing ASRSTR:AOLE as 1.
18 AOOLB	Pair B Output Task Overload When set, this bit indicates that pair B output task is oveloaded. This may help to check the reason why overload interrupt happens. The bit is cleared when writing ASRSTR:AOLE as 1.
17 AOOLA	Pair A Output Task Overload When set, this bit indicates that pair A output task is oveloaded. This may help to check the reason why overload interrupt happens. The bit is cleared when writing ASRSTR:AOLE as 1.
16 AIOLC	Pair C Input Task Overload When set, this bit indicates that pair C input task is oveloaded. This may help to check the reason why overload interrupt happens. The bit is cleared when writing ASRSTR:AOLE as 1.
15 AIOLB	Pair B Input Task Overload When set, this bit indicates that pair B input task is oveloaded. This may help to check the reason why overload interrupt happens. The bit is cleared when writing ASRSTR:AOLE as 1.
14 AIOLA	Pair A Input Task Overload When set, this bit indicates that pair A input task is oveloaded. This may help to check the reason why overload interrupt happens. The bit is cleared when writing ASRSTR:AOLE as 1.
13 AODOC	Output Data Buffer C has overflowed When set, this bit indicates that output data buffer C has overflowed. When clear, this bit indicates that output data buffer C has not overflowed The bit is cleared when writing ASRSTR:AOLE as 1.
12 AODOB	Output Data Buffer B has overflowed

Table continues on the next page...

ASRC_ASRSTR field descriptions (continued)

Field	Description
	When set, this bit indicates that output data buffer B has overflowed. When clear, this bit indicates that output data buffer B has not overflowed The bit is cleared when writing ASRSTR:AOLE as 1.
11 AODOA	Output Data Buffer A has overflowed When set, this bit indicates that output data buffer A has overflowed. When clear, this bit indicates that output data buffer A has not overflowed The bit is cleared when writing ASRSTR:AOLE as 1.
10 AIDUC	Input Data Buffer C has underflowed When set, this bit indicates that input data buffer C has underflowed. When clear, this bit indicates that input data buffer C has not underflowed. The bit is cleared when writing ASRSTR:AOLE as 1.
9 AIDUB	Input Data Buffer B has underflowed When set, this bit indicates that input data buffer B has underflowed. When clear, this bit indicates that input data buffer B has not underflowed. The bit is cleared when writing ASRSTR:AOLE as 1.
8 AIDUA	Input Data Buffer A has underflowed When set, this bit indicates that input data buffer A has underflowed. When clear, this bit indicates that input data buffer A has not underflowed. The bit is cleared when writing ASRSTR:AOLE as 1.
7 FPWT	FP is in wait states This bit is for debug only. When set, this bit indicates that ASRC is in wait states. When clear, this bit indicates that ASRC is not in wait states.
6 AOLE	Overload Error Flag When set, this bit indicates that the task rate is too high for the ASRC to handle. The reasons for overload may be: - too high input clock frequency, - too high output clock frequency, - incorrect selection of the pre-filter, - low ASRC processing clock, - too many channels, - underrun, - or any combination of the reasons above. Since the ASRC uses the same hardware resources to perform various tasks, the real reason for the overload is not straight forward, and it should be carefully analyzed by the programmer. If ASRIER:AOLIE=1, an interrupt will be proposed when this bit is set. Write any value with this bit set as one into the status register will clear this bit and the interrupt request proposed by this bit.
5 AODFC	Number of data in Output Data Buffer C is greater than threshold

Table continues on the next page...

ASRC_ASRSTR field descriptions (continued)

Field	Description
	When set, this bit indicates that number of data already existing in ASRDORC is greater than threshold and the processor can read data from ASRDORC. When AODFC is set, the ASRC generates data output C interrupt request to the processor, if enabled (that is, ASRIER:ADOEC = 1). A DMA request is always generated when the AODFC bit is set, but a DMA transfer takes place only if a DMA channel is active and triggered by this event.
4 AODFB	Number of data in Output Data Buffer B is greater than threshold When set, this bit indicates that number of data already existing in ASRDORB is greater than threshold and the processor can read data from ASRDORB. When AODFB is set, the ASRC generates data output B interrupt request to the processor, if enabled (that is, ASRIER:ADOEB = 1). A DMA request is always generated when the AODFB bit is set, but a DMA transfer takes place only if a DMA channel is active and triggered by this event.
3 AODFA	Number of data in Output Data Buffer A is greater than threshold When set, this bit indicates that number of data already existing in ASRDORA is greater than threshold and the processor can read data from ASRDORA. When AODFA is set, the ASRC generates data output A interrupt request to the processor, if enabled (that is, ASRIER:ADOEA = 1). A DMA request is always generated when the AODFA bit is set, but a DMA transfer takes place only if a DMA channel is active and triggered by this event.
2 AIDEC	Number of data in Input Data Buffer C is less than threshold When set, this bit indicates that number of data still available in ASRDIRC is less than threshold and the processor can write data to ASRDIRC. When AIDEC is set, the ASRC generates data input C interrupt request to the processor, if enabled (that is, ASRIER:ADIEC = 1). A DMA request is always generated when the AIDEC bit is set, but a DMA transfer takes place only if a DMA channel is active and triggered by this event.
1 AIDEB	Number of data in Input Data Buffer B is less than threshold When set, this bit indicates that number of data still available in ASRDIRB is less than threshold and the processor can write data to ASRDIRB. When AIDEB is set, the ASRC generates data input B interrupt request to the processor, if enabled (that is, ASRIER:ADIEB = 1). A DMA request is always generated when the AIDEB bit is set, but a DMA transfer takes place only if a DMA channel is active and triggered by this event.
0 AIDEA	Number of data in Input Data Buffer A is less than threshold When set, this bit indicates that number of data still available in ASRDIRA is less than threshold and the processor can write data to ASRDIRA. When AIDEA is set, the ASRC generates data input A interrupt request to the processor, if enabled (that is, ASRIER:ADIEA = 1). A DMA request is always generated when the AIDEA bit is set, but a DMA transfer takes place only if a DMA channel is active and triggered by this event.

16.7.9 ASRC Parameter Register n (ASRC_ASRPMn*n*)

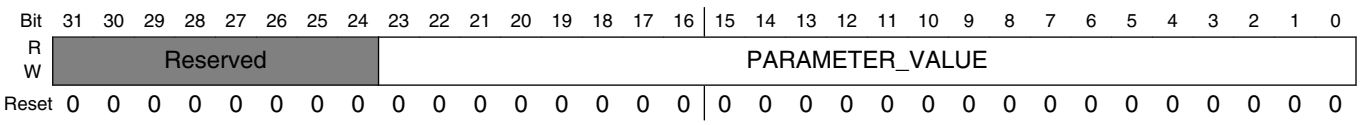
Parameter registers determine the performance of ASRC.

The parameter registers must be initialized by software before ASRC is enabled. Recommended values are given in ASRC Misc Control Register 1 for Pair C below,

Table 16-8. ASRC Parameter Registers (ASRPM1~ASRPM5)

Register	Offset	Access	Reset Value	Recommend Value
asrcpm1	0x40	R/W	0x00_0000	0x7fffff
asrcpm2	0x44	R/W	0x00_0000	0x255555
asrcpm3	0x48	R/W	0x00_0000	0xff7280
asrcpm4	0x4C	R/W	0x00_0000	0xff7280
asrcpm5	0x50	R/W	0x00_0000	0xff7280

Address: 203_4000h base + 40h offset + (4d × i), where i=0d to 4d



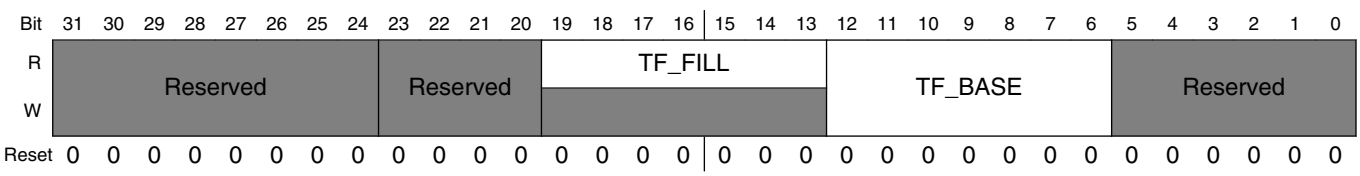
ASRC_ASRPMn*n* field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
PARAMETER_ VALUE	See recommended values table.

16.7.10 ASRC ASRC Task Queue FIFO Register 1 (ASRC_ASRTFR1)

The register defines and shows the parameters for ASRC inner task queue FIFOs.

Address: 203_4000h base + 54h offset = 203_4054h



ASRC_ASRTFR1 field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–20 -	This field is reserved. Reserved. Should be written as zero for compatibility.

Table continues on the next page...

ASRC_ASRTFR1 field descriptions (continued)

Field	Description
19–13 TF_FILL	Current number of entries in task queue FIFO.
12–6 TF_BASE	Base address for task queue FIFO. Set to 0x7C.
-	This field is reserved. Reserved. Should be written as zero for compatibility.

16.7.11 ASRC Channel Counter Register (ASRC_ASRCCR)

The ASRC channel counter register (ASRCCR) is a 24-bit read/write register that sets and reflects the current specific input/output FIFO being accessed through shared peripheral bus for each ASRC conversion pair.

Address: 203_4000h base + 5Ch offset = 203_405Ch

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

ASRC_ASRCCR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–20 ACOC	The channel counter for Pair C's output FIFO These bits stand for the current channel being accessed through shared peripheral bus for Pair C's output FIFO's usage. The value can be any value between [0, ANCC-1]
19–16 ACOB	The channel counter for Pair B's output FIFO These bits stand for the current channel being accessed through shared peripheral bus for Pair B's output FIFO's usage. The value can be any value between [0, ANCB-1]
15–12 ACOA	The channel counter for Pair A's output FIFO These bits stand for the current channel being accessed through shared peripheral bus for Pair A's output FIFO's usage. The value can be any value between [0, ANCA-1]
11–8 ACIC	The channel counter for Pair C's input FIFO These bits stand for the current channel being accessed through shared peripheral bus for Pair C's input FIFO's usage. The value can be any value between [0, ANCC-1]
7–4 ACIB	The channel counter for Pair B's input FIFO These bits stand for the current channel being accessed through shared peripheral bus for Pair B's input FIFO's usage. The value can be any value between [0, ANCB-1]
ACIA	The channel counter for Pair A's input FIFO

Table continues on the next page...

ASRC_ASRCCR field descriptions (continued)

Field	Description
	These bits stand for the current channel being accessed through shared peripheral bus for Pair A's input FIFO's usage. The value can be any value between [0, ANCA-1]

16.7.12 ASRC Data Input Register for Pair *x* (ASRC_ASRDI*n*)

These registers are the interface registers for the audio data input of pair A,B,C respectively. They are backed by FIFOs.

Address: 203_4000h base + 60h offset + (8d × i), where i=0d to 2d

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

ASRC_ASRDI*n* field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
DATA	Audio data input

16.7.13 ASRC Data Output Register for Pair *x* (ASRC_ASRDO*n*)

These registers are the interface registers for the audio data output of pair A,B,C respectively. They are backed by FIFOs.

Address: 203_4000h base + 64h offset + (8d × i), where i=0d to 2d

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

ASRC_ASRDO*n* field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
DATA	Audio data output

16.7.14 ASRC Ideal Ratio for Pair A-High Part (ASRC_ASRIDRHA)

The ideal ratio registers (ASRIDRHA, ASRIDRLA) hold the ratio value IDRATIOA. IDRATIOA = $Fs_{inA}/Fs_{outA} = Ts_{outA}/Ts_{inA}$ is a 32-bit fixed point value with 26 fractional bits. This value is only useful when ASRCTR:{USRA, IDRA}=2'b11.

Address: 203_4000h base + 80h offset = 203_4080h

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

ASRC_ASRIDRHA field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–8 -	This field is reserved. Reserved
IDRATIOA[31:24]	IDRATIOA[31:24]. High part of ideal ratio value for pair A

16.7.15 ASRC Ideal Ratio for Pair A -Low Part (ASRC_ASRIDRLA)

The ideal ratio registers (ASRIDRHA, ASRIDRLA) hold the ratio value IDRATIOA. IDRATIOA = $Fs_{inA}/Fs_{outA} = Ts_{outA}/Ts_{inA}$ is a 32-bit fixed point value with 26 fractional bits. This value is only useful when ASRCTR:{USRA, IDRA}=2'b11.

Address: 203_4000h base + 84h offset = 203_4084h

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

ASRC_ASRIDRLA field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
IDRATIOA[23:0]	IDRATIOA[23:0]. Low part of ideal ratio value for pair A

16.7.16 ASRC Ideal Ratio for Pair B-High Part (ASRC_ASRIDRHB)

The ideal ratio registers (ASRIDRHB, ASRIDRLB) hold the ratio value IDRATIOB. IDRATIOB = $Fs_{inB}/Fs_{outB} = Ts_{outB}/Ts_{inB}$ is a 32-bit fixed point value with 26 fractional bits. This value is only useful when ASRCTR:{USRB, IDRB}=2'b11.

Address: 203_4000h base + 88h offset = 203_4088h

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

ASRC_ASRIDRHB field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.

Table continues on the next page...

ASRC_ASRIDRHB field descriptions (continued)

Field	Description
23–8 -	This field is reserved. Reserved
IDRATIOB[31:24]	IDRATIOB[31:24]. High part of ideal ratio value for pair B.

16.7.17 ASRC Ideal Ratio for Pair B-Low Part (ASRC_ASRIDRLB)

The ideal ratio registers (ASRIDRHB, ASRIDRLB) hold the ratio value IDRATIOB. IDRATIOB = $Fs_{inB}/Fs_{outB} = Ts_{outB}/Ts_{inB}$ is a 32-bit fixed point value with 26 fractional bits. This value is only useful when ASRCTR:{USRB, IDRB}=2'b11.

Address: 203_4000h base + 8Ch offset = 203_408Ch

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

ASRC_ASRIDRLB field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
IDRATIOB[23:0]	IDRATIOB[23:0]. Low part of ideal ratio value for pair B.

16.7.18 ASRC Ideal Ratio for Pair C-High Part (ASRC_ASRIDRHC)

The ideal ratio registers (ASRIDRHC, ASRIDRLC) hold the ratio value IDRATIOC. IDRATIOC = $Fs_{inC}/Fs_{outC} = Ts_{outC}/Ts_{inC}$ is a 32-bit fixed point value with 26 fractional bits. This value is only useful when ASRCTR:{USRC, IDRC}=2'b11.

Address: 203_4000h base + 90h offset = 203_4090h

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

ASRC_ASRIDRHC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–8 -	This field is reserved. Reserved
IDRATIOC[31:24]	IDRATIOC[31:24]. High part of ideal ratio value for pair C.

16.7.19 ASRC Ideal Ratio for Pair C-Low Part (ASRC_ASRIDRLC)

The ideal ratio registers (ASRIDRHC, ASRIDRLC) hold the ratio value IDRATIOC. IDRATIOC = $Fs_{inC}/Fs_{outC} = Ts_{outC}/Ts_{inC}$ is a 32-bit fixed point value with 26 fractional bits. This value is only useful when ASRCTR:{USRC, IDRC}=2'b11.

Address: 203_4000h base + 94h offset = 203_4094h

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

ASRC_ASRIDRLC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
IDRATIOC[23:0]	IDRATIOC[23:0]. Low part of ideal ratio value for pair C.

16.7.20 ASRC 76 kHz Period in terms of ASRC processing clock (ASRC_ASR76K)

The register (ASR76K) holds the period of the 76 kHz sampling clock in terms of the ASRC processing clock with frequency Fs_{ASRC}. ASR76K = Fs_{ASRC}/Fs_{76k}. Reset value is 0x0A47 which assumes that Fs_{ASRC}=200 MHz. This register is used to help the ASRC internal logic to decide the pre-processing and the post-processing options automatically (see ASRC Misc Control Register 1 for Pair C and ASRC Misc Control Register 1 for Pair C). In a system when Fs_{ASRC} = 133 MHz, the value should be assigned explicitly as 0x06D6 in user application code.

Address: 203_4000h base + 98h offset = 203_4098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								Reserved								ASR76K															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	1	1

ASRC_ASR76K field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–17 -	This field is reserved. Reserved
ASR76K	Value for the period of the 76 kHz sampling clock.

16.7.21 ASRC 56 kHz Period in terms of ASRC processing clock (ASRC_ASR56K)

The register (ASR56K) holds the period of the 56 kHz sampling clock in terms of the ASRC processing clock with frequency Fs_{ASRC}. ASR56K = Fs_{ASRC}/Fs_{56k}. Reset value is 0x0DF3 which assumes that Fs_{ASRC} = 200 MHz. This register is used to help the ASRC internal logic to decide the pre-processing and the post-processing options automatically (see ASRC Misc Control Register 1 for Pair C and ASRC Misc Control Register 1 for Pair C). In a system when Fs_{ASRC} = 133 MHz, the value should be assigned explicitly as 0x0947 in user application code.

Address: 203_4000h base + 9Ch offset = 203_409Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								Reserved								ASR56K															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	1	1

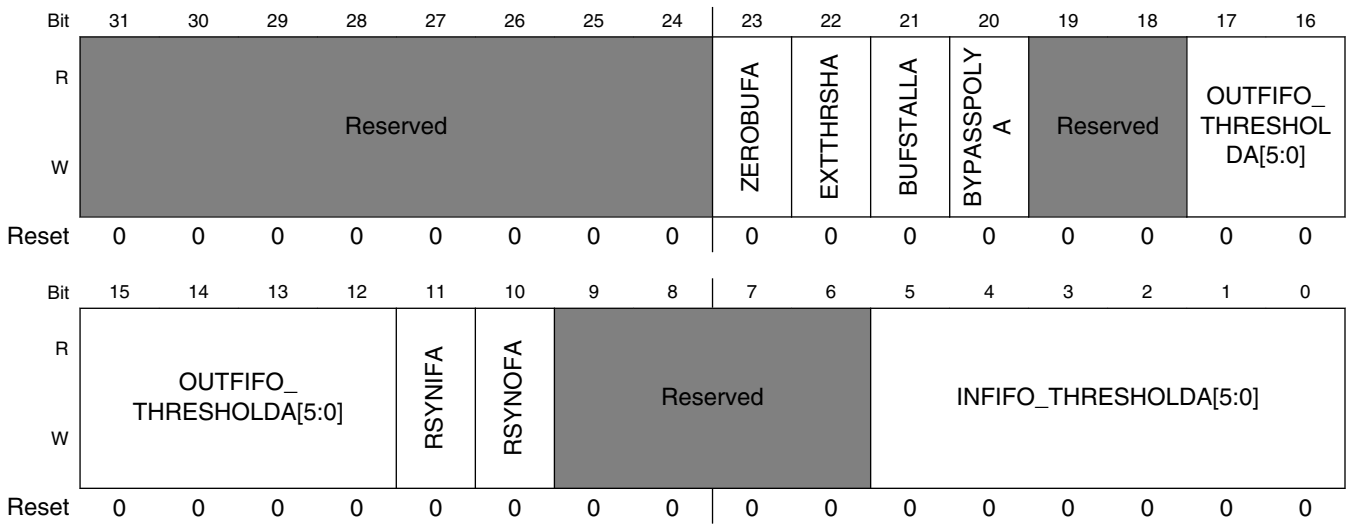
ASRC_ASR56K field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–17 -	This field is reserved. Reserved
ASR56K	Value for the period of the 56 kHz sampling clock

16.7.22 ASRC Misc Control Register for Pair A (ASRC_ASRMCRA)

The register (ASRMCRA) is used to control Pair A internal logic.

Address: 203_4000h base + A0h offset = 203_40A0h



ASRC_ASRMCRA field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 ZEROBUFA	Initialize buf of Pair A when pair A is enabled. Always clear option. This bit is used to control whether the buffer is to be zeroized when pair A is enabled. 1 Don't zeroize the buffer 0 Zeroize the buffer
22 EXTTHRSHA	Use external thresholds for FIFO control of Pair A This bit will determine whether the FIFO thresholds externally defined in this register is used to control ASRC internal FIFO logic for pair A. 1 Use external defined thresholds. 0 Use default thresholds.
21 BUFSTALLA	Stall Pair A conversion in case of Buffer Near Empty/Full Condition This bit will determine whether the near empty/full FIFO condition will stall the rate conversion for pair A. This option can only work when external ratio is used. Near empty condition is the condition when input FIFO has less than 4 useful samples per channel. Near full condition is the condition when the output FIFO has less than 4 vacant sample words to fill per channel.

Table continues on the next page...

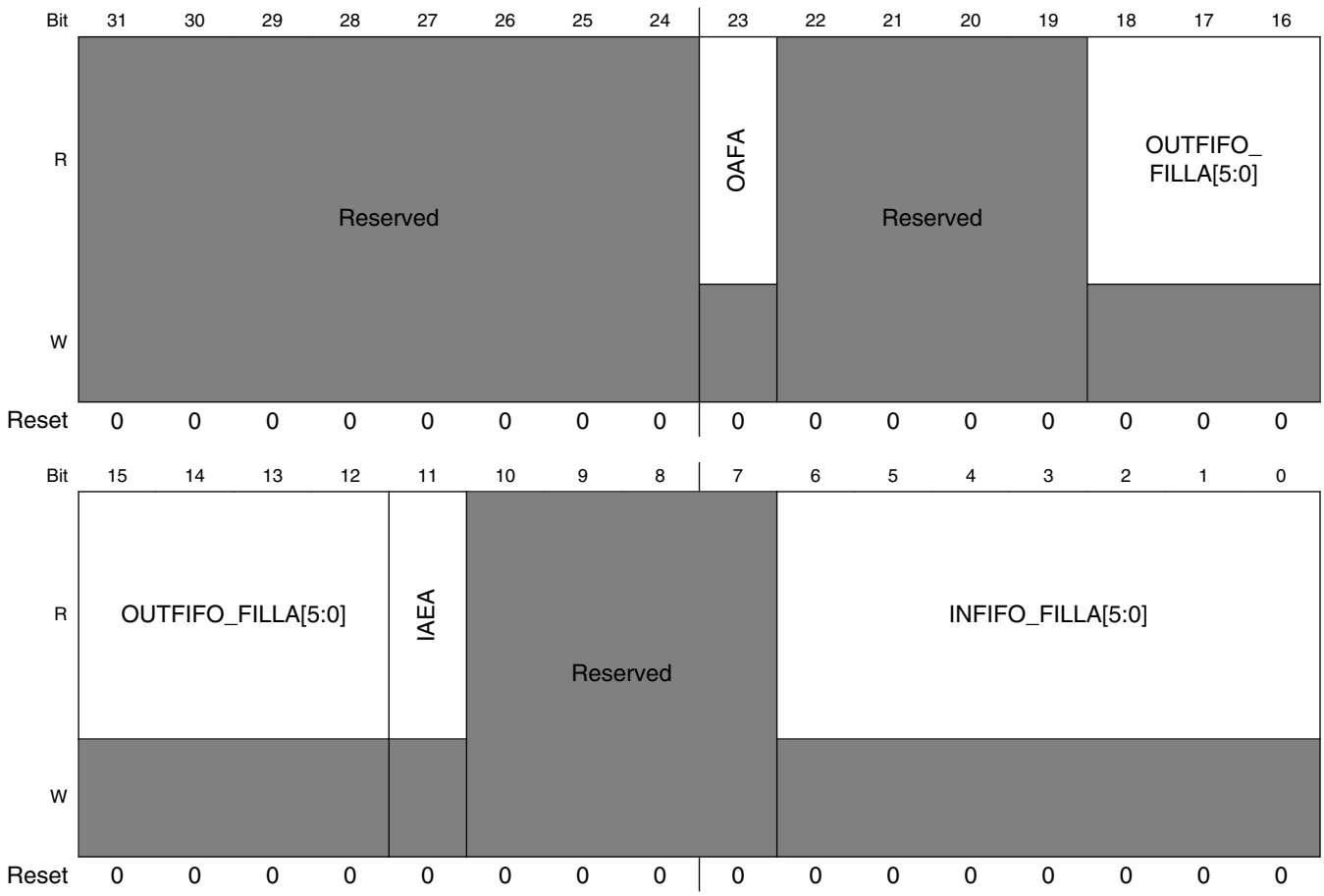
ASRC_ASRMCRA field descriptions (continued)

Field	Description
	1 Stall Pair A conversion in case of near empty/full FIFO conditions. 0 Don't stall Pair A conversion even in case of near empty/full FIFO conditions.
20 BYPASSPOLYA	Bypass Polyphase Filtering for Pair A This bit will determine whether the polyphase filtering part of Pair A conversion will be bypassed. 1 Bypass polyphase filtering. 0 Don't bypass polyphase filtering.
19–18 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
17–12 OUTFIFO_ THRESHOLDA[5:0]	The threshold for Pair A's output FIFO per channel These bits stand for the threshold for Pair A's output FIFO per channel. Possible range is [0,63]. When the value is n, it means that: when the number of output FIFO fillings of the pair is greater than n samples per channel, the output data ready flag is set; when the number of output FIFO fillings of the pair is less than or equal to n samples per channel, the output data ready flag is automatically cleared.
11 RSYNIFA	Re-sync Input FIFO Channel Counter If bit set, force ASRCCR:ACIA=0. If bit clear, untouch ASRCCR:ACIA.
10 RSYNOFA	Re-sync Output FIFO Channel Counter If bit set, force ASRCCR:ACOA=0. If bit clear, untouch ASRCCR:ACOA.
9–6 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
INFIFO_ THRESHOLDA[5:0]	The threshold for Pair A's input FIFO per channel These bits stand for the threshold for Pair A's input FIFO per channel. Possible range is [0,63]. When the value is n, it means that: when the number of input FIFO fillings of the pair is less than n samples per channel, the input data needed flag is set; when the number of input FIFO fillings of the pair is greater than or equal to n samples per channel, the input data needed flag is automatically cleared.

16.7.23 ASRC FIFO Status Register for Pair A (ASRC_ASRFSTA)

The register (ASRFSTA) is used to show Pair A internal FIFO conditions.

Address: 203_4000h base + A4h offset = 203_40A4h



ASRC_ASRFSTA field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 OAFA	Output FIFO is near Full for Pair A This bit is to indicate whether the output FIFO of Pair A is near full.
22–19 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
18–12 OUTFIFO_ FILLA[5:0]	The fillings for Pair A's output FIFO per channel These bits stand for the fillings for Pair A's output FIFO per channel. Possible range is [0,64].

Table continues on the next page...

ASRC_ASRFSTA field descriptions (continued)

Field	Description
11 IAEA	Input FIFO is near Empty for Pair A This bit is to indicate whether the input FIFO of Pair A is near empty.
10–7 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
INFIFO_FILLA[5:0]	The fillings for Pair A's input FIFO per channel These bits stand for the fillings for Pair A's input FIFO per channel. Possible range is [0,64].

16.7.24 ASRC Misc Control Register for Pair B (ASRC_ASRMCRB)

The register (ASRMCRB) is used to control Pair B internal logic.

Address: 203_4000h base + A8h offset = 203_40A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ZEROBUFB	EXTTHRSHB	BUFSTALLB	BYPASSPOLYB	Reserved		OUTFIFO_THRESHOLDB[5:0]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUTFIFO_THRESHOLDB[5:0]				RSYNIFB	RSYNOFB	Reserved				INFIFO_THRESHOLDB[5:0]					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ASRC_ASRMCRB field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 ZEROBUFB	Initialize buf of Pair B when pair B is enabled This bit is used to control whether the buffer is to be zeroized when pair B is enabled. 1 Don't zeroize the buffer 0 Zeroize the buffer

Table continues on the next page...

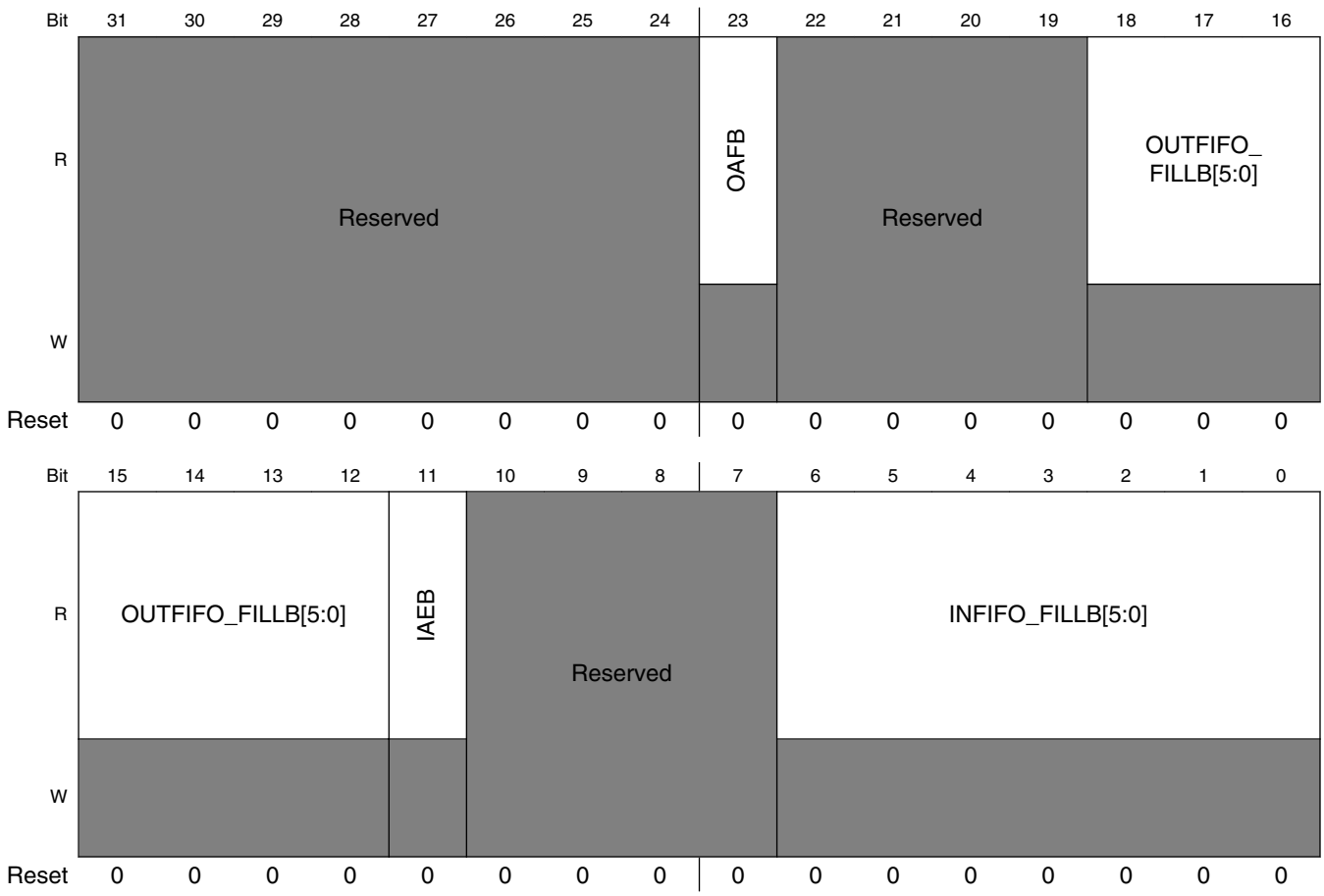
ASRC_ASRMCRB field descriptions (continued)

Field	Description
22 EXTTHRSHB	Use external thresholds for FIFO control of Pair B This bit will determine whether the FIFO thresholds externally defined in this register is used to control ASRC internal FIFO logic for pair B. 1 Use external defined thresholds. 0 Use default thresholds.
21 BUFSTALLB	Stall Pair B conversion in case of Buffer Near Empty/Full Condition This bit will determine whether the near empty/full FIFO condition will stall the rate conversion for pair B. This option can only work when external ratio is used. Near empty condition is the condition when input FIFO has less than 4 useful samples per channel. Near full condition is the condition when the output FIFO has less than 4 vacant sample words to fill per channel. 1 Stall Pair B conversion in case of near empty/full FIFO conditions. 0 Don't stall Pair B conversion even in case of near empty/full FIFO conditions.
20 BYPASSPOLYB	Bypass Polyphase Filtering for Pair B This bit will determine whether the polyphase filtering part of Pair B conversion will be bypassed. 1 Bypass polyphase filtering. 0 Don't bypass polyphase filtering.
19–18 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
17–12 OUTFIFO_ THRESHOLDB[5:0]	The threshold for Pair B's output FIFO per channel These bits stand for the threshold for Pair B's output FIFO per channel. Possible range is [0,63]. When the value is n, it means that: when the number of output FIFO fillings of the pair is greater than n samples per channel, the output data ready flag is set; when the number of output FIFO fillings of the pair is less than or equal to n samples per channel, the output data ready flag is automatically cleared.
11 RSYNIFB	Re-sync Input FIFO Channel Counter If bit set, force ASRCCR:ACIB=0. If bit clear, untouch ASRCCR:ACIB.
10 RSYNOFB	Re-sync Output FIFO Channel Counter If bit set, force ASRCCR:ACOB=0. If bit clear, untouch ASRCCR:ACOB.
9–6 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
INFIFO_ THRESHOLDB[5:0]	The threshold for Pair B's input FIFO per channel These bits stand for the threshold for Pair B's input FIFO per channel. Possible range is [0,63]. When the value is n, it means that: when the number of input FIFO fillings of the pair is less than n samples per channel, the input data needed flag is set; when the number of input FIFO fillings of the pair is greater than or equal to n samples per channel, the input data needed flag is automatically cleared.

16.7.25 ASRC FIFO Status Register for Pair B (ASRC_ASRFSTB)

The register (ASRFSTB) is used to show Pair B internal FIFO conditions.

Address: 203_4000h base + ACh offset = 203_40ACh



ASRC_ASRFSTB field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 OAFB	Output FIFO is near Full for Pair B This bit is to indicate whether the output FIFO of Pair B is near full.
22–19 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
18–12 OUTFIFO_ FILLB[5:0]	The fillings for Pair B's output FIFO per channel These bits stand for the fillings for Pair B's output FIFO per channel. Possible range is [0,64].

Table continues on the next page...

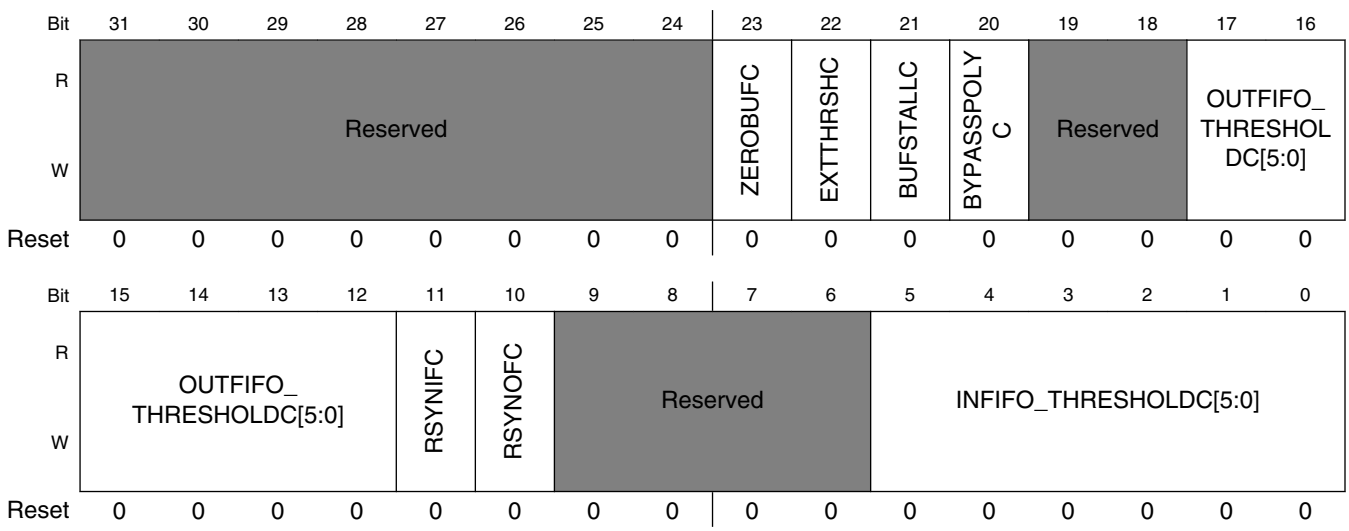
ASRC_ASRFSTB field descriptions (continued)

Field	Description
11 IAEB	Input FIFO is near Empty for Pair B This bit is to indicate whether the input FIFO of Pair B is near empty.
10–7 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
INFIFO_FILLB[5:0]	The fillings for Pair B's input FIFO per channel These bits stand for the fillings for Pair B's input FIFO per channel. Possible range is [0,64].

16.7.26 ASRC Misc Control Register for Pair C (ASRC_ASRMCRC)

The register (ASRMCRC) is used to control Pair C internal logic.

Address: 203_4000h base + B0h offset = 203_40B0h



ASRC_ASRMCRC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 ZEROBUFC	Initialize buf of Pair C when pair C is enabled This bit is used to control whether the buffer is to be zeroized when pair C is enabled. 1 Don't zeroize the buffer 0 Zeroize the buffer

Table continues on the next page...

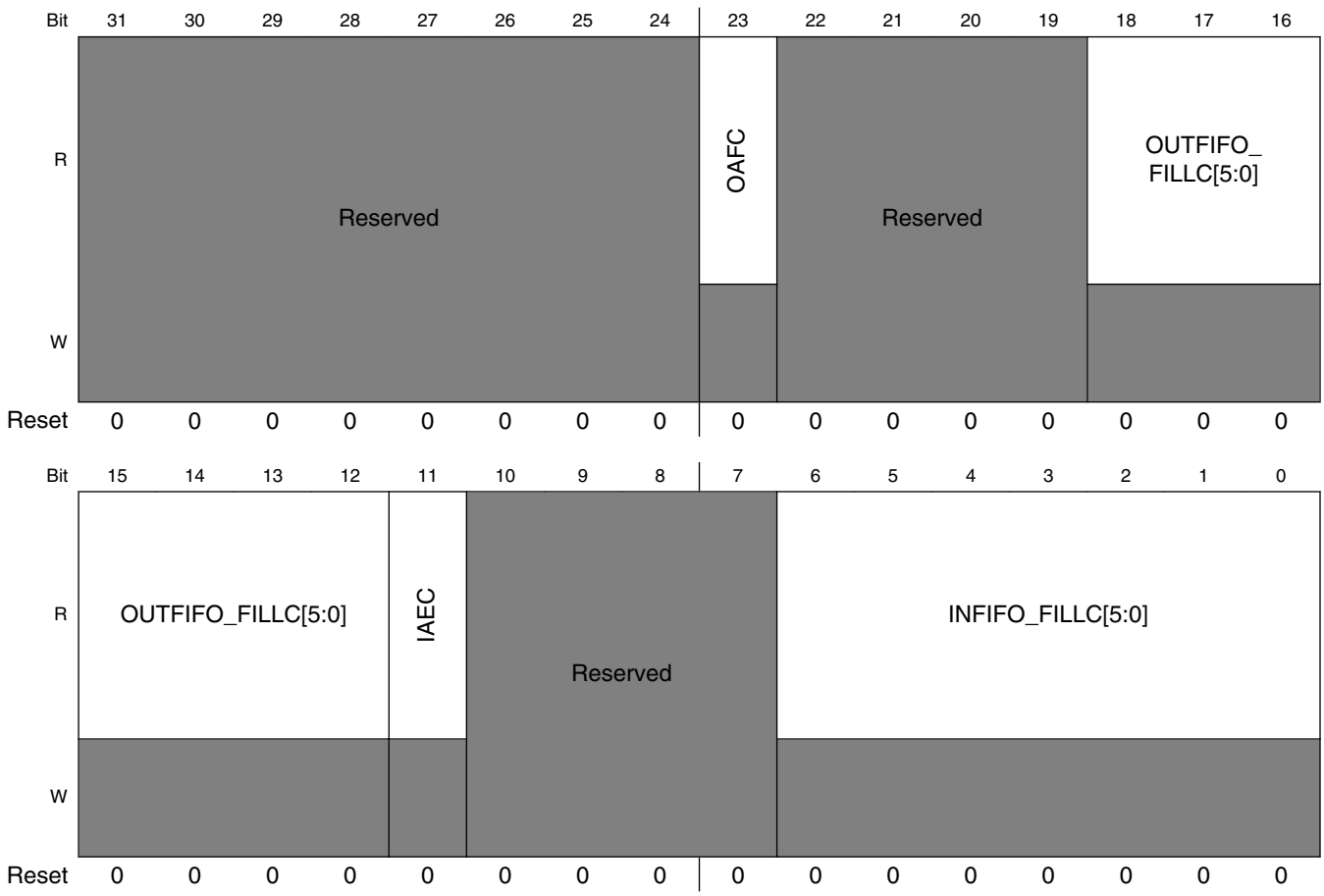
ASRC_ASRMCRC field descriptions (continued)

Field	Description
22 EXTTHRSHC	Use external thresholds for FIFO control of Pair C This bit will determine whether the FIFO thresholds externally defined in this register is used to control ASRC internal FIFO logic for pair C. 1 Use external defined thresholds. 0 Use default thresholds.
21 BUFSTALLC	Stall Pair C conversion in case of Buffer Near Empty/Full Condition This bit will determine whether the near empty/full FIFO condition will stall the rate conversion for pair C. This option can only work when external ratio is used. Near empty condition is the condition when input FIFO has less than 4 useful samples per channel. Near full condition is the condition when the output FIFO has less than 4 vacant sample words to fill per channel. 1 Stall Pair C conversion in case of near empty/full FIFO conditions. 0 Don't stall Pair C conversion even in case of near empty/full FIFO conditions.
20 BYPASSPOLYC	Bypass Polyphase Filtering for Pair C This bit will determine whether the polyphase filtering part of Pair C conversion will be bypassed. 1 Bypass polyphase filtering. 0 Don't bypass polyphase filtering.
19–18 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
17–12 OUTFIFO_ THRESHOLDC[5:0]	The threshold for Pair C's output FIFO per channel These bits stand for the threshold for Pair C's output FIFO per channel. Possible range is [0,63]. When the value is n, it means that: when the number of output FIFO fillings of the pair is greater than n samples per channel, the output data ready flag is set; when the number of output FIFO fillings of the pair is less than or equal to n samples per channel, the output data ready flag is automatically cleared.
11 RSYNIFC	Re-sync Input FIFO Channel Counter If bit set, force ASRCCR:ACIC=0. If bit clear, untouch ASRCCR:ACIC.
10 RSYNOFC	Re-sync Output FIFO Channel Counter If bit set, force ASRCCR:ACOC=0. If bit clear, untouch ASRCCR:ACOC.
9–6 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
INFIFO_ THRESHOLDC[5:0]	The threshold for Pair C's input FIFO per channel These bits stand for the threshold for Pair C's input FIFO per channel. Possible range is [0,63]. When the value is n, it means that: when the number of input FIFO fillings of the pair is less than n samples per channel, the input data needed flag is set; when the number of input FIFO fillings of the pair is greater than or equal to n samples per channel, the input data needed flag is automatically cleared.

16.7.27 ASRC FIFO Status Register for Pair C (ASRC_ASRFSTC)

The register (ASRFSTC) is used to show Pair C internal FIFO conditions.

Address: 203_4000h base + B4h offset = 203_40B4h



ASRC_ASRFSTC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 OAFC	Output FIFO is near Full for Pair C This bit is to indicate whether the output FIFO of Pair C is near full.
22–19 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
18–12 OUTFIFO_ FILLC[5:0]	The fillings for Pair C's output FIFO per channel These bits stand for the fillings for Pair C's output FIFO per channel. Possible range is [0,64].

Table continues on the next page...

ASRC_ASRFSTC field descriptions (continued)

Field	Description
11 IAEC	Input FIFO is near Empty for Pair C This bit is to indicate whether the input FIFO of Pair C is near empty.
10–7 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
INFIFO_FILLC[5:0]	The fillings for Pair C's input FIFO per channel These bits stand for the fillings for Pair C's input FIFO per channel. Possible range is [0,64].

16.7.28 ASRC Misc Control Register 1 for Pair X (ASRC_ASRMCR1*n*)

The register (ASRMCR1x) is used to control Pair *x* internal logic (for data alignment etc.).

The bit assignment for all the input data formats is the same as that supported by the SAI.

Address: 203_4000h base + C0h offset + (4d × i), where i=0d to 2d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				IWD[2:0]			IMSB	Reserved					OMSB	OSGN	OW16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ASRC_ASRMCR1*n* field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–12 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
11–9 IWD[2:0]	Data Width of the input FIFO These three bits will determine the bitwidth for the audio data into ASRC All other settings not shown are reserved. 3'b000 24-bit audio data.

Table continues on the next page...

ASRC_ASRMCR1*n* field descriptions (continued)

Field	Description
	3'b001 16-bit audio data. 3'b010 8-bit audio data.
8 IMSB	Data Alignment of the input FIFO This bit will determine the data alignment of the input FIFO. 1 MSB aligned. 0 LSB aligned.
7–3 -	This field is reserved. Reserved. Should be written as zero for future compatibility.
2 OMSB	Data Alignment of the output FIFO This bit will determine the data alignment of the output FIFO. 1 MSB aligned. 0 LSB aligned.
1 OSGN	Sign Extension Option of the output FIFO This bit will determine the sign extension option of the output FIFO. 1 Sign extension. 0 No sign extension.
0 OW16	Bit Width Option of the output FIFO This bit will determine the bit width option of the output FIFO. 1 16-bit output data 0 24-bit output data.

Chapter 17
40-BIT Correcting ECC Accelerator (BCH)

17.1 Overview

The hardware ECC accelerator provides a forward error-correction function for improving the reliability of various storage media that may be attached to the device.

For example, NAND flash devices use a spare area to store ecc codes to correct some hard bit errors in data stored within the device, allowing higher device yields and, therefore, lower NAND device costs.

The Bose, Ray-Chaudhuri, Hocquenghem (BCH) Encoder and Decoder module is capable of correcting from 2 to 40 single bit errors within a block of data no larger than about 1900 bytes (512 bytes or 1024 bytes are typical) in applications such as protecting data and resources stored on modern NAND flash devices. The correction level in the BCH block is programmable to provide flexibility for varying applications and configurations of flash page size. The design can be programmed to encode protection of 2 to 40 bit errors when writing flash and to correct the corresponding number of errors on decode. The correction level when decoding MUST be programmed to the same correction level as was used during the encode phase.

BCH-codes are a type of block-code, which implies that all error-correction is performed over a block of N-symbols. The BCH operation will be performed over GF(2^{13} = 8192) or GF(2^{14} = 16384), which is the Galois Field consisting of 8191 or 16383 one-bit symbols. BCH-encoding (or encode for any block-code) can be performed by two algorithms: systematic encoding or multiplicative encoding. Systematic encoding is the process of reading all the symbols which constitute a block, dividing continuously these symbols by the generator polynomial for the GF(8192) or GF(16384) and appending the resulting ***t*** parity symbols to the block to create a BCH codeword (where ***t*** is the number of correctable bits).

The BCH encode process creates *t**13 (or *t**14)-bit parity symbols for each data block when the data is written to the flash device. The parity symbols are written to the flash device after the corresponding data block, and together these are collectively called the codeword. The codeword can be used during the decode process to correct errors that occur in either the data or parity blocks.

The BCH decoder processes code words in a 4-step fashion:

1. Syndrome Calculation (SC): This is the process of reading in all of the symbols of the codeword and continuously dividing by the generator polynomial for the field. 2**t* syndromes must be calculated for each codeword and inspection of the syndromes determines if there are errors: a non-zero set of syndromes indicates one or more errors. This process is implemented parallel hardware to minimize processing time since it must be done every time the decode is performed.
2. Key Equation Solver (KES): The syndromes represent 2t-linear equations with 2t-unknown variables. The process of solving these equations and selecting from the numerous solutions constitutes the KES module. When the KES block completes its operations, it generates an error locator polynomial (sigma) that is used in the proceeding block to determine the locations and values of the errors.
3. Chien Search (CS): This block takes input from the KES block and uses the Chien Algorithm for finding the locations of the errors based on the error locator polynomial. The method basically involves substituting all 8191 symbols from the GF(8192) or 16383 symbols from the GF(16383) into the locator polynomial. All evaluations that produce a zero solution indicate locations of the various errors. Since each located error corresponds to a single bit, the bit in the original data may be corrected by simply flipping the polarity of the incorrect location.
4. Correction: this block has to convert the symbol index and mask information to memory byte indexes and masks.

The BCH block, shown in the figure, was designed to operate in a pipelined fashion to maximize throughput. Aside from the initial latency to fill the pipeline stages, the BCH throughput is about 7/4 cycles/byte. Thus, the bottleneck in performing NAND reads and error corrections is the BCH rate. Current GPMI read rates are approximately 1/2 cycles/byte maximally for the current generation of NAND flash. Fortunately, BCH has a different master clock from GPMI, this gives some flexibility to match the throughput rate. The CPU is not directly involved in generating parity symbols, checking for errors, or correcting them.

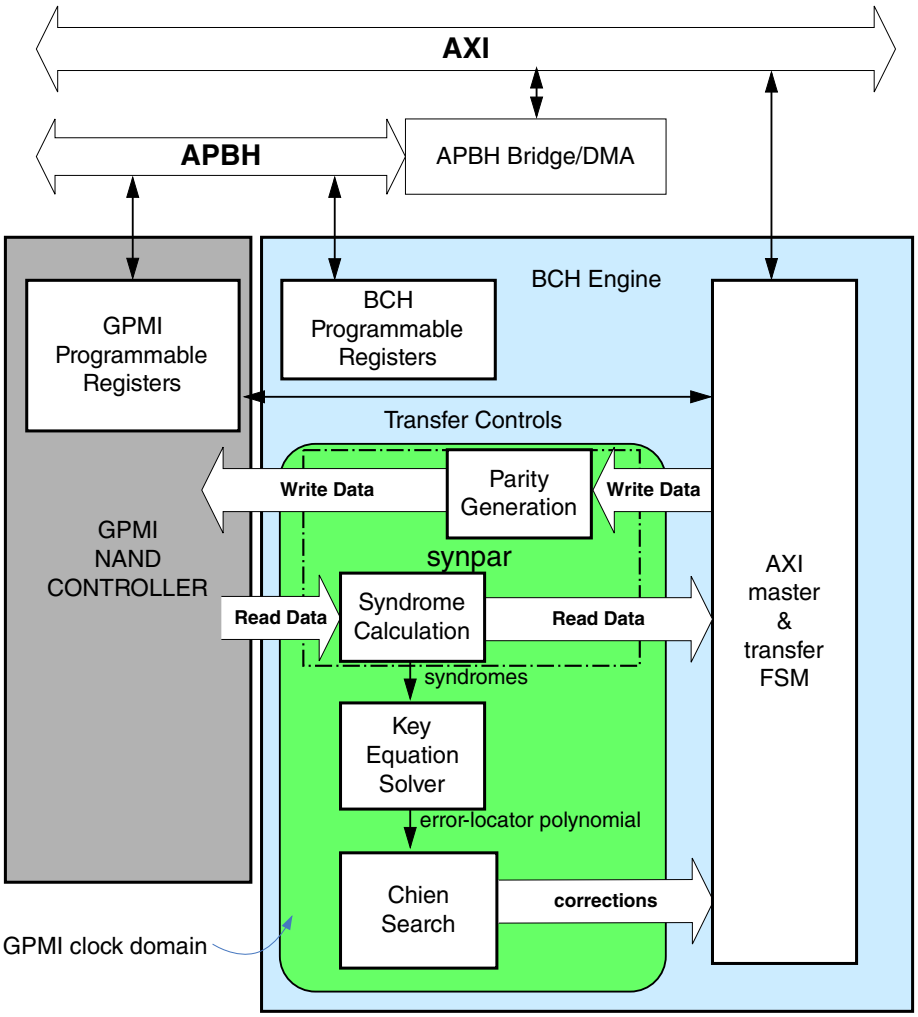


Figure 17-1. Hardware BCH Accelerator

17.2 Operation

Before performing any NAND flash read or write operations, software should first program the BCH's flash layout registers (see Flash Page Layout) to specify how data is to be formatted on the flash device.

The BCH hardware allows full programmability over the flash page layout to enable users flexibility in balancing ECC correction levels and ever-changing flash page sizes.

To initiate a NAND Flash write, software will program a GPMI DMA operation. The DMA need only program the GPMI control registers (and handle the requisite flash addressing handshakes) since the BCH will handle all data operations using its AXI bus interface. The BCH will then send the data to the GPMI controller to be written to flash

as it computes the parity symbols. At the end of each data block the BCH will insert the parity symbols into the data stream so that the GPMI sees only a continuous stream of data to be written.

NAND Flash read operations operate in a similar manner. As the GPMI controller reads the device, all data is sent to the BCH hardware for error detection/correction. The BCH controller writes all incoming read data to system memory and in parallel computes the syndromes used to detect bit errors. If errors are detected within a block, the BCH hardware activates the error correction logic to determine where bit errors have occurred and ultimately correct them in the data buffer in system memory. After an entire flash page has been read and corrected, the BCH will signal an interrupt to the CPU.

The figure below indicates how data read from the GPMI is operated on within the BCH hardware. As the BCH receives data from the GPMI (top row), it is written to memory by the BCH's Bus Interface Unit (BIU) (second row). For blocks requiring correction, the KES logic will be activated after the entire block has been received. Once the error locator polynomial has been computed, the corrections are determined by the Chien Search and fed back to the BIU, which performs a read, modify, write operation on the buffer in memory to correct the data.

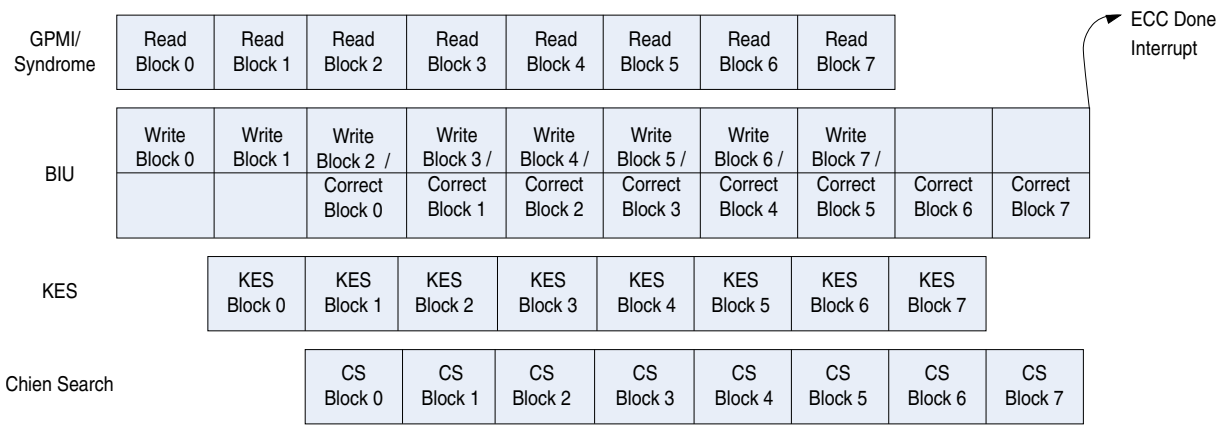


Figure 17-2. Block Pipeline while Reading Flash

17.2.1 BCH Limitations and Assumptions

- The BCH is programmable to support 2 to 40 bit error correction. ECC0 is supported as a pass-through, non-correcting mode.
- Data block sizes must be a multiple of 4 bytes and be aligned in system memory.
- The BCH supports a programmable number of metadata/auxiliary data bytes, from 0 to 255.

- Metadata will be written at the beginning of the flash page to facilitate fast access for filesystem operations.
- Metadata may be treated as an independent block for ECC purposes or combined with the first data block to conserve bits in the flash.
- The BCH does not support a partial page write (this can be accomplished by programming the BCH layout registers such that the BCH only sees a portion of the page).
- Flash read operations can read the entire page or the first block on the page.
- The BCH also supports a memory-to-memory mode of operation that does not require the use of DMA or the GPMI.

17.2.2 Flash Page Layout

The BCH supports a fully programmable flash page layout. The BCH maintains four independent layout registers that can describe four completely different NAND devices or layouts.

When the BCH initiates an operation, it selects one of the layouts by using the chip select as an index into the BCH_LAYOUTSELECT register that determines which layout should be used for the operation.

Three possible (generic) flash layout schemes are supported, as indicated in the figure below. (In each case, the metadata size may also be programmed to 0 bytes). Metadata may either be combined with the first block of data or the size of the first data block can be programmed to 0 to allow the metadata to be protected by its own ECC parity bits.

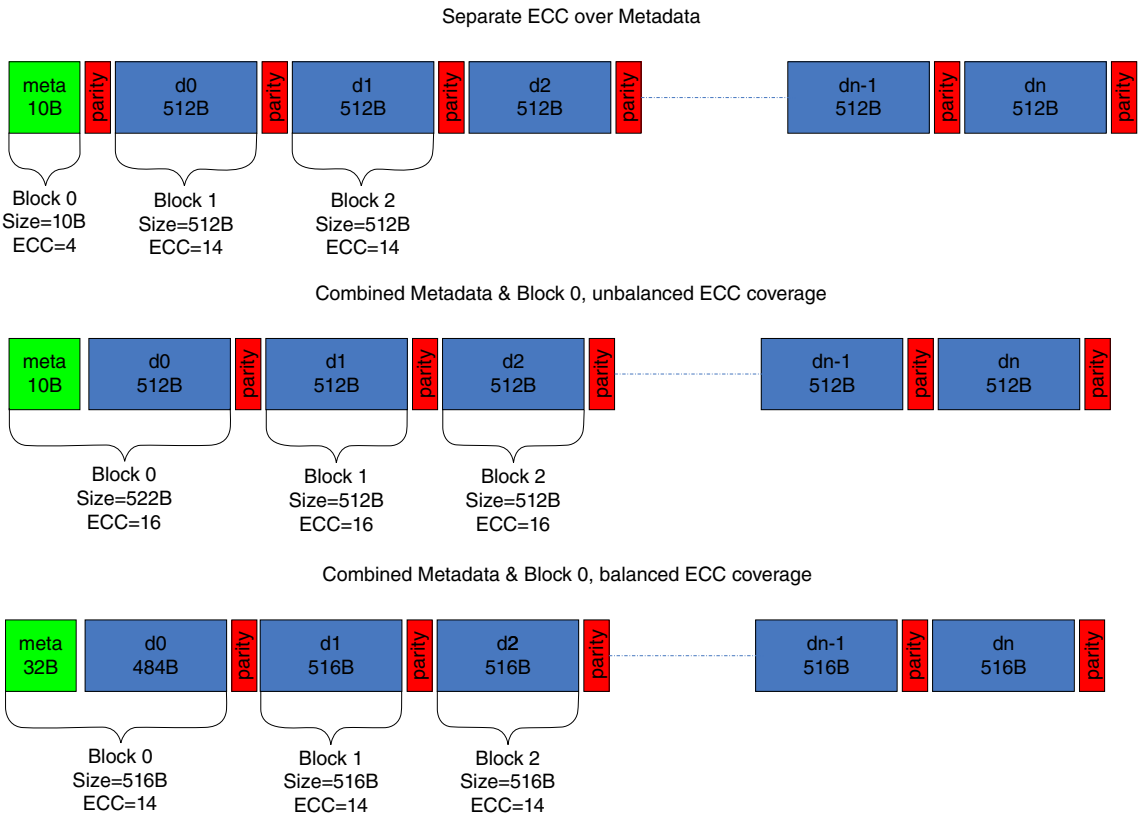


Figure 17-3. FLASH Page Layout Options

Each layout is determined by a pair of registers that define the following parameters:

- DATA0_SIZE: Indicates the number of data bytes in the first block on the page (this should not include parity or metadata bytes). This should be set to 0 when the metadata is to be covered separately with its own ECC. This must be a multiple of 4 bytes.
- ECC0: Indicates the ECC level to be used for the first block on the flash (data0+metadata).
- META_SIZE: Indicates the number of bytes (from 0-255) that are stored as metadata.
- NBLOCKS: Indicates the number of subsequent DATAN blocks on the flash, or the number of blocks following the DATA0 block.
- DATAN_SIZE: Indicates the number of data bytes in all subsequent data blocks. This MUST be a multiple of 4 bytes.
- ECCN: Indicates the ECC level to be used for the subsequent data blocks.

- GF0 or GFN: Indicates the Galois field the meta / data blocks are using
- PAGE_SIZE: Indicates the total number of bytes available per page on the physical flash device. This includes the spare area and is typically 4096+128, 4096+218, or 2048+64 bytes.

17.2.3 Determining the ECC layout for a device

Since the BCH is programmable, a system can trade off ECC levels for flash size and layout configurations.

The following examples indicate how to determine a valid layout based on the required storage space and flash size. For all cases, the size of the parity will be 13 (or 14 for GF(2^{14}))*ECC level ***bits***-- so for ECC8, 13 (or 14) bytes are required (per block).

17.2.3.1 4K+218 flash, 10 bytes metadata, 512 byte data blocks, separate metadata, Assuming GF(2^{13})

In this case, we have 8 data blocks each consisting of 512 bytes. Since the flash has 218 spare bytes (1744 bits), first estimate an ECC level for the data blocks by first subtracting the number of metadata bytes from the spare bytes (218 – 10 = 208 bytes = 1664 bits) then dividing the number of bits by 8 (number of blocks) and then by 13 (bits per ECC level).

(218 - 10) x 8 = 1664/13(8) = 16

Therefore all the data blocks could be covered by ECC16 if the metadata had no parity. This isn't acceptable, so assume ECC14 for all the data blocks. Now calculate the number of free bits for the metadata parity as

1664 - (14) x 13 x 8 = 208

Therefore, 208 bits remain for metadata parity. Dividing by 13 (bits/ECC) gives 16, so the metadata can be covered with ECC16. The settings for this device would then be

Table 17-1. Settings for 4K+218 FLASH

Setting	Value
PAGE_SIZE	4096+218=4314=0x10DA
META_SIZE	10=0x0A
DATA0_SIZE	0
ECC0	16=0x10

Table continues on the next page...

Table 17-1. Settings for 4K+218 FLASH (continued)

Setting	Value
GF0	GF(2^{13})
DATAN_SIZE	512=0x200 (in register interface, assigned as 0x80)
ECCN	14=0x0E
GFN	GF(2^{13})
NBLOCKS	8

17.2.3.2 4K+128 flash, 10 bytes metadata, 1024 byte data blocks, separate metadata, assuming GF(2^{13}) for data and GF(2^{14}) for metadata

This flash will have 118 bytes available for ECC (after subtracting the metadata size), therefore, 994 bits.

Dividing by 4*14 (number of blocks * ECC level) we get 17.75, therefore we can support ECC16 on the data blocks. The number of free spare bits becomes 944 - 16 * 4 * 14 = 944 - 896 = 48, divided by 13 = 3.69, therefore the metadata can be also covered by ECC2.

Table 17-2. Settings for 4K+128 FLASH

Setting	Value
PAGE_SIZE	4096+128=4224=0x1080
META_SIZE	10=0x0A
DATA0_SIZE	0
ECC0	2
GF0	GF(2^{13})
DATAN_SIZE	1024=0x400 (in register interface, assigned as 0x100)
ECCN	16
GFN	GF(2^{14})
NBLOCKS	4

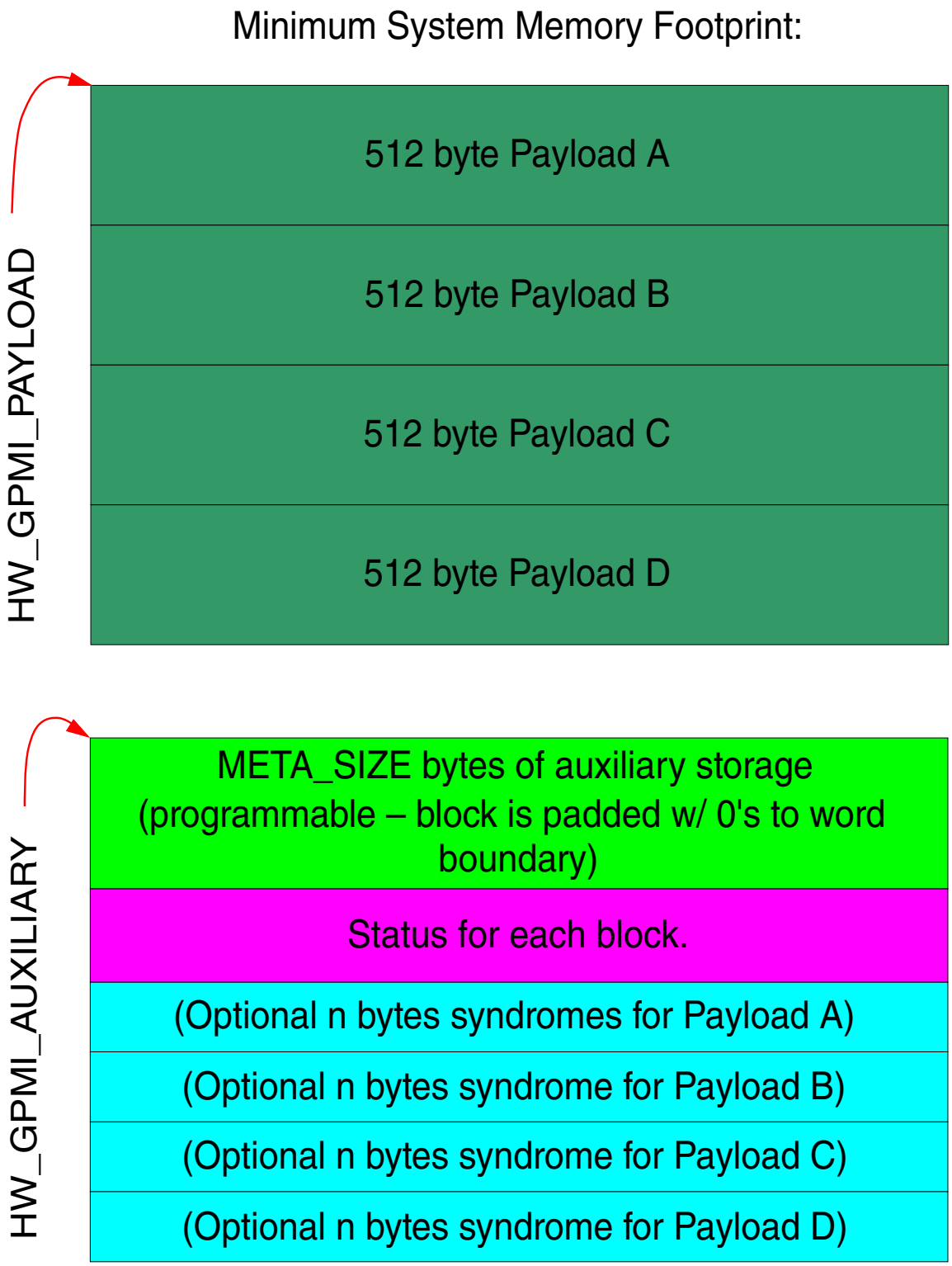
In this case, there will be additional unused spare bits, with the BCH will pad out with zeros.

17.2.4 Data Buffers in System Memory

While the data on the flash is interleaved with parity symbols, the BCH assumes that the data buffers in memory are contiguous.

Metadata read from the flash will be stored to the location pointed to by the GPMI_AUXILIARY register and data will be written to the address specified in the GPMI_PAYLOAD register as is shown in the following figure where the block length is 512 bytes for example. Since the number of blocks on a flash page is programmable, the BCH also writes individual block correction status to the auxiliary pointer at the word-aligned address following the end of the metadata. Optionally, the computed syndromes may also be written to the auxiliary area if the DEBUGSYNDROME bit is set in the control register.

As blocks complete processing, the bus master will accumulate the status for each block and write it to the auxiliary data buffer following the metadata. The metadata area will be padded with 0's until the next word boundary and the status for blocks 0-3 will be written to the next word. The status for subsequent blocks will then be written to the buffer. The status for the first block (metadata block) is also stored in the STATUS_BLK0 register in the BCH_STATUS register. The completion codes for the blocks are indicated in the Table 17-3. Note that the definition of the bytes and their ordering in the auxiliary and payload storage areas are user defined. When this data is read back from the flash and put into memory, it will resemble the original buffer that was written out to the flash.



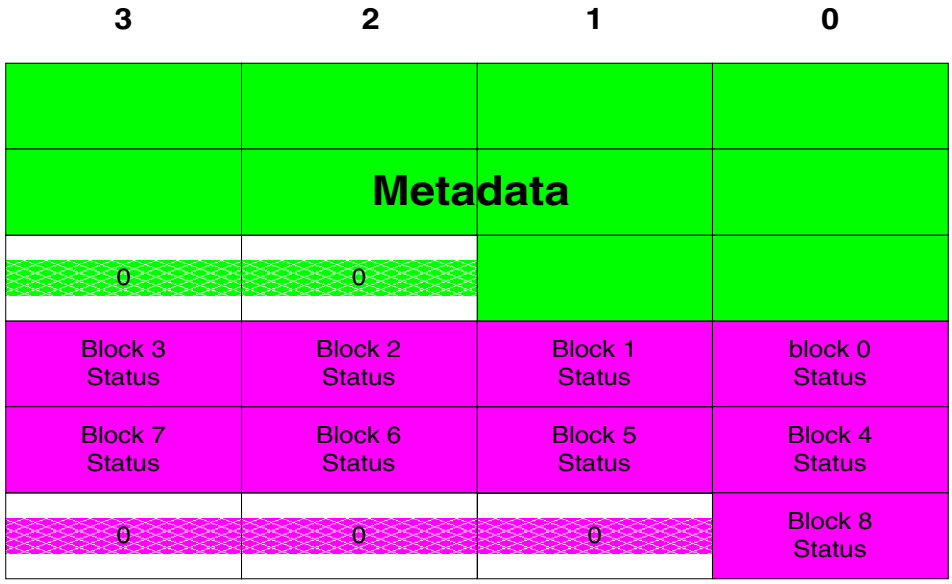
*Computed syndrome area consists of 2*t 13 (or 14)-bit symbols written as 16-bit half word.*

Figure 17-4. BCH Data Buffers in Memory

Table 17-3. Status Block Completion Codes

Code	Description
0xFF	Block is erased
0xFE	Block is uncorrectable
0x00	No errors found
0x01-0x28	Number of errors corrected

The following figure shows the layout of the bytes within the status field.



Status bytes are allocated based on the NBLOCKS programmed into the flash format register. The number of status bytes will be computed by the NBLOCKS+1. The status area will be padded with zeros to the next word boundary.

Syndrome data written for debug purposes will follow the end of the status block.

Figure 17-5. Memory-to-Memory Operations

17.3 Memory to Memory (Loopback) Operation

The BCH supports a memory-to-memory mode of operation where both the encoded and decoded buffers reside in system memory.

This can be useful for applications where data must be protected by ECC, but the storage device does not reside on the GPMI bus.

The BCH operation in memory to memory mode is much simpler than in GPMI mode since DMAs are not required to manage the operation. Instead, software simply writes the BCH_DATAPTR and BCH_METAPTR with the addresses of the data and metadata (auxiliary) buffers and the BCH_ENCODEPTR with the address of the buffer for encoded data. To initiate the operation, software simply sets the M2M_ENCODE and M2M_ENABLE bits in the control register. The BCH can be programmed to either issue an interrupt at the end of the operation or software may poll the status bits for completion.

Memory to memory decode operations work in a similar manner. The encoded data address is written to the BCH_ENCODEPTR and the data and meta pointers are written to buffers that correspond to the desired decoded data addresses. To initiate a decode, software must set the M2M_ENCODE bit to 0 while writing the M2M_ENABLE bit. Note that the addresses written to the BCH_DATAPTR, BCH_METAPTR and BCH_ENCODEPTR registers should always be aligned on a 4 byte boundary. In other words, the 2 lower bits of the address should always be written with zeros.

17.4 Programming the BCH/GPMI Interfaces

Programming the BCH for NAND operations consists largely of disabling the soft reset and clock bits (SFTRST and CLKGATE) from the BCH_CTRL register and then programming the flash layout registers to correspond to the format of the attached NAND device(s).

The BCH_LAYOUTSELECT register should also be programmed to map the chip select of each attached device into one of the four layout registers.

The bulk of the programming is actually applied to the GPMI through PIO operations embedded in DMA command structures. The DMA will perform all the requisite handshaking with the GPMI interface to negotiate the address portion of the transfer, then the BCH will handle all the movement of data from memory to the GPMI (writes) or the GPMI to memory (reads). The BCH will direct all data blocks to the buffer pointed to by the PAYLOAD_BUFFER and the metadata will be written to the AUXILIARY_BUFFER. Both of these registers are located in the GPMI PIO data space and are communicated to the BCH hardware at the beginning of the transfer. Thus, the normal multi-NAND DMA based device interleaving is preserved, that is, four NANDs on four separate chip selects can be scheduled for read or write operations using the BCH. Whichever channel finishes its ready wait first and enters the DMA arbiter with its lock bit set owns the GPMI command interface and through it owns the BCH resources for the duration of its processing.

17.4.1 BCH Encoding for NAND Writes

The BCH encoder flowchart in Figure 17-6 shows the detailed steps involved in programming and using the BCH encoder. This flowchart shows how to use the BCH block with the GPMI.

To use the BCH encoder with the GPMI's DMA, create a DMA command chain containing ten descriptor structures, as shown in Figure 17-8 and detailed in the DMA structure code example that follows it in DMA Structure Code Example. The ten descriptors perform the following tasks:

1. Disable the BCH block (in case it was enabled) and issue NAND write setup command byte (under CLE) and address bytes (under ALE).
2. Configure and enable the BCH and GPMI blocks to perform the NAND write.
3. Disable the BCH block and issue NAND write execute command byte (under CLE).
4. Wait for the NAND device to finish writing the data by watching the ready signal.
5. Check for NAND timeout through PSENSE.
6. Issue NAND status command byte (under CLE).
7. Read the status and compare against expected.
8. If status is incorrect or incomplete, branch to error handling descriptor chain.
9. Otherwise, write is complete and emit GPMI interrupt.

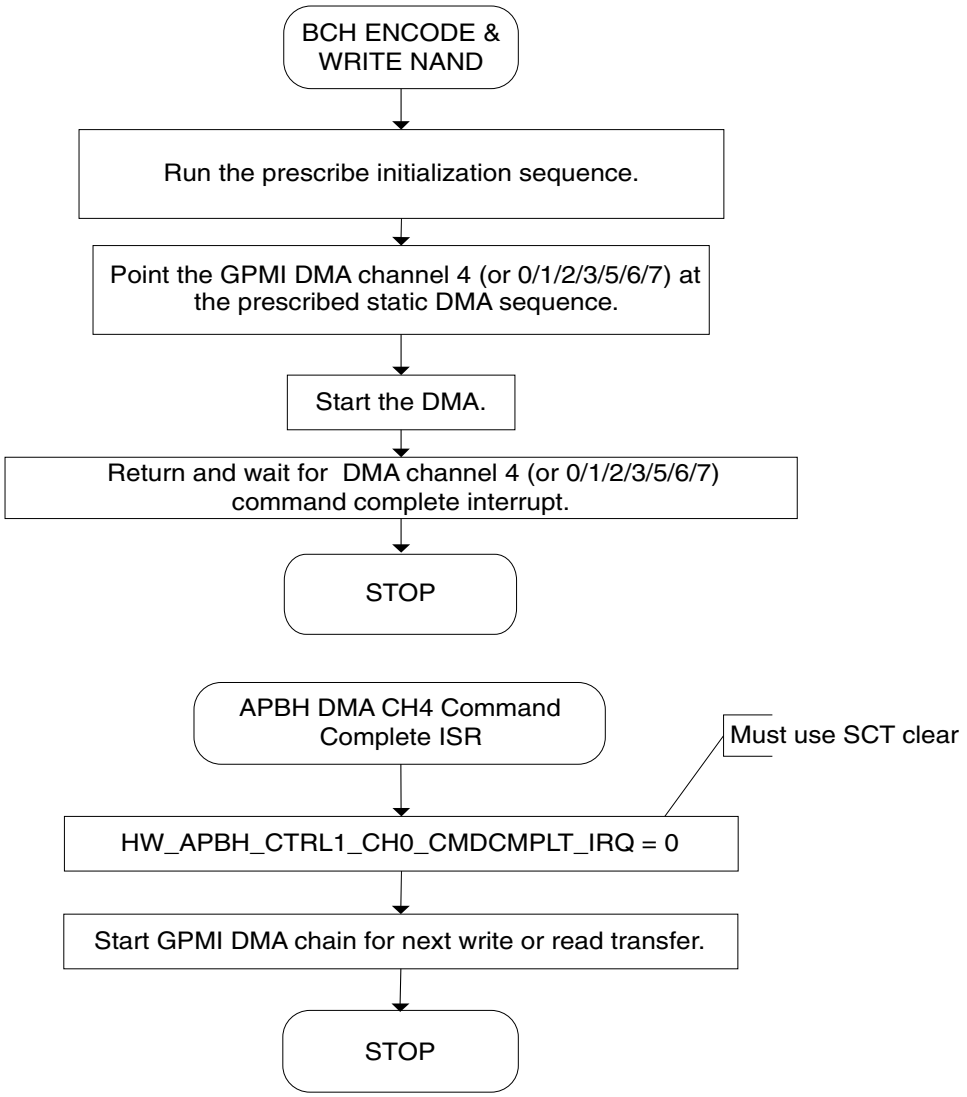


Figure 17-6. BCH Encode Flowchart

Descriptor Legend

NEXT CMD ADDR										
CMD	<=	xfer_count	cmdwords	wait4endcmd	semaphore	nandwait4ready	nandlock	irqoncmplt	chain	command
BUFFER ADDR										
HW_GPMI_CTRL0	<=	command_mode	word_length	lock_cs	CS	address	address_increment	xfer_count		
HW_GPMI_COMPARE	<=	mask				reference				
HW_GPMI_ECCCTRL	<=	ecc_cmd			enable_ecc				buffer_mask	
HW_GPMI_ECCCOUNT										
HW_GPMI_PAYLOAD										
HW_GPMI_AUXILIARY										

Figure 17-7. BCH DMA Descriptor Legend

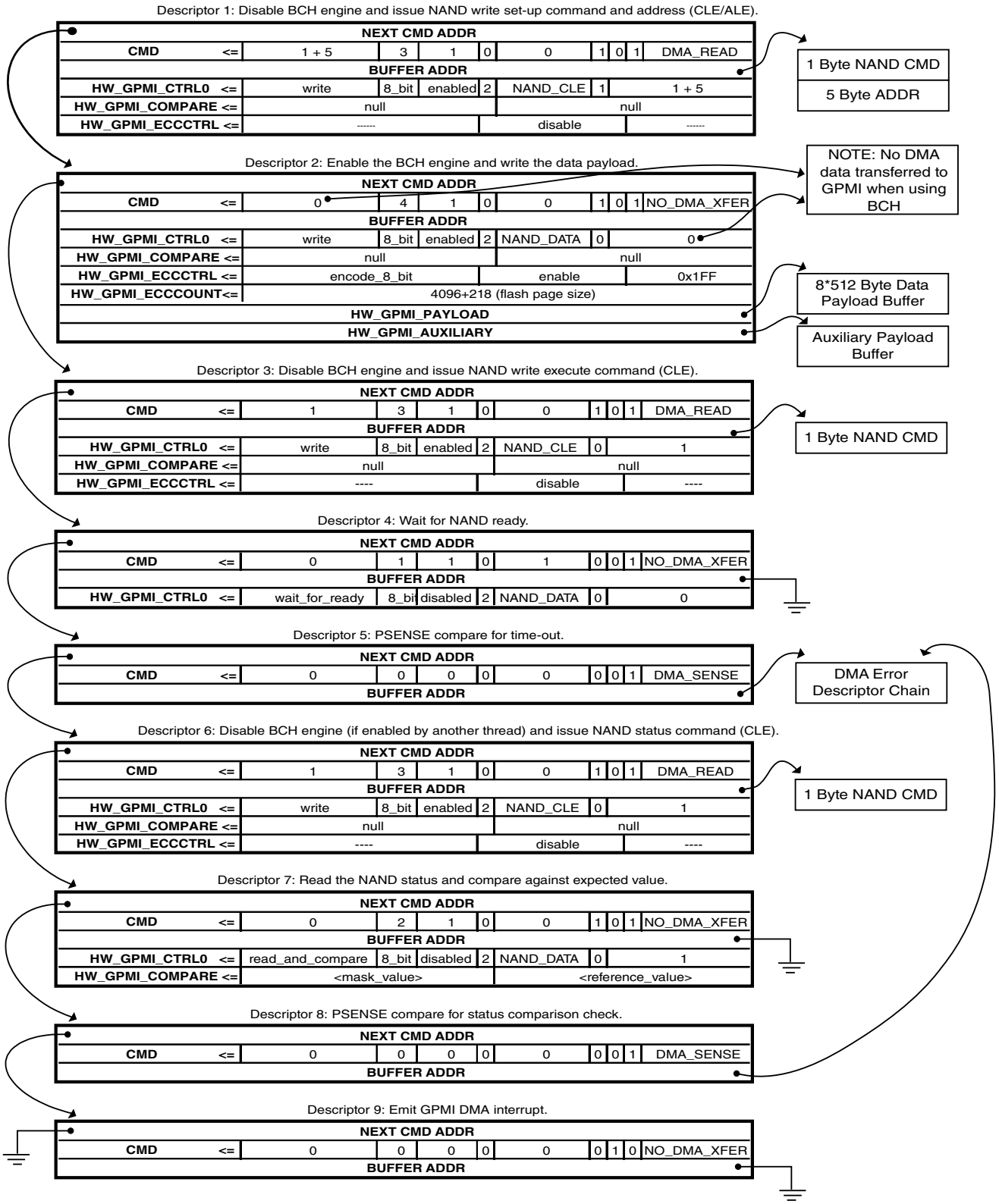


Figure 17-8. BCH Encode DMA Descriptor Chain

17.4.1.1 DMA Structure Code Example

The following code sample illustrates the coding for one write transaction involving 4096 bytes of data payload (eight 512-byte blocks) and 10 bytes of auxiliary payload (also referred to as metadata) to a 4K NAND page sitting on GPMI CS2.

```
//-----------------------------------------------------------------------------
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//-----------------------------------------------------------------------------
// allocate 10 descriptors for doing a NAND ECC Write
//-----------------------------------------------------------------------------
GENERIC_DESCRIPTOR write[10];
//-----------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//-----------------------------------------------------------------------------
unsigned int * dma_error_handler;
//-----------------------------------------------------------------------------
// 8 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is write setup command
//   bytes 1-5 is the NAND address
//   byte 6 is write execute command
//   byte 7 is status command
//-----------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[8];
//-----------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int write_payload_buffer[(4096/4)];
//-----------------------------------------------------------------------------
// 65 byte meta-data to be written to NAND
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int write_aux_buffer[65];
//-----------------------------------------------------------------------------
// Descriptor 1: issue NAND write setup command (CLE/ALE)
//-----------------------------------------------------------------------------
write[0].dma_nxtcmdar = &write[1];                          // point to the next descriptor
write[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5)|   // 1 byte command, 5 byte address
                   BF_APBH_CHn_CMD_CMDWORDS      (3)    |   // send 3 words to the GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |  // wait for command to finish
before
                                                            //    continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                   BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels
from
                                                            //    taking over
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                   BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
```

```
                BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to
NAND
write[0].dma_bar = &nand_cmd_addr_buffer;           // byte 0 write setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
write[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)     | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)   |
                    BF_GPMI_CTRL0_CS              (2)      | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)       | // send command and
address
                    BF_GPMI_CTRL0_XFER_COUNT       (1 + 5);    // 1 byte command, 5 byte
address
write[0].gpmi_compare = NULL;                            // field not used but necessary to
set eccctrl
write[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 2: write the data payload (DATA)
//-------------------------------------------------------------------------
write[1].dma_nxtcmdar = &write[2];                        // point to the next descriptor
write[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (0)|  // NOTE: No DMA data transfer
                   BF_APBH_CHn_CMD_CMDWORDS        (4)|  // send 4 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD     (1)|  // Wait to end
                   BF_APBH_CHn_CMD_SEMAPHORE       (0)|
                   BF_APBH_CHn_CMD_NANDWAIT4READY  (0)|
                   BF_APBH_CHn_CMD_NANDLOCK        (1)|  // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT      (0)|
                   BF_APBH_CHn_CMD_CHAIN           (1)|  // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_NO_XFER);  // No data transferred
write[1].dma_bar = &write_payload_buffer;                       // pointer for the 4K byte
data area
// 4 words sent to the GPMI
write[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)     | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)   |
                    BF_GPMI_CTRL0_CS                (2)       | // must correspond to NAND
CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)|
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (0);        // NOTE: this field
contains
                                                                  //  the total amount
                             // DMA transferred to GPMI via DMA (0)!
write[1].gpmi_compare = NULL;                                  // field not used but necessary
to
set eccctrl
write[1].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    ENCODE_8_BIT) | // specify t = 8
mode
                        BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       | // enable ECC module
                     BF_GPMI_ECCCTRL_BUFFER_MASK   (0x1FF);          // write all 8 data
blocks
                                                                    //   and 1 aux block
write[1].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);      // specify number of bytes
                                                                //   written to NAND
write[1].gpmi_data_pointer = &write_payload_pointer;            // data buffer address
write[1].gpmi_aux_pointer  = &write_aux_pointer;                // metadata pointer
//-------------------------------------------------------------------------
// Descriptor 3: issue NAND write execute command (CLE)
//-------------------------------------------------------------------------
write[2].dma_nxtcmdar = &write[3];                          // point to the next descriptor
write[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1) |     // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3) |     // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |     // wait for command to finish
before
                                                          //    continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0) |
                   BF_APBH_CHn_CMD_NANDLOCK      (1) |     // maintain resource lock
```

```
                    BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                    BF_APBH_CHn_CMD_CHAIN         (1) |     // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);   // read data from DMA, write to
NAND
write[2].dma_bar = &nand_cmd_addr_buffer[6];             // point to byte 6, write execute
command
// 3 words sent to the GPMI
write[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                    BF_GPMI_CTRL0_CS               (2)      | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);       // 1 byte command
write[2].gpmi_compare = NULL;                           // field not used but necessary to set
eccctrl
write[2].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 4: wait for ready (CLE)
//-------------------------------------------------------------------------
write[3].dma_nxtcmdar = &write[4];                       // point to the next descriptor

write[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0) |   // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (1) |   // send 1 word to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |   // wait for command to finish before
                                                         //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(1) |   // wait for nand to be ready
                   BF_APBH_CHn_CMD_NANDLOCK      (0) |   // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                   BF_APBH_CHn_CMD_CHAIN         (1) |   // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);          // no dma transfer
write[3].dma_bar = NULL;                                                // field not used
// 1 word sent to the GPMI
write[3].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND
ready
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)         |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)      |
                  BF_GPMI_CTRL0_CS              (2)         | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)     |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)            |
                      BF_GPMI_CTRL0_XFER_COUNT         (0);
//-------------------------------------------------------------------------
// Descriptor 5: psense compare (time out check)
//-------------------------------------------------------------------------
write[4].dma_nxtcmdar = &write[5];                         // point to the next descriptor
write[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)       | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)       | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)       | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)       |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)       |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)       |
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)       |
                   BF_APBH_CHn_CMD_CHAIN         (1)       | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[4].dma_bar = dma_error_handler;                // if sense check fails, branch to error
handler
//-------------------------------------------------------------------------
// Descriptor 6: issue NAND status command (CLE)
//-------------------------------------------------------------------------
write[5].dma_nxtcmdar = &write[6];                       // point to the next descriptor
write[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)    |  // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3)    |  // send 3 words to the GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |  // wait for command to finish
before
continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
```

```
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    |  // prevent other DMA channels from
taking over
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                   BF_APBH_CHn_CMD_CHAIN         (1)    |  // follow chain to next command
                 BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to
NAND
write[5].dma_bar = &nand_cmd_addr_buffer[7];                // point to byte 7, status
command
write[5].gpmi_compare = NULL;                             // field not used but necessary to set
eccctrl
write[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
// 3 words sent to the GPMI
write[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                    BF_GPMI_CTRL0_CS               (2)     | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);       // 1 byte command
//-----------------------------------------------------------------------------
// Descriptor 7: read status and compare (DATA)
//-----------------------------------------------------------------------------
write[6].dma_nxtcmdar = &write[7];                          // point to the next descriptor
write[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)  |    // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (2)  |    // send 2 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)  |    // wait for command to finish
before
                                                          //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)  |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)  |
                   BF_APBH_CHn_CMD_NANDLOCK      (1)  |    // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)  |
                   BF_APBH_CHn_CMD_CHAIN         (1)  |    // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
write[6].dma_bar = NULL;                                    // field not used
// 2 word sent to the GPMI
write[6].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ_AND_COMPARE) | // read from the
                                                                  //       NAND and
//                                                               // compare to expect
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                    BF_GPMI_CTRL0_CS             (2)       |   // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)        |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);
write[6].gpmi_compare = <MASK_AND_REFERENCE_VALUE>;  // NOTE: mask and reference values are
NAND
                                                          // SPECIFIC to evaluate the NAND
status
//-----------------------------------------------------------------------------
// Descriptor 8: psense compare (time out check)
//-----------------------------------------------------------------------------
write[7].dma_nxtcmdar = &write[8];                            // point to the next descriptor
write[7].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)        | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)        | // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                   BF_APBH_CHn_CMD_CHAIN         (1)        | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[7].dma_bar = dma_error_handler;             // if sense check fails, branch to error
handler
//-----------------------------------------------------------------------------
// Descriptor 9: emit GPMI interrupt
//-----------------------------------------------------------------------------
write[8].dma_nxtcmdar = NULL;                                 // not used since this is
```

```
last
descriptor
write[8].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT     (0)          | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS       (0)          | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD    (0)          | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE      (0)          |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)           |
                   BF_APBH_CHn_CMD_NANDLOCK       (0)          |
                   BF_APBH_CHn_CMD_IRQONCMPLT     (1)          | // emit GPMI interrupt
                   BF_APBH_CHn_CMD_CHAIN          (0)          | // terminate DMA chain
processing
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
```

17.4.1.2 Using the BCH Encoder

To use the BCH encoder, first turn off the module-wide soft reset bit in both the GPMI and BCH blocks before starting any DMA activity.

Turning off the soft reset must take place by itself, prior to programming the rest of the control registers. Turn off the BCH bus master soft reset bit. Turn off the clock gate bits.

Program the remainder of the GPMI, BCH and APBH DMA as follows:

```
// bring APBH out of reset
APBH_CTRL0_CLR(BM_APBH_CTRL0_SFRST);
APBH_CTRL0_CLR(BM_APBH_CTRL0_CLKGATE);

// bring BCH out of reset
BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);

// bring gpmi out of reset
GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
GPMI_CTRL1_SET(BM_GPMI_CTRL1_DEV_RESET | // deassert reset
                   BM_GPMI_CTRL1_BCH_MODE  );    // enable BCH mode

// enable pinctrl
PINCTRL_CTRL_WR(0x00000000);

// enable gpmi pins
PINCTRL_MUXSEL0_CLR(0x0000ffff);    // data bits
PINCTRL_MUXSEL1_CLR(0x03ffffff);    // control bits
PINCTRL_MUXSEL8_CLR(0x0003f3ff);    // control bits
PINCTRL_MUXSEL8_SET(0x00015155);    // control bits
```

Note that for writing NANDs (ECC encoding), only GPMI DMA command complete interrupts are used. The BCH engine is used for writing to the NAND but may optionally produces an interrupt. From the sample code in DMA Structure Code Example:

- DMA descriptor 1 prepares the NAND for data write by using the GPMI to issue a write setup command byte under CLE, then sends a 5-byte address under ALE. The BCH engine is disabled and not used for these commands.

- DMA descriptor 2 enables the BCH engine for encoding to begin the initial writing of the NAND data by specifying where the data and auxiliary payload are coming from in system memory.
- DMA descriptor 3 issues the write commit command byte under CLE to the NAND.
- DMA descriptor 4 waits for the NAND to complete the write commit/transfer by watching the NAND's ready line status. This descriptor relinquishes the NANDLOCK on the GPMI to enable the other DMA channels to initiate NAND transactions on different NAND CS lines.
- DMA descriptor 6 issues a NAND status command byte under "CLE" to check the status of the NAND device following the page write.
- DMA descriptor 7 reads back the NAND status and compares the status with an expected value. If there are differences, then the DMA processing engine follows an error-handling DMA descriptor path.
- DMA descriptor 8 disables the BCH engine and emits a GPMI interrupt to indicate that the NAND write has been completed.

17.4.2 BCH Decoding for NAND Reads

When a page is read from NAND flash, BCH syndromes will be computed and, if correctable errors are found, they will be corrected on a per block basis within the NAND page.

This decoding process is fully overlapped with other NAND data reads and with CPU execution. The BCH decoder flowchart in the figure below shows the steps involved in programming the decoder. The hardware flow of reading and decoding a 4096-byte page is shown in Figure 17-10.

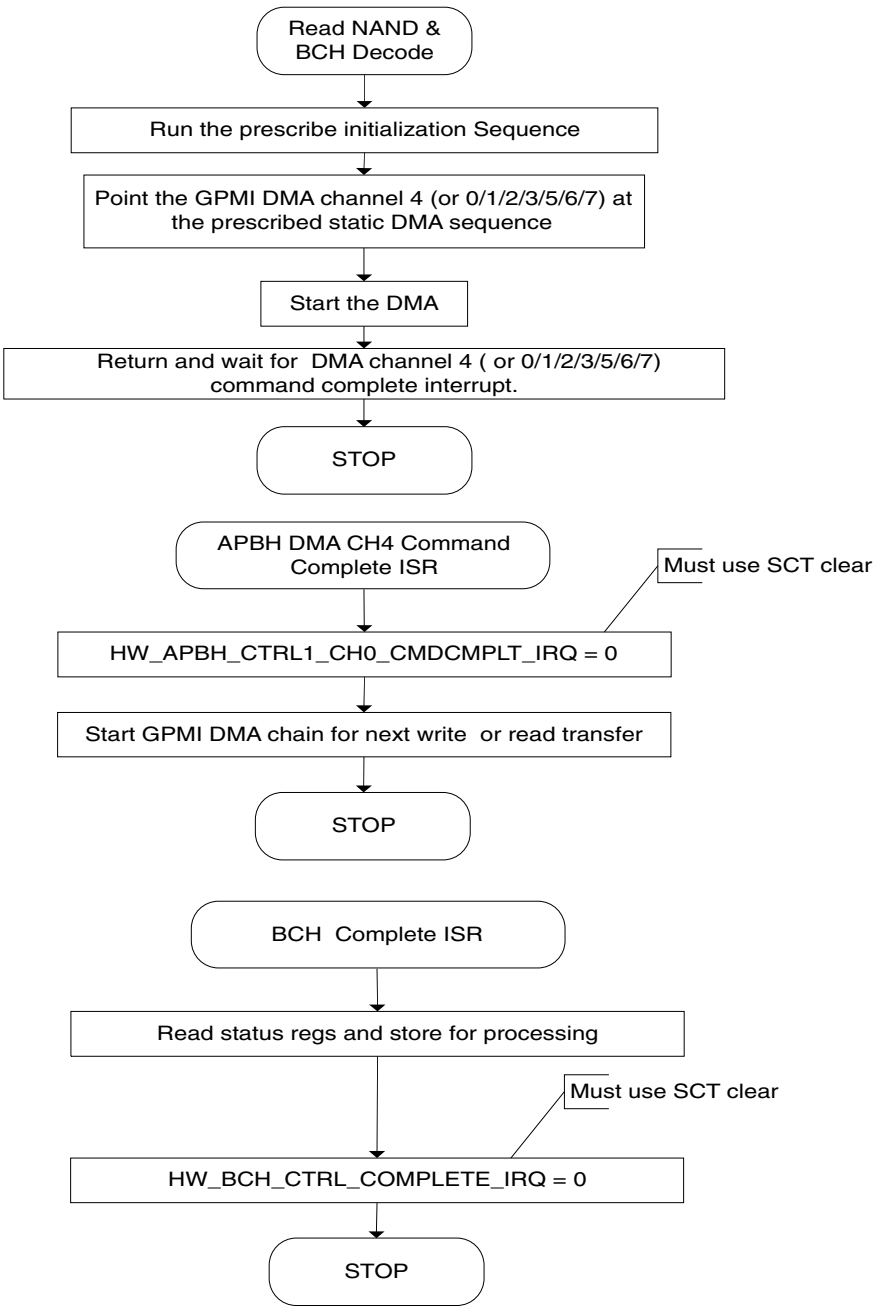


Figure 17-9. BCH Decode Flowchart

Conceptually, an APHB DMA Channel command chain with seven command structures linked together is used to perform the BCH decode operation (as shown in Figure 17-10).

Note

The GPMI's DMA command structures controls the BCH decode operation.

To use the BCH decoder with the GPMI's DMA, create a DMA command chain containing seven descriptor structures, as shown in the figure below and detailed in the DMA structure code example that follows it in DMA Structure Code Example. The seven DMA descriptors perform the following tasks:

1. Issue NAND read setup command byte (under "CLE") and address bytes (under "ALE").
2. Issue NAND read execute command byte (under "CLE").
3. Wait for the NAND device to complete accessing the block data by watching the ready signal.
4. Check for NAND timeout through "PSENSE".
5. Configure and enable the BCH block and read the NAND block data.
6. Disable the BCH block.
7. Descriptor NOP to allow NANDLOCK in the previous descriptor to the thread-safe.

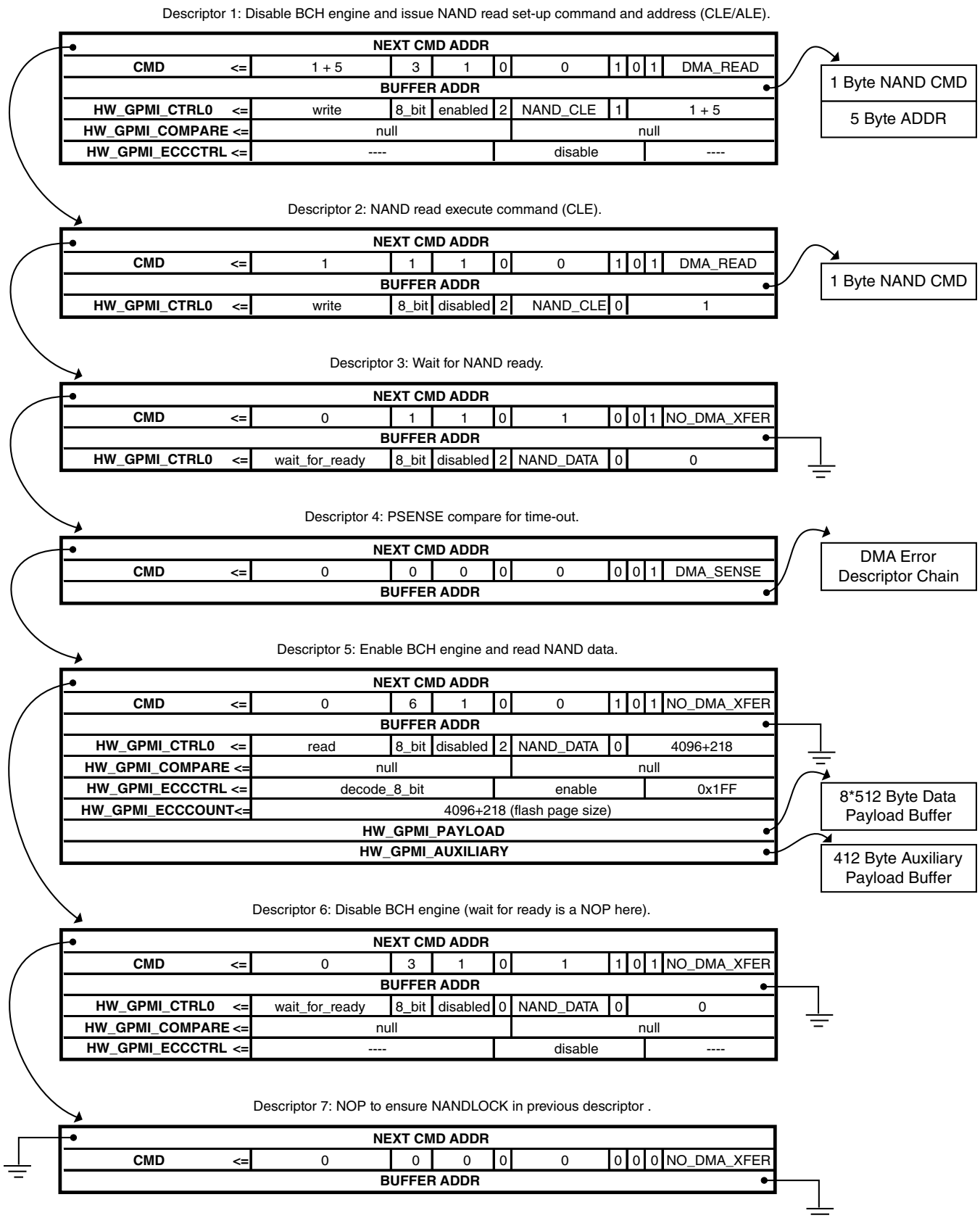


Figure 17-10. BCH Decode DMA Descriptor Chain

17.4.2.1 DMA Structure Code Example

The following sample code illustrates the coding for one read transaction, consisting of a seven DMA command structure chain for reading all 4096 bytes of payload data (eight 512-byte blocks) and 65 bytes of metadata with the associative parity bytes (8 * (18) + 9) from a 4K NAND page sitting on GPMI CS2.

```
//-----------------------------------------------------------------------------
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//-----------------------------------------------------------------------------
// allocate 7 descriptors for doing a NAND ECC Read
//-----------------------------------------------------------------------------
GENERIC_DESCRIPTOR read[7];
//-----------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//-----------------------------------------------------------------------------
unsigned int * dma_error_handler;
//-----------------------------------------------------------------------------
// 7 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is read setup command
//   bytes 1-5 is the NAND address
//   byte 6 is read execute command
//-----------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[7];
//-----------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int read_payload_buffer[(4096/4)];
//-----------------------------------------------------------------------------
// 412 byte auxiliary buffer used for reads
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int read_aux_buffer[(412/4)];
//-----------------------------------------------------------------------------
// Descriptor 1: issue NAND read setup command (CLE/ALE)
//-----------------------------------------------------------------------------
read[0].dma_nxtcmdar = &read[1];                          // point to the next descriptor
read[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5) |  // 1 byte command, 5 byte address
                  BF_APBH_CHn_CMD_CMDWORDS      (3)     |  // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)     |  // wait for command to finish
                                                           //     before continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)     |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)     |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)     |  // prevent other DMA channels from
                                                           //    taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)     |
```

```
                BF_APBH_CHn_CMD_CHAIN         (1)      |   // follow chain to next command
                BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to
NAND
read[0].dma_bar = &nand_cmd_addr_buffer;            // byte 0 read setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
read[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                     BF_GPMI_CTRL0_CS                 (2)       | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)       | // send command and address
                     BF_GPMI_CTRL0_XFER_COUNT         (1 + 5);    // 1 byte command, 5 byte
address
read[0].gpmi_compare = NULL;                        // field not used but necessary to set
eccctrl
read[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 2: issue NAND read execute command (CLE)
//-------------------------------------------------------------------------
read[1].dma_nxtcmdar = &read[2];                          // point to the next descriptor
read[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)    |   // 1 byte read command
                  BF_APBH_CHn_CMD_CMDWORDS      (1)    |   // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |   // wait for command to finish
before
                                                        //    continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels from
                                                        //     taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                  BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);     // read data from DMA, write
to NAND
read[1].dma_bar = &nand_cmd_addr_buffer[6];              // point to byte 6, read execute
command
// 1 word sent to the GPMI
read[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED) |
                     BF_GPMI_CTRL0_CS                 (2)       | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                     BF_GPMI_CTRL0_XFER_COUNT         (1);        // 1 byte command
//-------------------------------------------------------------------------
// Descriptor 3: wait for ready (DATA)
//-------------------------------------------------------------------------
read[2].dma_nxtcmdar = &read[3];                          // point to the next descriptor
read[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)      | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (1)      | // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)      | // wait for command to finish
before
                                                        //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)      |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)      | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (0)      | // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)      |
                  BF_APBH_CHn_CMD_CHAIN         (1)      | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);   // no dma transfer
read[2].dma_bar = NULL;                                    // field not used
// 1 word sent to the GPMI
read[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND
ready
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)          |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)       |
                     BF_GPMI_CTRL0_CS                 (2)             | // must correspond
to NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)      |
```

```
                    BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)             |
                    BF_GPMI_CTRL0_XFER_COUNT         (0);
//-----------------------------------------------------------------------------
// Descriptor 4: psense compare (time out check)
//-----------------------------------------------------------------------------
read[3].dma_nxtcmdar = &read[4];                                     // point to the next
descriptor
read[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        |      // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)        |      // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        |      // do not wait to continue
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)        |
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                  BF_APBH_CHn_CMD_CHAIN         (1)        |      // follow chain to next
command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);      // perform a sense check
read[3].dma_bar = dma_error_handler;                        // if sense check fails, branch to
error handler
//-----------------------------------------------------------------------------
// Descriptor 5: read 4K page plus 65 byte meta-data Nand data
//               and send it to ECC block (DATA)
//-----------------------------------------------------------------------------
read[4].dma_nxtcmdar = &read[5];                               // point to the next descriptor
read[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)    | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (6)    | // send 6 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    | // wait for command to finish before
                                                         //       continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    | // prevent other DMA channels from
taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)    | // ECC block generates BCH interrupt
                                                         //      on completion
                  BF_APBH_CHn_CMD_CHAIN         (1)    | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);    // no DMA transfer,
                                                                 // ECC block handles
transfer
read[4].dma_bar = NULL;                                          // field not used
// 6 words sent to the GPMI
read[4].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)       |  // read from the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)      |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)   |
                     BF_GPMI_CTRL0_CS                 (2)         |  // must correspond to
NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)         |
                     BF_GPMI_CTRL0_XFER_COUNT         (4096+218);    // eight 512 byte data
blocks
                                                                     // metadata, and parity

read[4].gpmi_compare = NULL;                                  // field not used but necessary to set
eccctrl
// GPMI ECCCTRL PIO This launches the 4K byte transfer through BCH's
// bus master. Setting the ECC_ENABLE bit redirects the data flow
// within the GPMI so that read data flows to the BCH engine instead
// of flowing to the GPMI's DMA channel.
read[4].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    DECODE_8_BIT) |    // specify t = 8
mode
                       BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       |    // enable ECC
module
                       BF_GPMI_ECCCTRL_BUFFER_MASK     (0X1FF);  // read all 8 data blocks
and 1 aux block
read[4].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);        // specify number of bytes
                                                                 //        read from NAND
read[4].gpmi_data_ptr = &read_payload_buffer;                    // pointer for the 4K byte
                                                                 //    data area
read[4].gpmi_aux_ptr = &read_aux_buffer;                         // pointer for the 65 byte
aux area +
                                                                 // parity and syndrome
```

```
bytes for both
                                                                  // data and aux blocks.
//-------------------------------------------------------------------------
// Descriptor 6: disable ECC block
//-------------------------------------------------------------------------
read[5].dma_nxtcmdar = &read[6];                                  // point to the next descriptor
read[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)       | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (3)       | // send 3 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)       | // wait for command to finish
before
                                                            //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)       |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)       | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (1)       | // need nand lock to be
                                                            // thread safe while turn-off BCH
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)       |
                  BF_APBH_CHn_CMD_CHAIN         (1)       | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[5].dma_bar = NULL;                                           // field not used
// 3 words sent to the GPMI
read[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)       |
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)      |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)   |
                     BF_GPMI_CTRL0_CS                 (2)         | // must correspond to
NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)         |
                     BF_GPMI_CTRL0_XFER_COUNT         (0);
read[5].gpmi_compare = NULL;                                      // field not used but necessary to set
eccctrl
read[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE);  // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 7: deassert nand lock
//-------------------------------------------------------------------------
read[6].dma_nxtcmdar = NULL;                                      // not used since this is last
descriptor
read[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)    |    // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)    |    // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)    |    // wait for command to finish
before
                                                            //        continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)    |    // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |    // BCH engine generates interrupt
                  BF_APBH_CHn_CMD_CHAIN         (0)    |    // terminate DMA chain processing
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[6].dma_bar = NULL;                                           // field not used
```

17.4.2.2 Using the Decoder

As illustrated in Figure 17-10 and the sample code in DMA Structure Code Example :

- DMA descriptor 1 prepares the NAND for data read by using the GPMI to issue a NAND read setup command byte under CLE, then sends a 5-byte address under ALE. The BCH engine is not used for these commands.
- DMA descriptor 2 issues a one-byte read execute command to the NAND device that triggers its read access. The NAND then goes not ready.

- DMA descriptor 3 performs a wait for ready operation allowing the DMA chain to remain dormant until the NAND device completes its read access time.
- DMA descriptor 5 handles the reading and error correction of the NAND data. This command's PIOs activate the BCH engine to write the read NAND data to system memory and to process it for any errors that need to be corrected. This DMA descriptor contains two PIO values that are system memory addresses pointing to the PAYLOAD data area and to the AUXILIARY data area. These addresses are used by the BCH engine's AHB master to move data into system memory and to correct it. While this example is reading an entire 4K page—payload plus metadata—it is equally possible to read just one 512-byte payload block or just the uniquely protected metadata block in a single 7 DMA structure transfer.
- DMA descriptor 6 disables the BCH engine with the NANDLOCK asserted. This is necessary to ensure that the GPMI resource is not arbitrated to another DMA channel when multiple DMA channels are active concurrently.
- DMA descriptor 7 deasserts the NANDLOCK to free up the GPMI resource to another channel.

As the BCH block receives data from the GPMI:

- The decoder transforms the read NAND data block into a BCH code word and computes the codeword syndrome.
- If no errors are present, then the BCH block can immediately report back to firmware. This report is passed as the BCH_CTRL_COMPLETE_IRQ interrupt status bit and the associated status registers in BCH_STATUS0/1 registers.
- If an error is present, then the BCH block corrects the necessary data block or parity block bytes, if possible (not all errors are correctable).

As the BCH decoder reads the data and parity blocks, it records a special condition, i.e., that all of the bits of a payload data block or metadata block are one, including any associated parity bytes. The all-ones case for both parity and data indicates an erased block in the NAND device.

The BCH_STATUS0 register contains a 4-bit field that indicates the final status of the auxiliary block. A value of 0x0 indicates no errors found for a block.

- A value of 1 to 20 inclusive indicates that many correctable errors were found and fixed.
- A value of 0xFE indicates uncorrectable errors detected on the block.

- A value of 0xFF indicates that the block was in the special ALL ONES state and is therefore considered to be an ERASED block.
- All other values are disallowed by the hardware design.

Recall that up to eight NAND devices can have DMA chains in-flight at once, i.e. they can all be contending for access to the GPMI data bus. It is impossible to predict which NAND device will enter the BCH engine with a transfer first, because each chain includes a wait4ready command structure. As a result, firmware should look at the BCH_STATUS0_COMPLETED_CE bit field to determine which block is being reported in the status register. There is also a 16-bit HANDLE field in the GPMI_ECCCTRL register that is passed down the pipeline with each transaction. This handle field can be used to speed firmware's detection of which transaction is being reported.

These examples of reading and writing have focused on full page transfers of 4K page NAND devices. Other device configurations can be specified by changing the ECCOUNT field in the GPMI registers and reprogramming the BCH's FLASHnLAYOUTm registers.

The BCH and GPMI blocks are designed to be very efficient at reading single 512 (or 1024)-byte pages in one transaction. With no errors, the transaction takes less than 20 HCLKs longer than the time to read the raw data from the NAND.

To summarize, the APBH DMA command chain for a BCH decode operation is shown in Figure 17-10. Seven DMA command structures must be present for each NAND read transaction decoded by the BCH. The seven DMA command structures for multiple NAND read transaction blocks can be chained together to make larger units of work for the BCH, and each will produce an appropriate error report in the BCH PIO space. Multiple NAND devices can have such multiple chains scheduled. The results can come back out of order with respect to the multiple chains.

17.4.3 Interrupts

There are two interrupt sources used in processing BCH protected NAND read and write transfers.

Since all BCH operations are initiated by GPMI DMA command structures, the DMA completion interrupt for the GPMI is an important ISR. Both of the flow charts of Figure 17-6 and Figure 17-9 show the GPMI DMA complete ISR skeleton. In both reads and writes, the GPMI DMA completion interrupt is used to schedule work ***INTO*** the error correction pipeline. As the front end processing completes, the DMA interrupt is

generated and additional work, such as DMA chains, are passed to the GPMI DMA to keep it *fed*. For write operations, this is the only interrupt that is generated for processing the NAND write transfer.

For reads, however, two interrupts are needed. Every read is started by a GPMI DMA command chain and the front end queue is fed as described above. The back end of the read pipeline is drained by monitoring the BCH completion interrupt found int HW_BCH_CTRL_COMPLETE_IRQ.

An BCH transaction consists of reading or writing all of the blocks requested in the HW_GPMI_ECCCTRL_BUFFER_MASK bit field. As every read transaction completes, it posts the status of all of the blocks to the HW_BCH_STATUS0 and HW_BCH_STATUS1 registers and sets the completion interrupt. The five stages of the BCH read pipeline completes, one in the GPMI and four in the BCH, are independently stalled as they complete and try to deliver to the next stage in the data flow. Several of these stages can be skipped if no-errors are found or once an uncorrectable error is found in a block.

In any case, the final stage will stall if the status register is busy waiting for the CPU to take status register results. The hardware monitors the state of the HW_BCH_CTRL_COMPLETE_IRQ bit. If it is still set when the last pipeline stage is ready to post data, then the stage will stall. It follows that the next previous stage will stall when it is ready to hand off work to the final stage, and so on up the pipeline.

CAUTION

It is important that firmware read the STATUS0/1 results and save them before clearing the interrupt request bit. Otherwise, a transaction and its results could be completely lost.

17.5 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST.

The reset process gates the clocks automatically. The exemplary code is shown below.

```
// A soft reset can take multiple clocks to complete, so do NOT gate the
// clock when setting soft reset. The reset process will gate the clock
// automatically. Poll until this has happened before subsequently
// preparing soft-reset and clock gate
  BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
  BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
// asserting soft-reset
  BCH_CTRL_SET(BM_BCH_CTRL_SFTRST);
// waiting for confirmation of soft-reset
  while (!BCH_CTRL.B.CLKGATE)
```

```
   {
// busy wait
   }
// Done.
   BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
   BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
```

17.6 BCH Memory Map/Register Definition

BCH Hardware Register Format Summary

BCH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_8000	Hardware BCH ECC Accelerator Control Register (BCH_CTRL)	32	R/W	C000_0000h	17.6.1/578
180_8004	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_SET)	32	R/W	C000_0000h	17.6.1/578
180_8008	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_CLR)	32	R/W	C000_0000h	17.6.1/578
180_800C	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_TOG)	32	R/W	C000_0000h	17.6.1/578
180_8010	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0)	32	R	0000_0010h	17.6.2/580
180_8014	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0_SET)	32	R	0000_0010h	17.6.2/580
180_8018	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0_CLR)	32	R	0000_0010h	17.6.2/580
180_801C	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0_TOG)	32	R	0000_0010h	17.6.2/580
180_8020	Hardware ECC Accelerator Mode Register (BCH_MODE)	32	R/W	0000_0000h	17.6.3/582
180_8024	Hardware ECC Accelerator Mode Register (BCH_MODE_SET)	32	R/W	0000_0000h	17.6.3/582
180_8028	Hardware ECC Accelerator Mode Register (BCH_MODE_CLR)	32	R/W	0000_0000h	17.6.3/582
180_802C	Hardware ECC Accelerator Mode Register (BCH_MODE_TOG)	32	R/W	0000_0000h	17.6.3/582
180_8030	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR)	32	R/W	0000_0000h	17.6.4/583
180_8034	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR_SET)	32	R/W	0000_0000h	17.6.4/583
180_8038	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR_CLR)	32	R/W	0000_0000h	17.6.4/583
180_803C	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR_TOG)	32	R/W	0000_0000h	17.6.4/583

Table continues on the next page...

BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_8040	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR)	32	R/W	0000_0000h	17.6.5/583
180_8044	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR_SET)	32	R/W	0000_0000h	17.6.5/583
180_8048	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR_CLR)	32	R/W	0000_0000h	17.6.5/583
180_804C	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR_TOG)	32	R/W	0000_0000h	17.6.5/583
180_8050	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR)	32	R/W	0000_0000h	17.6.6/584
180_8054	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR_SET)	32	R/W	0000_0000h	17.6.6/584
180_8058	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR_CLR)	32	R/W	0000_0000h	17.6.6/584
180_805C	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR_TOG)	32	R/W	0000_0000h	17.6.6/584
180_8070	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT)	32	R/W	E4E4_E4E4h	17.6.7/584
180_8074	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT_SET)	32	R/W	E4E4_E4E4h	17.6.7/584
180_8078	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT_CLR)	32	R/W	E4E4_E4E4h	17.6.7/584
180_807C	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT_TOG)	32	R/W	E4E4_E4E4h	17.6.7/584
180_8080	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0)	32	R/W	070A_4080h	17.6.8/585
180_8084	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0_SET)	32	R/W	070A_4080h	17.6.8/585
180_8088	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0_CLR)	32	R/W	070A_4080h	17.6.8/585
180_808C	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0_TOG)	32	R/W	070A_4080h	17.6.8/585
180_8090	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1)	32	R/W	10DA_4080h	17.6.9/587
180_8094	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1_SET)	32	R/W	10DA_4080h	17.6.9/587
180_8098	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1_CLR)	32	R/W	10DA_4080h	17.6.9/587
180_809C	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1_TOG)	32	R/W	10DA_4080h	17.6.9/587
180_80A0	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0)	32	R/W	070A_4080h	17.6.10/ 588
180_80A4	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0_SET)	32	R/W	070A_4080h	17.6.10/ 588

Table continues on the next page...

BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_80A8	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0_CLR)	32	R/W	070A_4080h	17.6.10/ 588
180_80AC	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0_TOG)	32	R/W	070A_4080h	17.6.10/ 588
180_80B0	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1)	32	R/W	10DA_4080h	17.6.11/ 590
180_80B4	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1_SET)	32	R/W	10DA_4080h	17.6.11/ 590
180_80B8	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1_CLR)	32	R/W	10DA_4080h	17.6.11/ 590
180_80BC	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1_TOG)	32	R/W	10DA_4080h	17.6.11/ 590
180_80C0	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0)	32	R/W	070A_4080h	17.6.12/ 591
180_80C4	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0_SET)	32	R/W	070A_4080h	17.6.12/ 591
180_80C8	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0_CLR)	32	R/W	070A_4080h	17.6.12/ 591
180_80CC	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0_TOG)	32	R/W	070A_4080h	17.6.12/ 591
180_80D0	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1)	32	R/W	10DA_4080h	17.6.13/ 593
180_80D4	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1_SET)	32	R/W	10DA_4080h	17.6.13/ 593
180_80D8	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1_CLR)	32	R/W	10DA_4080h	17.6.13/ 593
180_80DC	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1_TOG)	32	R/W	10DA_4080h	17.6.13/ 593
180_80E0	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0)	32	R/W	070A_4080h	17.6.14/ 594
180_80E4	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0_SET)	32	R/W	070A_4080h	17.6.14/ 594
180_80E8	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0_CLR)	32	R/W	070A_4080h	17.6.14/ 594
180_80EC	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0_TOG)	32	R/W	070A_4080h	17.6.14/ 594
180_80F0	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1)	32	R/W	10DA_4080h	17.6.15/ 596
180_80F4	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1_SET)	32	R/W	10DA_4080h	17.6.15/ 596
180_80F8	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1_CLR)	32	R/W	10DA_4080h	17.6.15/ 596
180_80FC	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1_TOG)	32	R/W	10DA_4080h	17.6.15/ 596

Table continues on the next page...

BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_8100	Hardware BCH ECC Debug Register0 (BCH_DEBUG0)	32	R/W	0000_0000h	17.6.16/ 597
180_8104	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_SET)	32	R/W	0000_0000h	17.6.16/ 597
180_8108	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_CLR)	32	R/W	0000_0000h	17.6.16/ 597
180_810C	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_TOG)	32	R/W	0000_0000h	17.6.16/ 597
180_8110	KES Debug Read Register (BCH_DBGKESREAD)	32	R	0000_0000h	17.6.17/ 599
180_8114	KES Debug Read Register (BCH_DBGKESREAD_SET)	32	R	0000_0000h	17.6.17/ 599
180_8118	KES Debug Read Register (BCH_DBGKESREAD_CLR)	32	R	0000_0000h	17.6.17/ 599
180_811C	KES Debug Read Register (BCH_DBGKESREAD_TOG)	32	R	0000_0000h	17.6.17/ 599
180_8120	Chien Search Debug Read Register (BCH_DBGCSFEREAD)	32	R	0000_0000h	17.6.18/ 600
180_8124	Chien Search Debug Read Register (BCH_DBGCSFEREAD_SET)	32	R	0000_0000h	17.6.18/ 600
180_8128	Chien Search Debug Read Register (BCH_DBGCSFEREAD_CLR)	32	R	0000_0000h	17.6.18/ 600
180_812C	Chien Search Debug Read Register (BCH_DBGCSFEREAD_TOG)	32	R	0000_0000h	17.6.18/ 600
180_8130	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD)	32	R	0000_0000h	17.6.19/ 600
180_8134	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD_SET)	32	R	0000_0000h	17.6.19/ 600
180_8138	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD_CLR)	32	R	0000_0000h	17.6.19/ 600
180_813C	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD_TOG)	32	R	0000_0000h	17.6.19/ 600
180_8140	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD)	32	R	0000_0000h	17.6.20/ 601
180_8144	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD_SET)	32	R	0000_0000h	17.6.20/ 601
180_8148	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD_CLR)	32	R	0000_0000h	17.6.20/ 601
180_814C	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD_TOG)	32	R	0000_0000h	17.6.20/ 601
180_8150	Block Name Register (BCH_BLOCKNAME)	32	R	2048_4342h	17.6.21/ 601
180_8154	Block Name Register (BCH_BLOCKNAME_SET)	32	R	2048_4342h	17.6.21/ 601

Table continues on the next page...

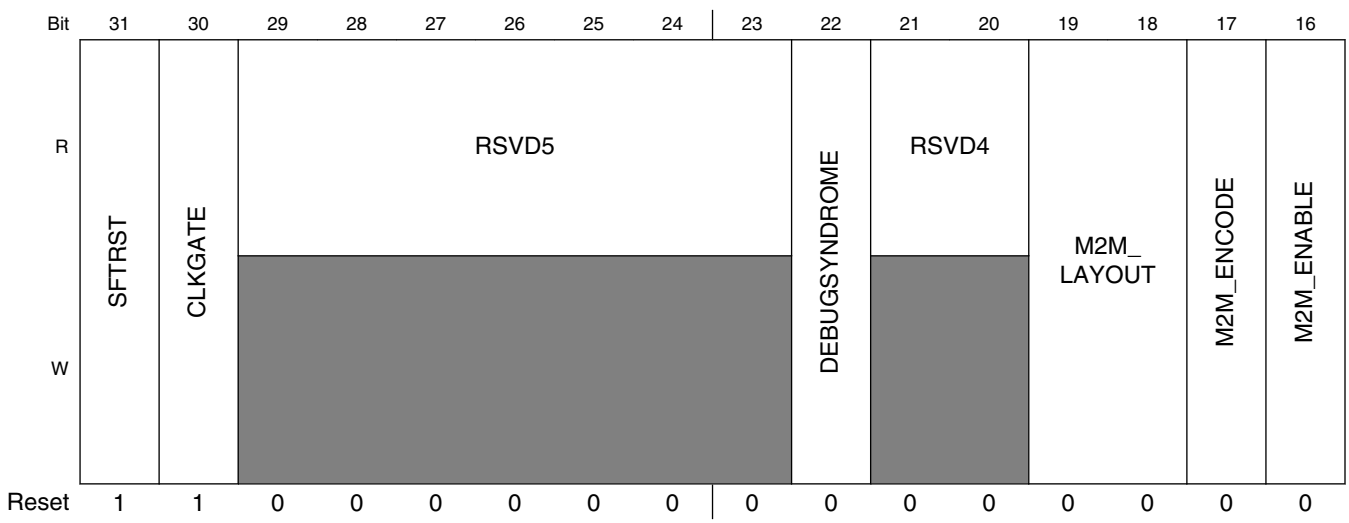
BCH memory map (continued)

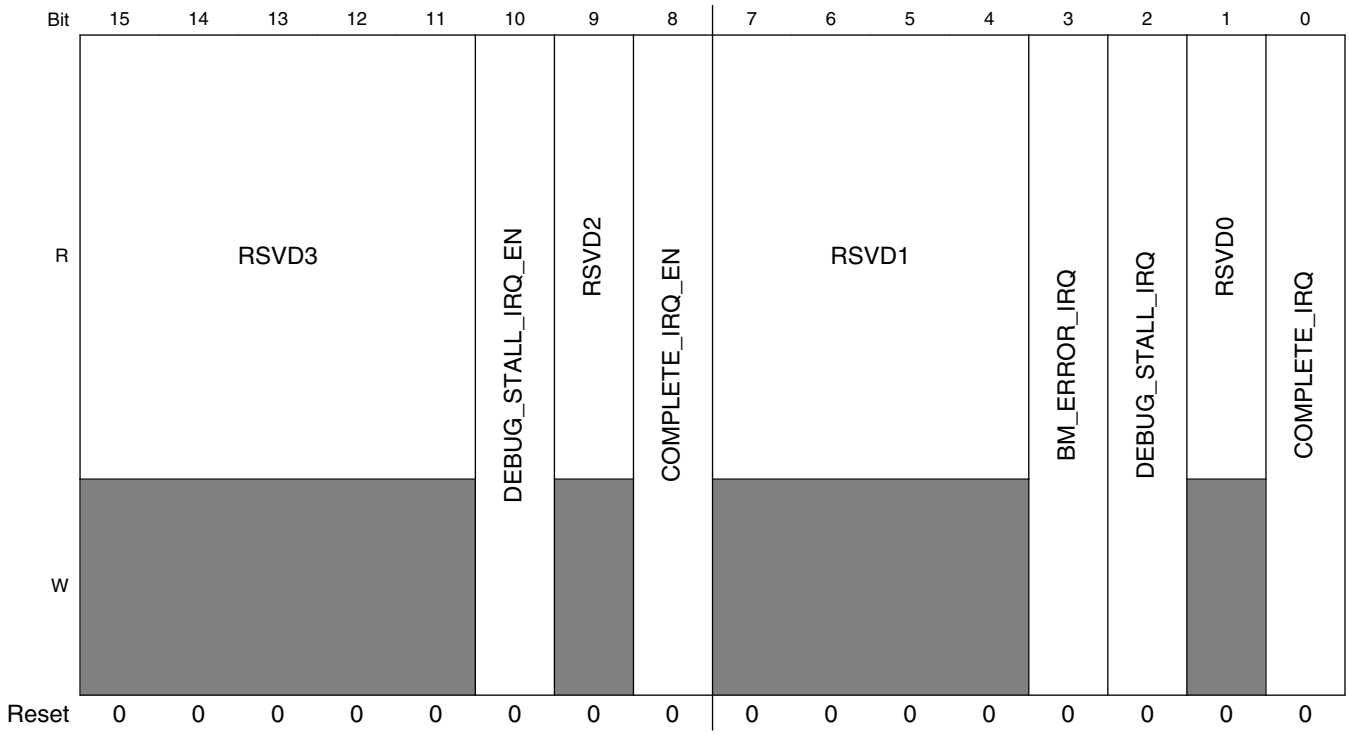
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_8158	Block Name Register (BCH_BLOCKNAME_CLR)	32	R	2048_4342h	17.6.21/ 601
180_815C	Block Name Register (BCH_BLOCKNAME_TOG)	32	R	2048_4342h	17.6.21/ 601
180_8160	BCH Version Register (BCH_VERSION)	32	R	0100_0000h	17.6.22/ 602
180_8164	BCH Version Register (BCH_VERSION_SET)	32	R	0100_0000h	17.6.22/ 602
180_8168	BCH Version Register (BCH_VERSION_CLR)	32	R	0100_0000h	17.6.22/ 602
180_816C	BCH Version Register (BCH_VERSION_TOG)	32	R	0100_0000h	17.6.22/ 602
180_8170	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1)	32	R/W	0000_0000h	17.6.23/ 603
180_8174	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1_SET)	32	R/W	0000_0000h	17.6.23/ 603
180_8178	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1_CLR)	32	R/W	0000_0000h	17.6.23/ 603
180_817C	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1_TOG)	32	R/W	0000_0000h	17.6.23/ 603

17.6.1 Hardware BCH ECC Accelerator Control Register (BCH_CTRL*n*)

The BCH CTRL provides overall control of the hardware ECC accelerator

Address: 180_8000h base + 0h offset + (4d × i), where i=0d to 3d





BCH_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set this bit to 0 to enable normal BCH operation. Set this bit to 1 (default) to disable clocking with the BCH and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the BCH block to its default state. This bit resets all state machines except for the AHB master state machine 0x0 **RUN** — Allow BCH to operate normally. 0x1 **RESET** — Hold BCH in reset.
30 CLKGATE	This bit must be set to 0 for normal operation. When set to 1 it gates off the clocks to the block. 0x0 **RUN** — Allow BCH to operate normally. 0x1 **NO_CLKS** — Do not clock BCH gates in order to minimize power consumption.
29–23 RSVD5	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUGSYNDROME	(For debug purposes only). Enable write of computed syndromes to memory on BCH decode operations. Computed syndromes will be written to the auxiliary buffer after the status block. Syndromes will be written as padded 16-bit values.
21–20 RSVD4	This field is reserved. This read-only field is reserved and always has the value 0
19–18 M2M_LAYOUT	Selects the flash page format for memory-to-memory operations.
17 M2M_ENCODE	Selects encode (parity generation) or decode (correction) mode for memory-to-memory operations.
16 M2M_ENABLE	NOTE! WRITING THIS BIT INITIATES A MEMORY-TO-MEMORY OPERATION. The BCH module must be inactive (not processing data from the GPMI) when this bit is set. The M2M_ENCODE and

Table continues on the next page...

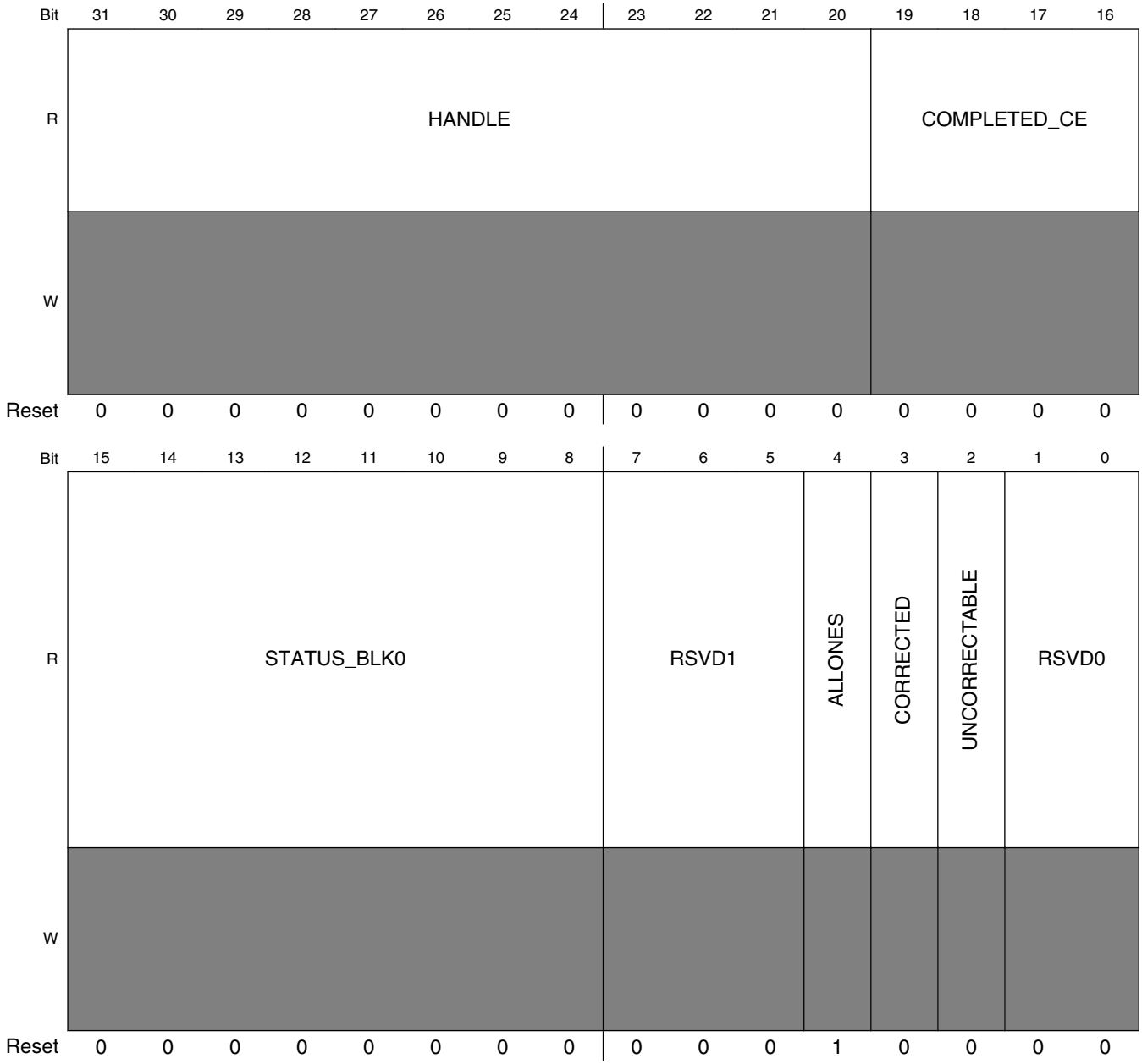
BCH_CTRL*n* field descriptions (continued)

Field	Description
	M2M_LAYOUT bits as well as the ENCODEPTR, DATAPTR, and METAPTR registers are used for memory-to-memory operations and must be correctly programmed before writing this bit.
15–11 RSVD3	This field is reserved. This read-only field is reserved and always has the value 0
10 DEBUG_STALL_ IRQ_EN	1 = interrupt on debug stall mode is enabled. The IRQ is raised on every block
9 RSVD2	This field is reserved. This read-only field is reserved and always has the value 0.
8 COMPLETE_IRQ_ EN	1 = interrupt on completion of correction is enabled.
7–4 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
3 BM_ERROR_IRQ	AHB Bus interface Error Interrupt Status. Write a 1 to the SCT clear address to clear the interrupt status bit.
2 DEBUG_STALL_IRQ	DEBUG STALL Interrupt Status. Write a 1 to the SCT clear address to clear the interrupt status bit.
1 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.
0 COMPLETE_IRQ	This bit indicates the state of the external interrupt line. Write a 1 to the SCT clear address to clear the interrupt status bit. NOTE: subsequent ECC completions will be held off as long as this bit is set. Be sure to read the data from BCH_STATUS0, 1 before clearing this interrupt bit.

17.6.2 Hardware ECC Accelerator Status Register 0 (BCH_STATUS0*n*)

The BCH STAT register provides visibility into the run-time status of the BCH and status information when processing is complete. It provides overall status of the hardware ECC accelerator.

Address: 180_8000h base + 10h offset + (4d × i), where i=0d to 3d



BCH_STATUS0*n* field descriptions

Field	Description
31–20 HANDLE	Software supplies a 12 bit handle for this transfer as part of the GPMI DMA PIO operation that started the transaction. That handle passes down the pipeline and ends up here at the time the BCH interrupt is signaled.
19–16 COMPLETED_CE	This is the chip enable number corresponding to the NAND device from which this data came.
15–8 STATUS_BLK0	Count of symbols in error during processing of first block of flash (metadata block). The number of errors reported will be in the range of 0 to the ECC correction level for block 0. 0x00 **ZERO** — No errors found on block.

Table continues on the next page...

BCH_STATUS0*n* field descriptions (continued)

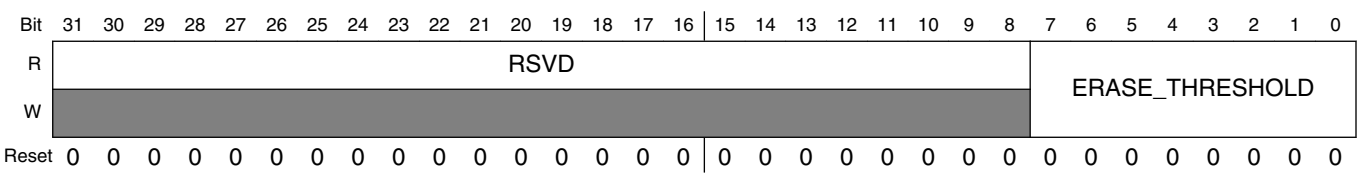
Field	Description
	0x01 **ERROR1** — One error found on block. 0x02 **ERROR2** — One errors found on block. 0x03 **ERROR3** — One errors found on block. 0x04 **ERROR4** — One errors found on block. 0xFE **UNCORRECTABLE** — Block exhibited uncorrectable errors. 0xFF **ERASED** — Page is erased.
7–5 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
4 ALLONES	1 = All data bits of this transaction are ONE.
3 CORRECTED	1 = At least one correctable error encountered during last processing cycle.
2 UNCORRECTABLE	1 = Uncorrectable error encountered during last processing cycle.
RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.

17.6.3 Hardware ECC Accelerator Mode Register (BCH_MODE*n*)

The BCH MODE register provides additional mode controls.

Contains additional global mode controls for the BCH engine.

Address: 180_8000h base + 20h offset + (4d × i), where i=0d to 3d



BCH_MODE*n* field descriptions

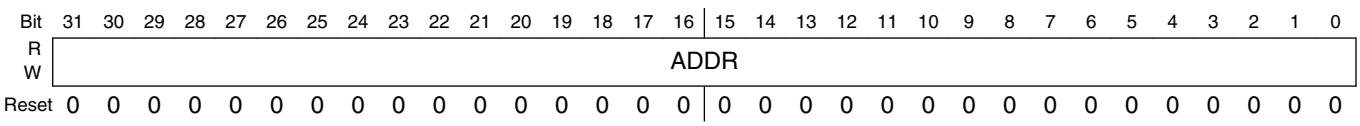
Field	Description
31–8 RSVD	This field is reserved. This read-only field is reserved and always has the value 0.
ERASE_ THRESHOLD	This value indicates the maximum number of zero bits on a flash subpage for it to be considered erased. For SLC NAND devices, this value should be programmed to 0 (meaning that the entire page should consist of bytes of 0xFF. For MLC NAND devices, bit errors may occur on reads (even on blank pages), so this threshold can be used to tune the erased page checking algorithm.

17.6.4 Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR*n*)

When performing memory to memory operations, indicates the address of the encode buffer. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register.

For memory to memory operations, this register is used as the pointer to the encoded data, which is an output when encoding and an input while decoding.

Address: 180_8000h base + 30h offset + (4d × i), where i=0d to 3d



BCH_ENCODEPTR*n* field descriptions

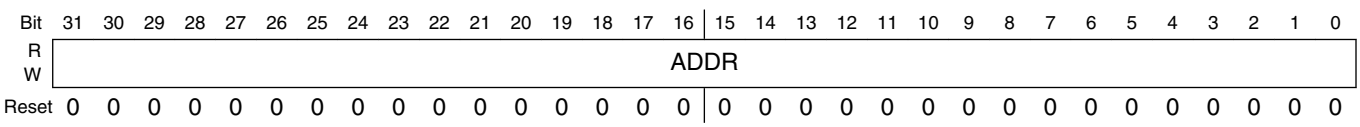
Field	Description
ADDR	Address pointer to encode buffer. This is the source for decode operations and the destination for encode operations. This value must be aligned on a 4 bytes boundary.

17.6.5 Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR*n*)

When performing memory to memory operations, indicates the address of the data buffer.

For memory to memory operations, this register is used as the pointer to the data to encode or the destination buffer for decode operations.

Address: 180_8000h base + 40h offset + (4d × i), where i=0d to 3d



BCH_DATAPTR*n* field descriptions

Field	Description
ADDR	Address pointer to data buffer. This is the source for encode operations and the destination for decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 byte boundary.

17.6.6 Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR*n*)

When performing memory to memory operations, indicates the address of the metadata buffer.

For memory to memory operations, this register is used as the pointer to the metadata to encode or the extracted metadata for decode operations.

Address: 180_8000h base + 50h offset + (4d × i), where i=0d to 3d

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

BCH_METAPTR*n* field descriptions

Field	Description
ADDR	Address pointer to metadata buffer. This is the source for encode metadata read operations and the destination for metadata decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 bytes boundary.

17.6.7 Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT*n*)

The BCH LAYOUTSELECT register provides a mapping of chip selects to layout registers.

When the BCH engine receives a request to process a data block from the GPMI interface, it will use this register to map the incoming chip select to one of the four possible flash layout registers

Address: 180_8000h base + 70h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CS15_SELECT		CS14_SELECT		CS13_SELECT		CS12_SELECT		CS11_SELECT		CS10_SELECT		CS9_SELECT		CS8_SELECT	
Reset	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CS7_SELECT		CS6_SELECT		CS5_SELECT		CS4_SELECT		CS3_SELECT		CS2_SELECT		CS1_SELECT		CS0_SELECT	
Reset	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0

BCH_LAYOUTSELECT*n* field descriptions

Field	Description
31–30 CS15_SELECT	Selects which layout is used for chip select 15.
29–28 CS14_SELECT	Selects which layout is used for chip select 14.
27–26 CS13_SELECT	Selects which layout is used for chip select 13.
25–24 CS12_SELECT	Selects which layout is used for chip select 12.
23–22 CS11_SELECT	Selects which layout is used for chip select 11.
21–20 CS10_SELECT	Selects which layout is used for chip select 10.
19–18 CS9_SELECT	Selects which layout is used for chip select 9.
17–16 CS8_SELECT	Selects which layout is used for chip select 8.
15–14 CS7_SELECT	Selects which layout is used for chip select 7.
13–12 CS6_SELECT	Selects which layout is used for chip select 6.
11–10 CS5_SELECT	Selects which layout is used for chip select 5.
9–8 CS4_SELECT	Selects which layout is used for chip select 4.
7–6 CS3_SELECT	Selects which layout is used for chip select 3.
5–4 CS2_SELECT	Selects which layout is used for chip select 2.
3–2 CS1_SELECT	Selects which layout is used for chip select 1.
CS0_SELECT	Selects which layout is used for chip select 0.

17.6.8 Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH0LAYOUT1 register to control the format for the devices selecting layout 0 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data,

metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See the BCH programming reference manual for more information on setting up the flash layout registers.

Address: 180_8000h base + 80h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NBLOCKS								META_SIZE							
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECC0					GF13_0_GF14_1	DATA0_SIZE									
W																
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH0LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed

Table continues on the next page...

BCH_FLASH0LAYOUT0*n* field descriptions (continued)

Field	Description
	0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block only contains metadata.

17.6.9 Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH0LAYOUT0 register to control the format for the device selecting layout 0 in the LAYOUTSELECT register.

Address: 180_8000h base + 90h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH0LAYOUT1*n* field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

17.6.10 Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0*n*)

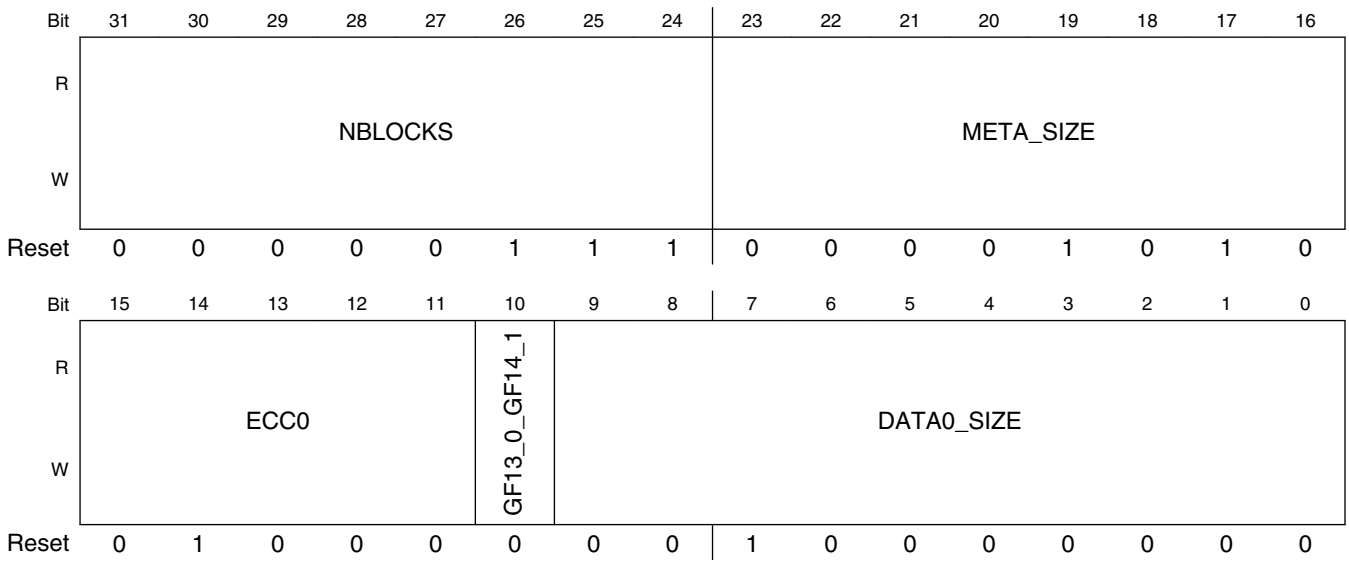
The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH1LAYOUT1 register to control the format for the devices selecting layout 1 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data,

metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See the BCH programming reference manual for more information on setting up the flash layout registers.

Address: 180_8000h base + A0h offset + (4d × i), where i=0d to 3d



BCH_FLASH1LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design supports from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data is in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed

Table continues on the next page...

BCH_FLASH1LAYOUT0*n* field descriptions (continued)

Field	Description
	0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block will contains metadata.

17.6.11 Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH1LAYOUT0 register to control the format for the device selecting layout 1 in the LAYOUTSELECT register.

Address: 180_8000h base + B0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH1LAYOUT1*n* field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

17.6.12 Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH2LAYOUT1 register to control the format for the devices selecting layout 2 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data,

metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See the BCH programming reference manual for more information on setting up the flash layout registers.

Address: 180_8000h base + C0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NBLOCKS								META_SIZE							
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECC0					GF13_0_GF14_1	DATA0_SIZE									
W																
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH2LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that eight subsequent blocks are present for a total of nine blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed

Table continues on the next page...

BCH_FLASH2LAYOUT0*n* field descriptions (continued)

Field	Description
	0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block will only contain metadata.

17.6.13 Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH2LAYOUT0 register to control the format for the device selecting layout 2 in the LAYOUTSELECT register.

Address: 180_8000h base + D0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH2LAYOUT1*n* field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

17.6.14 Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0*n*)

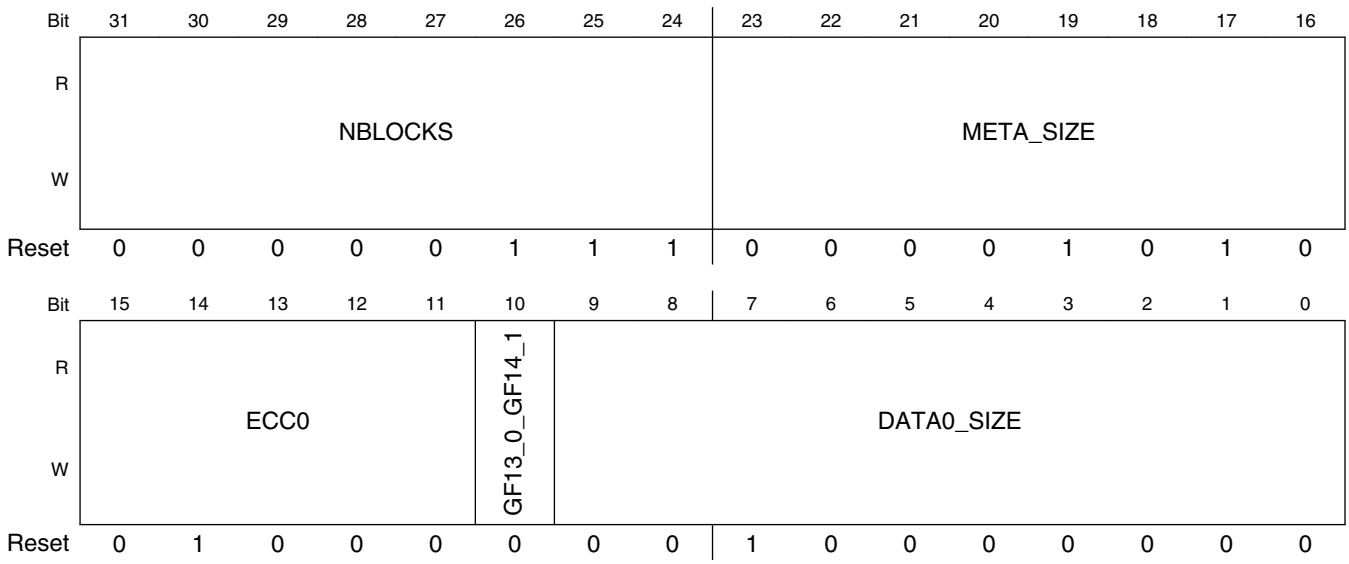
The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH3LAYOUT1 register to control the format for the devices selecting layout 3 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data,

metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See the BCH programming reference manual for more information on setting up the flash layout registers.

Address: 180_8000h base + E0h offset + (4d × i), where i=0d to 3d



BCH_FLASH3LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed

Table continues on the next page...

BCH_FLASH3LAYOUT0*n* field descriptions (continued)

Field	Description
	0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block will only contain metadata.

17.6.15 Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH3LAYOUT0 register to control the format for the device selecting layout 3 in the LAYOUTSELECT register.

Address: 180_8000h base + F0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH3LAYOUT1*n* field descriptions

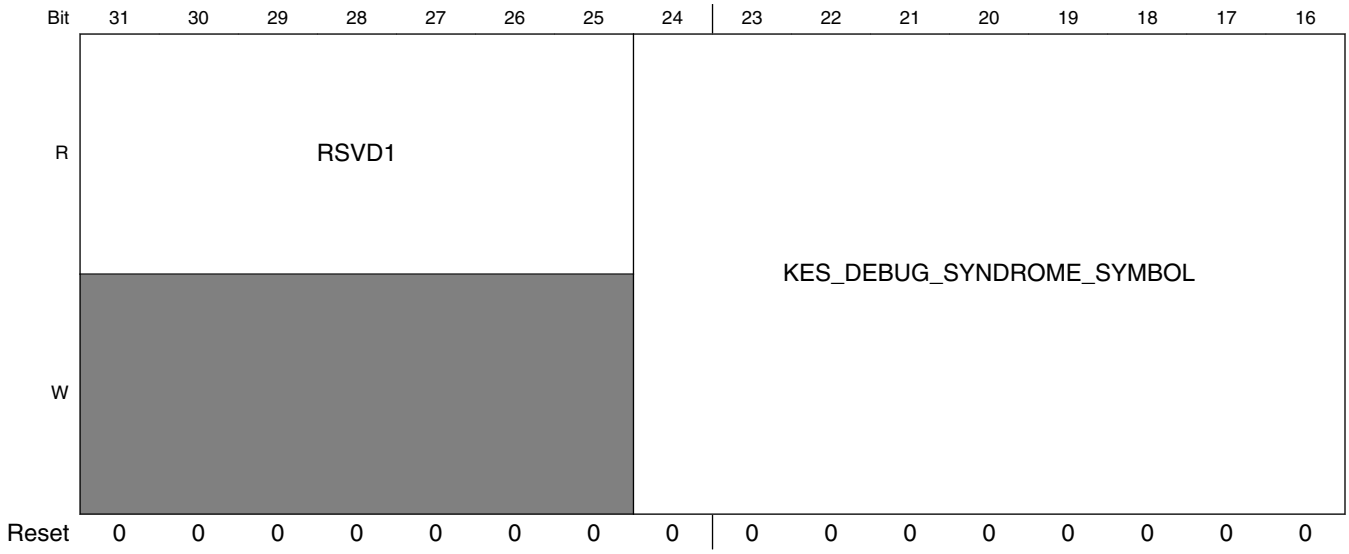
Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed 0x11 **ECC34** — ECC 34 to be performed 0x12 **ECC36** — ECC 36 to be performed 0x13 **ECC38** — ECC 38 to be performed 0x14 **ECC40** — ECC 40 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

17.6.16 Hardware BCH ECC Debug Register0 (BCH_DEBUG0*n*)

The hardware BCH accelerator internal state machines and signals can be seen in the ECC debug register.

The BCH_DEBUG0 register provides access to various internal state information which might prove useful during hardware debug and validation.

Address: 180_8000h base + 100h offset + (4d × i), where i=0d to 3d



BCH_DEBUG0*n* field descriptions

Field	Description
31–25 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
24–16 KES_DEBUG_ SYNDROME_ SYMBOL	The 9 bit value in this bit field shifts into the syndrome register array at the input of the KES engine whenever BCH_DEBUG0_KES_DEBUG_SHIFT_SYND is toggled. 0x0 **NORMAL** — Bus master address generator for SYND_GEN writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxiliary block.
15 KES_DEBUG_ SHIFT_SYND	Toggling this bit causes the value in BCH_DEBUG0_KES_SYNDROME_SYMBOL to be shift into the syndrome register array at the input to the KES engine. After shifting in 16 symbols, one can kick off both KES and CF cycles by toggling BCH_DEBUG0_KES_DEBUG_KICK. Make sure that set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.

Table continues on the next page...

BCH_DEBUG0*n* field descriptions (continued)

Field	Description
14 KES_DEBUG_ PAYLOAD_FLAG	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input payload flag. 0x1 **DATA** — Payload is set for 512 bytes data block. 0x1 **AUX** — Payload is set for 65 or 19 bytes auxiliary block.
13 KES_DEBUG_ MODE4K	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input mode (4K or 2K pages). 0x1 **4k** — Mode is set for 4K NAND pages. 0x1 **2k** — Mode is set for 2K NAND pages.
12 KES_DEBUG_ KICK	Toggling causes KES engine FSM to start as if kick by the Bus Master. This allows stand alone testing of the KES and Chien Search engines. Be sure to set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.
11 KES_ STANDALONE	Set to one, cause the KES engine to suppress toggling the KES_BM_DONE signal to the bus master and suppress toggling the CF_BM_DONE signal by the CF engine. 0x0 **NORMAL** — Bus master address generator for SYND_GEN writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxiliary block.
10 KES_DEBUG_ STEP	Toggling this bit causes the KES FSM to skip passed the stall state if it is in DEBUG_STALL mode and completed processing a block.
9 KES_DEBUG_ STALL	Set to one to cause KES FSM to stall after notifying Chien search engine to start processing its block but before notifying the bus master that the KES computation is complete. This allows a diagnostic to stall the FSM after each blocks key equations are solved. This also has the effect of stalling the CSFE search engine so it's state can be examined after it finishes processing the KES stalled block. 0x0 **NORMAL** — KES FSM proceeds to next block supplied by bus master. 0x1 **WAIT** — KES FSM waits after current equations are solved and the search engine is started.
8 BM_KES_TEST_ BYPASS	1 = Point all SYND_GEN writes to dummy area at the end of the AUXILLIARY block so that diagnostics can preload all payload, parity bytes and computed syndrome bytes for test the KES engine. 0x0 **NORMAL** — Bus master address generator for SYND_GEN writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxiliary block.
7–6 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.
DEBUG_REG_ SELECT	The value loaded in this bit field is used to select the internal register state view of KES engine or the Chien search engine.

17.6.17 KES Debug Read Register (BCH_DBGKESREAD*n*)

The hardware BCH ECC accelerator key equation solver internal state machines and signals can be seen in the ECC debug registers.

Address: 180_8000h base + 110h offset + (4d × i), where i=0d to 3d

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

BCH_DBGKESREAD*n* field descriptions

Field	Description
VALUES	This register returns the ROM BIST CRC value after a BIST test.

17.6.18 Chien Search Debug Read Register (BCH_DBGCSFEREAD*n*)

The hardware BCH ECC accelerator Chien Search internal state machines and signals can be seen in the ECC debug registers.

Address: 180_8000h base + 120h offset + (4d × i), where i=0d to 3d

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

BCH_DBGCSFEREAD*n* field descriptions

Field	Description
VALUES	Reserved

17.6.19 Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD*n*)

The hardware BCH ECC accelerator syndrome generator internal state machines and signals can be seen in the ECC debug registers.

Address: 180_8000h base + 130h offset + (4d × i), where i=0d to 3d

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

BCH_DBGSYNDGENREAD*n* field descriptions

Field	Description
VALUES	Reserved

17.6.20 Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD*n*)

The hardware BCH ECC accelerator bus master, ECC controller internal state machines, and signals can be seen in the ECC debug registers.

Address: 180_8000h base + 140h offset + (4d × i), where i=0d to 3d

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

BCH_DBGAHBMREAD*n* field descriptions

Field	Description
VALUES	Reserved

17.6.21 Block Name Register (BCH_BLOCKNAME*n*)

Read only view of the block name string BCH.

Fixed pattern read only value is for test purposes. It can be read as an ASCII string with the zero termination coming from the first byte of the BLOCKVERSION register.

Address: 180_8000h base + 150h offset + (4d × i), where i=0d to 3d

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

BCH_BLOCKNAME*n* field descriptions

Field	Description
NAME	The name is in the ASCII characters BCH (0x20, H, C, B).

17.6.22 BCH Version Register (BCH_VERSION*n*)

This register always returns a known read value for debug purposes and indicates the version of the block and RTL version in use.

Address: 180_8000h base + 160h offset + (4d × i), where i=0d to 3d

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

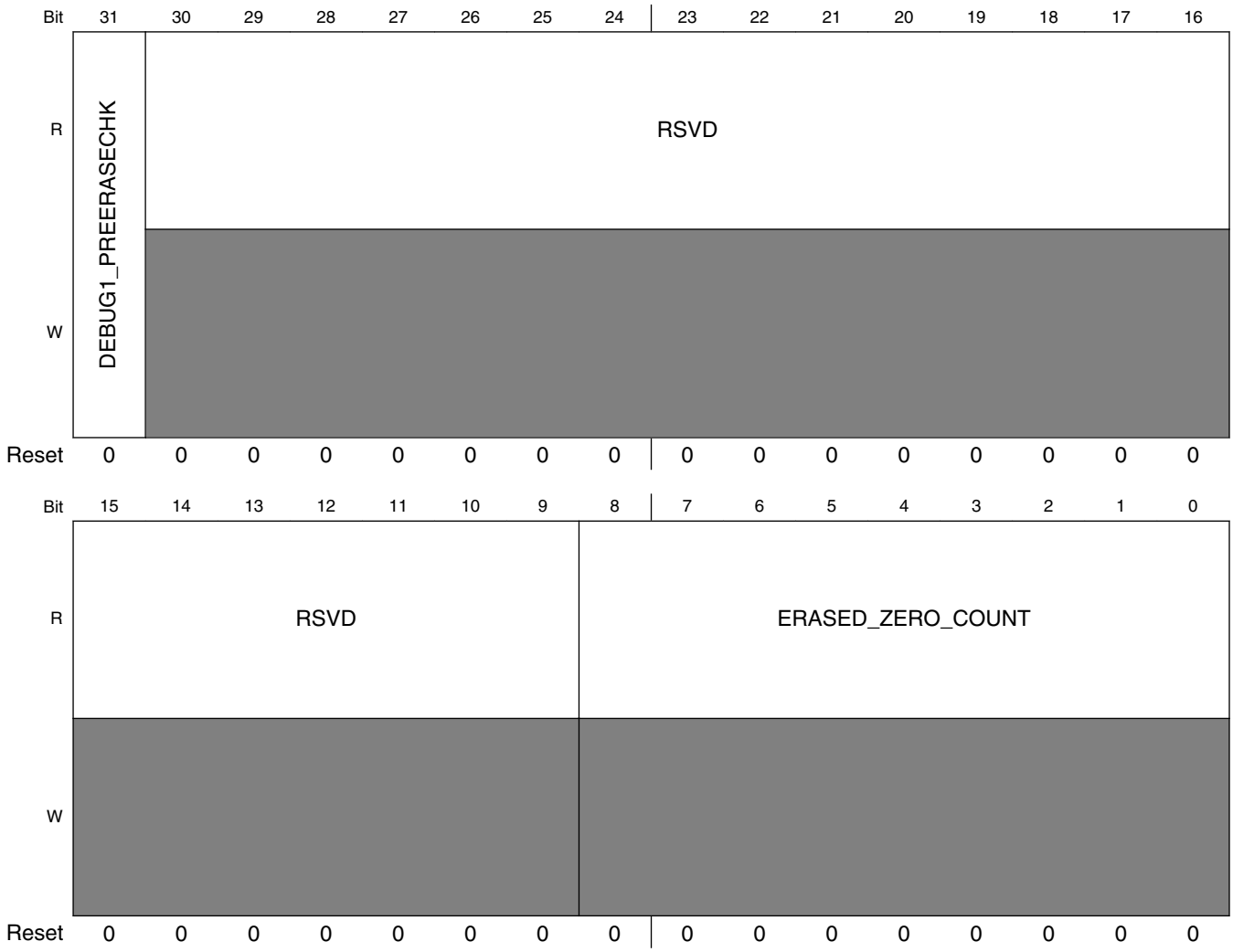
BCH_VERSION*n* field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value indicates the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value indicates the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

17.6.23 Hardware BCH ECC Debug Register 1 (BCH_DEBUG1*n*)

The BCH_DEBUG1 register provides erased zero count information and pre-erase check.

Address: 180_8000h base + 170h offset + (4d × i), where i=0d to 3d



BCH_DEBUG1*n* field descriptions

Field	Description
31 DEBUG1_ PREERASECHK	Blank page enables pre-erase check. 0x0 Turn off pre-erase check 0x1 Turn on pre-erase check
30–9 RSVD	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

BCH_DEBUG1*n* field descriptions (continued)

Field	Description
ERASED_ ZERO_COUNT	The zero counts on one page.

Chapter 18
Clock Controller Module (CCM)

18.1 Overview

The Clock Control Module (CCM) generates and controls clocks to the various modules in the design and manages low power modes. This module uses the available clock sources to generate the clock roots.

The Clock Controller Module controls the following functions:

- Uses the available clock sources to generate clock roots to various parts of the chip:
  - PLL1 also referenced as ARM PLL
  - PLL2 also referenced as System PLL
  - PLL3 also referenced as USB1 PLL
  - PLL4 also referenced as Audio PLL
  - PLL5 also referenced as Video PLL
  - PLL6 also referenced as ENET PLL
  - PLL7 also referenced as USB2 PLL (This PLL is only used by the USB UTM interface through a direct connection.)
- Uses programmable bits to control frequencies of the clock roots.
- Controls the low power mechanism.
- Provides control signals to LPCG for gating clocks.
- Provides handshake with SRC for reset performance.
- Provides handshake with GPC for support of DVFS and power gating operations.

18.1.1 Features

The CCM includes these distinctive features:

- Provides root clock to SoC modules based on several source clocks.
- ARM core root clock is generated from a dedicated source clock.

- Includes separate dividers to control generation of core and bus root clocks (AXI, AHB, IPG).
- Includes separate dividers and clock source selectors for each serial root clock.
- Optional external clocks to bypass PLL clocks.
- Selects clock signals to output on CCM_CLKO onto the pads for observability.
- Controllable registers are accessible via IP bus.
- Manages the Low Power Modes, namely RUN, WAIT and STOP. The gating of the peripheral clocks is programmable in RUN and WAIT modes.
- Manages frequency scaling procedure for ARM core clock by shifting between PLL sources, without loss of clocks.
- Manages frequency scaling procedure for peripheral root clock by programmable divider. The division is done on the fly without loss of clocks.
- Interface for the following IPs:
    - PLL - Interfaces for each PLL
    - LPCG - Low Power Clock Gating unit
    - SRC - System Reset Controller
    - GPC - General Power Controller

18.1.2 CCM Block Diagram

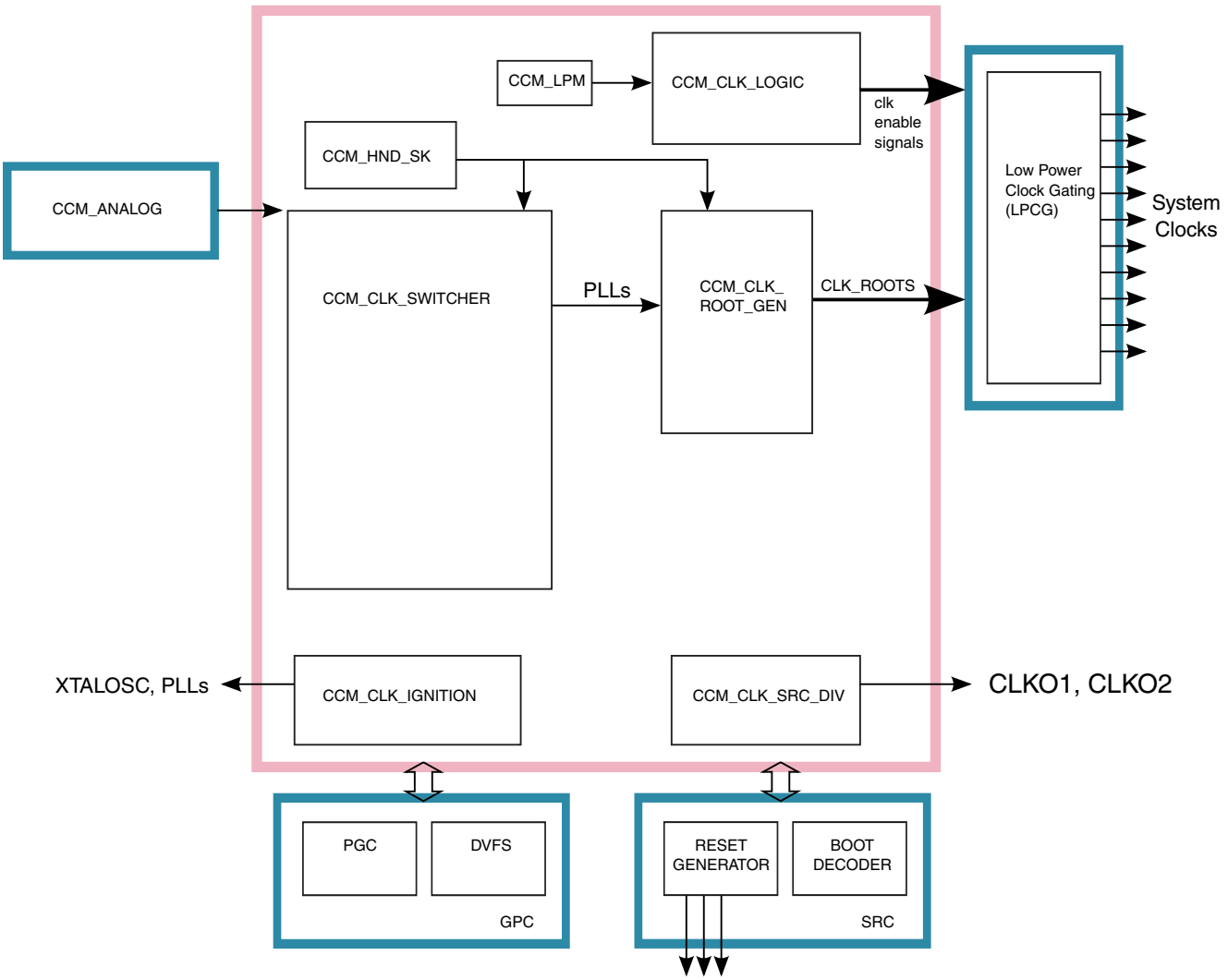


Figure 18-1. Block Diagram

CCM Contains the following sub-blocks:

Table 18-1. CCM Sub-blocks

CCM_CLK_SRC_DIV Sub-block	Description
CCM_CLK_IGNITION	Manages the ignition process. This module starts its functionality once CCM comes out of reset. It manages the process that begins with starting the OSC, PLLs and finishes with creation of stable output root clocks after reset.
CCM_CLK_SWITCHER	Receives the clock outputs of the PLLs, together with the bypass clocks for the PLLs, and generates three switcher clock outputs (pll1_sw_clk, pll3_sw_clk) for the CCM_CLK_ROOT_GEN sub-module.
CCM_CLK_ROOT_GEN	Receives the main clocks (PLLs / PFDs) and generates the output root clocks.
CCM_CLK_LOGIC	Generates the clock enables. It generates the clock enable signals based on info from CCM_LPM and CCM_IP. The clock enables are used in LPCG to turn off and on the split clocks.

Table continues on the next page...

Table 18-1. CCM Sub-blocks (continued)

CCM_CLK_SRC_DIV Sub-block	Description
CCM_LPM	Manages the low power modes of the IC.
CCM_HND_SK	Manages the handshake when changing root clock dividers that require handshake, and manages the frequency change in case of dvfs scenario.

18.2 External Signals

The following table describes the external signals of CCM:

Table 18-2. CCM External Signals

Signal	Description	Pad	Mode	Direction
CCM_CLKO1	Observability clock 1 output	JTAG_TMS	ALT3	O
		SD1_DATA2	ALT6	
CCM_CLKO2	Observability clock 2 output	JTAG_TDO	ALT3	O
		SD1_DATA3	ALT6	
CCM_REF_EN_B	Enable external reference clock (CKIH)	GPIO1_IO06	ALT7	O

18.3 CCM Clock Tree

The figure found here shows the clock tree configuration and clock roots for CCM.

For detailed sub-block information, please see:

- Clock Switcher
- Clock Root Generator
- Low Power Clock Gating module (LPCG)
- System Clocks

NOTE

The PLL and PFD control registers may be programmed according to the speed grade of the SoC, but should not exceed that maximum setting for that speed grade. The default frequency values (in MHz) for the PLLs and PFDs are shown in the Clock Tree below.

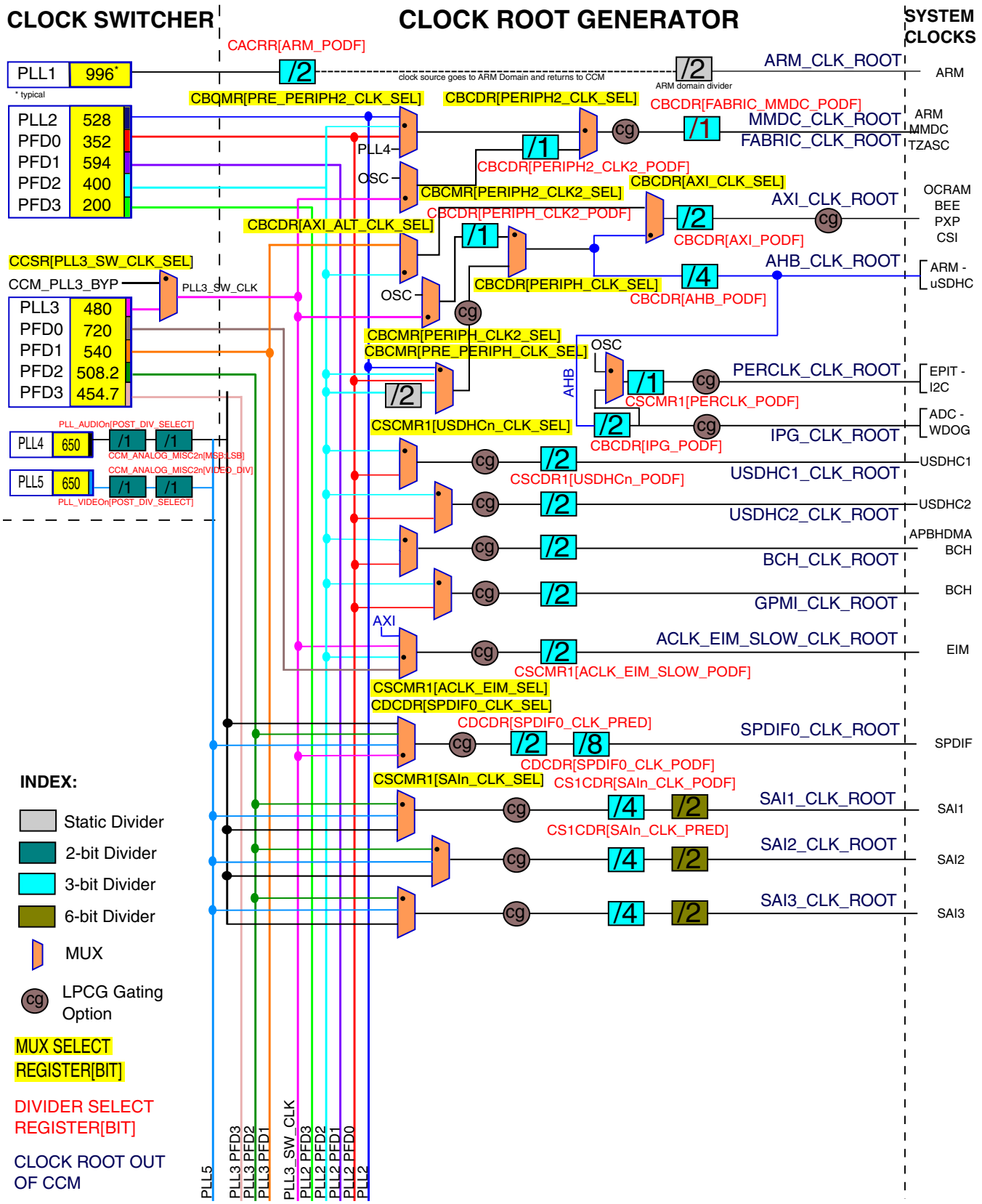


Figure 18-2. Clock Tree - Part 1

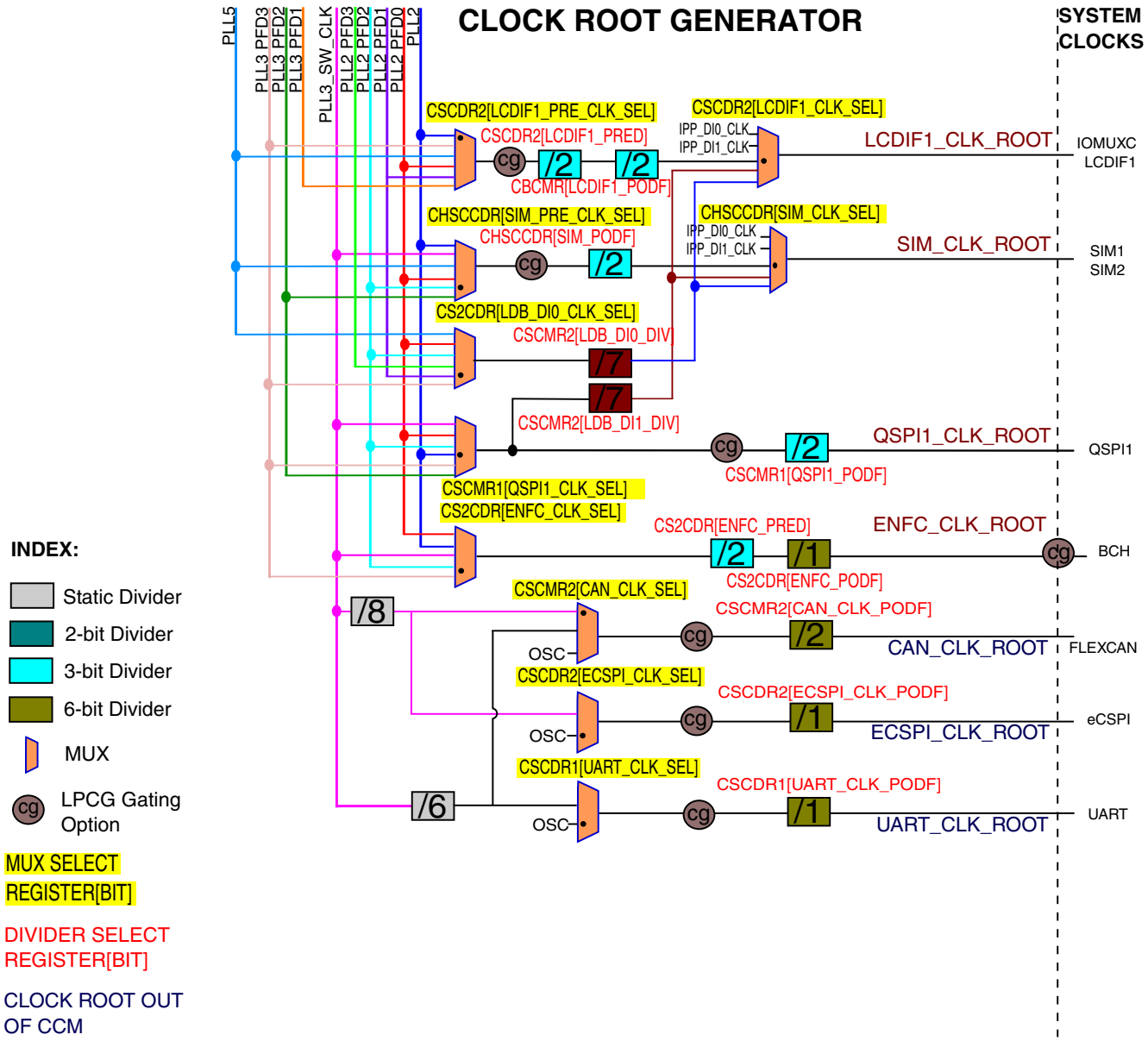


Figure 18-3. Clock Tree - Part 2

18.4 System Clocks

The table found here shows the CCM output clocks' system-level connectivity.

The gating option in the table can either be CGR bit or clock enable from the block itself. Applicable override bits are also shown.

Table 18-3. System Clocks, Gating, and Override

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
ADC	IPG_CLK	IPG_CLK_ROOT	CCGR1[CG8] (ADC1_CLK_ENABLE) CCGR1[CG4] (ADC2_CLK_ENABLE)	
AIPSTZ*n*	HCLK	AHB_CLK_ROOT	CCGR0[CG0] (AIPS_TZ1_CLK_ENABLE) CCGR0[CG1] (AIPS_TZ2_CLK_ENABLE)	
APBHDMA	HCLK	BCH_CLK_ROOT	CCGR0[CG2] (APBHDMA_HCLK_ENABLE)	
	SEC_MST_HCLK	BCH_CLK_ROOT	CCGR0[CG2] (APBHDMA_HCLK_ENABLE)	
ARM	TRACE_CLK_IN	AHB_CLK_ROOT	CCGR0[CG11] (ARM_DBG_CLK_ENABLE)	
	CLKDIV_PATCH_IPG_CLK	AHB_CLK_ROOT	CCGR3[CG9] (A7CLKDIV_PATCH_CLK_ENABLE)	
ASRC	ASRCK_CLOCK_D	SPDIF0_CLK_ROOT	CCGR0[CG3] (ASRC_CLK_ENABLE)	
	IPG_CLK	AHB_CLK_ROOT	CCGR0[CG3] (ASRC_CLK_ENABLE)	
	MEM_CLK	AHB_CLK_ROOT	CCGR0[CG3] (ASRC_CLK_ENABLE)	
BCH	U_BCH_INPUT_APB_CLK	BCH_CLK_ROOT	CCGR4[CG12] (RAWNAND_U_BCH_INPUT_APB_CLK_ENABLE)	
	U_GPMI_BCH_INPUT_BCH_CLK	GPMI_CLK_ROOT	CCGR4[CG13] (RAWNAND_U_GPMI_BCH_INPUT_BCH_CLK_ENABLE)	
	U_GPMI_BCH_INPUT_GPMI_IO_CLK	ENFC_CLK_ROOT	CCGR4[CG14] (RAWNAND_U_GPMI_BCH_INPUT_GPMI_IO_CLK_ENABLE)	
	U_GPMI_INPUT_APB_CLK	BCH_CLK_ROOT	CCGR4[CG15] (RAWNAND_U_GPMI_INPUT_APB_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
CAAM	SECURE_MEM_CLK	AHB_CLK_ROOT	CCGR0[CG4] (CAAM_SECURE_MEM_CLK_ENABLE)	
	ACLK	AHB_CLK_ROOT	CCGR0[CG5] (CAAM_WRAPPER_ACLK_ENABLE)	
	CKIL	CKIL_SYNC_CLK_ROOT		
	IPG_CLK	IPG_CLK_ROOT	CCGR0[CG5] (CAAM_WRAPPER_ACLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR0[CG5] (CAAM_WRAPPER_ACLK_ENABLE)	
	ACLK_EXSC	AHB_CLK_ROOT	CCGR0[CG5] (CAAM_WRAPPER_ACLK_ENABLE)	
CSI	CSI_HCLK	AXI_CLK_ROOT	CCGR2[CG1] (CSI_CLK_ENABLE)	
	IPG_CLK	AXI_CLK_ROOT	CCGR2[CG1] (CSI_CLK_ENABLE)	
	IPG_CLK_S	AXI_CLK_ROOT	CCGR2[CG1] (CSI_CLK_ENABLE)	
	IPG_CLK_S_RAW	AXI_CLK_ROOT	CCGR2[CG1] (CSI_CLK_ENABLE)	
	MEM_RXFIFO_CLK	AXI_CLK_ROOT	CCGR2[CG1] (CSI_CLK_ENABLE)	
CSU	AP_CKIL_CLK	CKIL_SYNC_CLK_ROOT	CCGR1[CG14] (CSU_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR1[CG14] (CSU_CLK_ENABLE)	
ECSPI*n*	IPG_CLK	IPG_CLK_ROOT	CCGR1[CG0] (ECSPI1_CLK_ENABLE) CCGR1[CG1] (ECSPI2_CLK_ENABLE) CCGR1[CG2] (ECSPI3_CLK_ENABLE) CCGR1[CG3] (ECSPI4_CLK_ENABLE)	
	IPG_CLK_32K	CKIL_SYNC_CLK_ROOT		
	IPG_CLK_PER	ECSPI_CLK_ROOT	CCGR1[CG0] (ECSPI1_CLK_ENABLE) CCGR1[CG1] (ECSPI2_CLK_ENABLE) CCGR1[CG2] (ECSPI3_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
			CCGR1[CG3] (ECSPI4_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR1[CG0] (ECSPI1_CLK_ENABLE) CCGR1[CG1] (ECSPI2_CLK_ENABLE) CCGR1[CG2] (ECSPI3_CLK_ENABLE) CCGR1[CG3] (ECSPI4_CLK_ENABLE)	
EIM	ACLK	ACLK_EIM_SLOW_CLK_ROOT	CCGR6[CG5] (EIM_SLOW_CLK_ENABLE)	
	ACLK_SLOW	ACLK_EIM_SLOW_CLK_ROOT	CCGR6[CG5] (EIM_SLOW_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT		
	ACLK_EXSC	ACLK_EIM_SLOW_CLK_ROOT	CCGR6[CG5] (EIM_SLOW_CLK_ENABLE)	
	IPG_CLK	IPG_CLK_ROOT		
ENET*n*	IPG_CLK	AHB_CLK_ROOT	CCGR3[CG2] (ENET_CLK_ENABLE)	
	IPG_CLK_MAC0	AHB_CLK_ROOT	CCGR3[CG2] (ENET_CLK_ENABLE)	
	IPG_CLK_MAC0_S	IPG_CLK_ROOT	CCGR3[CG2] (ENET_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR3[CG2] (ENET_CLK_ENABLE)	
	IPG_CLK_TIME	REF_ENETPLL2_CLK	CCGR3[CG2] (ENET_CLK_ENABLE)	
	MAC0_RXMEM_CLK	AHB_CLK_ROOT	CCGR3[CG2] (ENET_CLK_ENABLE)	
	MAC0_TXMEM_CLK	AHB_CLK_ROOT	CCGR3[CG2] (ENET_CLK_ENABLE)	
EPIT*n*	IPG_CLK	PERCLK_CLK_ROOT	CCGR1[CG6] (EPIT1_CLK_ENABLE) CCGR1[CG7] (EPIT2_CLK_ENABLE)	CMEOR[MOD_EN_OV_EPIT]
	IPG_CLK_32K	CKIL_SYNC_CLK_ROOT		
	IPG_CLK_HIGHFREQ	PERCLK_CLK_ROOT	CCGR1[CG6] (EPIT1_CLK_ENABLE) CCGR1[CG7] (EPIT2_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
	IPG_CLK_S	PERCLK_CLK_ROOT	CCGR1[CG6] (EPIT1_CLK_ENABLE) CCGR1[CG7] (EPIT2_CLK_ENABLE)	
FLEXCAN*n*	IPG_CLK	IPG_CLK_ROOT	CCGR0[CG7] (CAN1_CLK_ENABLE) CCGR0[CG9] (CAN2_CLK_ENABLE)	
	IPG_CLK_CHI	IPG_CLK_ROOT	CCGR0[CG7] (CAN1_CLK_ENABLE) CCGR0[CG9] (CAN2_CLK_ENABLE)	
	IPG_CLK_PE	CAN_CLK_ROOT	CCGR0[CG8] (CAN1_SERIAL_CLK_ENABLE) CCGR0[CG10] (CAN2_SERIAL_CLK_ENABLE)	CMEOR[MOD_EN_OV_CAN*n*_CPI]
	IPG_CLK_PE_NOGATE	CAN_CLK_ROOT	CCGR0[CG8] (CAN1_SERIAL_CLK_ENABLE) CCGR0[CG10] (CAN2_SERIAL_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR0[CG7] (CAN1_CLK_ENABLE) CCGR0[CG9] (CAN2_CLK_ENABLE)	
	RAM_CLK	IPG_CLK_ROOT	CCGR0[CG7] (CAN1_CLK_ENABLE) CCGR0[CG9] (CAN2_CLK_ENABLE)	
GPC	IPG_CLK	IPG_CLK_ROOT		
	IPG_CLK_S	IPG_CLK_ROOT		
	PGC_CLK	IPG_CLK_ROOT		
	SPARE_IN[0]	CKIL_SYNC_CLK_ROOT		
	SYS_CLK	IPG_CLK_ROOT		
GPIO*n*	IPG_CLK_S	IPG_CLK_ROOT 0	CCGR1[CG13] (GPIO1_CLK_ENABLE) CCGR0[CG15] (GPIO2_CLK_ENABLE) CCGR2[CG13] (GPIO3_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
			CCGR3[CG6] (GPIO4_CLK_ENABLE) CCGR1[CG15] (GPIO5_CLK_ENABLE)	
GPT*n*	IPG_CLK	PERCLK_CLK_ROOT	CCGR1[CG10] (GPT1_CLK_ENABLE) CCGR0[CG12] (GPT2_CLK_ENABLE)	CMEOR[MOD_EN_OV_GPT]
	IPG_CLK_24M	CKIH_SYNC_CLK_ROOT		CMEOR[MOD_EN_OV_GPT]
	IPG_CLK_32K	CKIL_SYNC_CLK_ROOT		
	IPG_CLK_HIGHFREQ	PERCLK_CLK_ROOT	CCGR1[CG11] (GPT1_SERIAL_CLK_ENABLE) CCGR0[CG13] (GPT2_SERIAL_CLK_ENABLE)	
	IPG_CLK_S	PERCLK_CLK_ROOT	CCGR1[CG10] (GPT1_CLK_ENABLE) CCGR0[CG12] (GPT2_CLK_ENABLE)	
HS	CLK	IPG_CLK_ROOT		
I2C*n*	IPG_CLK_PATREF	PERCLK_CLK_ROOT	CCGR2[CG3] (I2C1_SERIAL_CLK_ENABLE) CCGR2[CG4] (I2C2_SERIAL_CLK_ENABLE) CCGR2[CG5] (I2C3_SERIAL_CLK_ENABLE) CCGR6[CG12] (I2C4_SERIAL_CLK_ENABLE)	
	IPG_CLK_S	PERCLK_CLK_ROOT	CCGR2[CG3] (I2C1_SERIAL_CLK_ENABLE) CCGR2[CG4] (I2C2_SERIAL_CLK_ENABLE) CCGR2[CG5] (I2C3_SERIAL_CLK_ENABLE) CCGR6[CG12] (I2C4_SERIAL_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
IOMUXC	IPT_CLK_IO	LCDIF_PIX_CLK_ROOT	CCGR2[CG7] (IOMUX_IPT_CLK_IO_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR4[CG1] (IOMUXC_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR4[CG2] (IOMUXC_GPR_CLK_ENABLE)	
KPP*n*	IPG_CLK_32K	CKIL_SYNC_CLK_ROOT		
	IPG_CLK_S	IPG_CLK_ROOT	CCGR5[CG4] (KPP_CLK_ENABLE)	
LCDIF	APB_CLK	AXI_CLK_ROOT	CCGR2[CG14] (LCD_CLK_ENABLE)	
	PIX_CLK	LCDIF_PIX_CLK_ROOT	CCGR3[CG5] (LCDIF_PIX_CLK_ENABLE)	
MMDC	ACLK_FAST_CORE_P0	MMDC_CLK_ROOT	CCGR3[CG10] (MMDC_CORE_ACLK_FAST_CORE_P0_ENABLE)	
	ACLK_FAST_PHY_P0	MMDC_CLK_ROOT	CCGR3[CG10] (MMDC_CORE_ACLK_FAST_CORE_P0_ENABLE)	
	CLK32	CKIL_SYNC_CLK_ROOT	CCGR3[CG13] (MMDC_CORE_IPG_CLK_P1_ENABLE)	
	IPG_CLK_P0	IPG_CLK_ROOT	CCGR3[CG12] (MMDC_CORE_IPG_CLK_P0_ENABLE)	
	ACLK_EXSC	MMDC_CLK_ROOT	CCGR3[CG10] (MMDC_CORE_ACLK_FAST_CORE_P0_ENABLE)	
OCOTP	IPG_CLK	IPG_CLK_ROOT	CCGR2[CG6] (IIM_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR2[CG6] (IIM_CLK_ENABLE)	
OCRAM	CTRL_CLK	AXI_CLK_ROOT	CCGR3[CG14] (OCRAM_CLK_ENABLE)	
	ACLK_EXSC	AXI_CLK_ROOT	CCGR3[CG14] (OCRAM_CLK_ENABLE)	
	MEM_CLK	AXI_CLK_ROOT	CCGR3[CG14] (OCRAM_CLK_ENABLE)	
PWM*n*	IPG_CLK	PERCLK_CLK_ROOT	CCGR4[CG11:CG8] (PWM[4:1]_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
			CCGR6[CG15:CG13] (PWM[7:5]_CLK_ENABLE) CCGR6[CG8] (PWM8_CLK_ENABLE)	
	IPG_CLK_32K	CKIL_SYNC_CLK_ROOT		
	IPG_CLK_HIGHFREQ	PERCLK_CLK_ROOT	CCGR4[CG11:CG8] (PWM[4:1]_CLK_ENABLE) CCGR6[CG15:CG13] (PWM[7:5]_CLK_ENABLE) CCGR6[CG8] (PWM8_CLK_ENABLE)	
	IPG_CLK_S	PERCLK_CLK_ROOT	CCGR4[CG11:CG8] (PWM[4:1]_CLK_ENABLE) CCGR6[CG15:CG13] (PWM[7:5]_CLK_ENABLE) CCGR6[CG8] (PWM8_CLK_ENABLE)	
PXP	CLK	AXI_CLK_ROOT	CCGR2[CG15] (PXP_CLK_ENABLE)	
QSPI	AHB_CLK	AHB_CLK_ROOT	CCGR3[CG7] (QSPI_CLK_ENABLE)	
	IPG_CLK	IPG_CLK_ROOT	CCGR3[CG7] (QSPI_CLK_ENABLE)	
	IPG_CLK_4XSFIF	QSPI*n*_CLK_ROOT	CCGR3[CG7] (QSPI_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR3[CG7] (QSPI_CLK_ENABLE)	
	SEC_IPG_CLK	IPG_CLK_ROOT	CCGR3[CG7] (QSPI_CLK_ENABLE)	
	SEC_IPG_CLK_S	IPG_CLK_ROOT	CCGR3[CG7] (QSPI_CLK_ENABLE)	
	MST_HCLK	AHB_CLK_ROOT	CCGR3[CG7] (QSPI_CLK_ENABLE)	
ROMCP	ROM_CLK	AHB_CLK_ROOT	CCGR5[CG0] (ROM_CLK_ENABLE)	
	HCLK	AHB_CLK_ROOT	CCGR5[CG0] (ROM_CLK_ENABLE)	
	HCLK_REG	IPG_CLK_ROOT	CCGR5[CG0] (ROM_CLK_ENABLE)	
	MST_HCLK	AHB_CLK_ROOT	CCGR5[CG0] (ROM_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
SAI*n*	IPG_CLK	IPG_CLK_ROOT	CCGR5[CG14] (SAI1_CLK_ENABLE) CCGR5[CG15] (SAI2_CLK_ENABLE) CCGR5[CG11] (SAI3_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR5[CG14] (SAI1_CLK_ENABLE) CCGR5[CG15] (SAI2_CLK_ENABLE) CCGR5[CG11] (SAI3_CLK_ENABLE)	
	IPG_CLK_SAI_MCLK1	SAIN_CLK_ROOT	CCGR5[CG14] (SAI1_CLK_ENABLE) CCGR5[CG15] (SAI2_CLK_ENABLE) CCGR5[CG11] (SAI3_CLK_ENABLE)	
	IPT_CLK_SAI_BCLK	SAIN_CLK_ROOT	CCGR5[CG14] (SAI1_CLK_ENABLE) CCGR5[CG15] (SAI2_CLK_ENABLE) CCGR5[CG11] (SAI3_CLK_ENABLE)	
	IPT_CLK_SAI_BCLK_B	SAIN_CLK_ROOT	CCGR5[CG14] (SAI1_CLK_ENABLE) CCGR5[CG15] (SAI2_CLK_ENABLE) CCGR5[CG11] (SAI3_CLK_ENABLE)	
	IPT_CLK_SAI_MCLK	SAIN_CLK_ROOT	CCGR5[CG14] (SAI1_CLK_ENABLE) CCGR5[CG15] (SAI2_CLK_ENABLE) CCGR5[CG11] (SAI3_CLK_ENABLE)	
SDMA	IPS_HOSTCTRL_CLK	IPG_CLK_ROOT	CCGR5[CG3] (SDMA_CLK_ENABLE)	
	SDMA_AP_AHB_CLK	AHB_CLK_ROOT	CCGR5[CG3] (SDMA_CLK_ENABLE)	
	SDMA_CORE_CLK	IPG_CLK_ROOT	CCGR5[CG3] (SDMA_CLK_ENABLE)	
	EVENTS_SYNC_CLK	AHB_CLK_ROOT	CCGR5[CG3] (SDMA_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
SIM*n*	IPG_CLK	IPG_CLK_ROOT	CCGR6[CG3] (SIM1_CLK_ENABLE) CCGR6[CG4] (SIM2_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR6[CG3] (SIM1_CLK_ENABLE) CCGR6[CG4] (SIM2_CLK_ENABLE)	
	IPG_PERCLK	SIM_CLK_ROOT	CCGR6[CG3] (SIM1_CLK_ENABLE) CCGR6[CG4] (SIM2_CLK_ENABLE)	
	IPG_LOW_REF_CLK	CKIL_SYNC_CLK_ROOT		
SNVS	HP_IPG_CLK	IPG_CLK_ROOT	CCGR5[CG9] (SNVS_HP_CLK_ENABLE)	
	HP_IPG_CLK_S	IPG_CLK_ROOT	CCGR5[CG9] (SNVS_HP_CLK_ENABLE)	
	HP_IPG_HP_RTC_CLK	CKIL_SYNC_CLK_ROOT		
	LP_IPG_CLK	IPG_CLK_ROOT	CCGR5[CG10] (SNVS_LP_CLK_ENABLE)	
	IPG_DRYICE_CLK_S	IPG_CLK_ROOT	CCGR5[CG2] (SNVS_DRYICE_CLK_ENABLE)	
	LP_IPG_CLK_S	IPG_CLK_ROOT	CCGR5[CG10] (SNVS_LP_CLK_ENABLE)	
SPBA	IPG_CLK	IPG_CLK_ROOT	CCGR5[CG6] (SPBA_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR5[CG6] (SPBA_CLK_ENABLE)	
	DISP_IPG_CLK	IPG_CLK_ROOT	CCGR5[CG6] (SPBA_CLK_ENABLE)	
	DISP_IPG_CLK_S	IPG_CLK_ROOT	CCGR5[CG6] (SPBA_CLK_ENABLE)	
SPDIF	GCLKW_T0	IPG_CLK_ROOT	CCGR5[CG7] (SPDIF_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT		
	TX_CLK	SPDIF0_CLK_ROOT	CCGR5[CG7] (SPDIF_CLK_ENABLE)	
SRC	IPG_CLK	IPG_CLK_ROOT		
	SRC_IPG_CLK_S	IPG_CLK_ROOT		

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
TZASC*n*	ACLK	MMDC_CLK_ROOT	CCGR2[CG11] (IPSYNC_IP2APB_TZASC1_IPG_MASTER_CLK_ENABLE)	
TSC_DIG	IPG_CLK_S	IPG_CLK_ROOT	CCGR4[CG5] (TSC_CLK_ENABLE)	
	IPG_CLK	IPG_CLK_ROOT	CCGR4[CG5] (TSC_CLK_ENABLE)	
	LP_CLK	CKIL_SYNC_CLK_ROOT		
UART*n*	IPG_CLK	IPG_CLK_ROOT	CCGR5[CG12] (UART1_CLK_ENABLE) CCGR0[CG14] (UART2_CLK_ENABLE) CCGR1[CG5] (UART3_CLK_ENABLE) CCGR1[CG12] (UART4_CLK_ENABLE) CCGR3[CG1] (UART5_CLK_ENABLE) CCGR3[CG3] (UART6_CLK_ENABLE) CCGR5[CG13] (UART7_CLK_ENABLE) CCGR6[CG7] (UART8_CLK_ENABLE)	
	IPG_CLK_S	IPG_CLK_ROOT	CCGR5[CG12] (UART1_CLK_ENABLE) CCGR0[CG14] (UART2_CLK_ENABLE) CCGR1[CG5] (UART3_CLK_ENABLE) CCGR1[CG12] (UART4_CLK_ENABLE) CCGR3[CG1] (UART5_CLK_ENABLE) CCGR3[CG3] (UART6_CLK_ENABLE) CCGR5[CG13] (UART7_CLK_ENABLE) CCGR6[CG7] (UART8_CLK_ENABLE)	
	IPG_PERCLK	UART_CLK_ROOT	CCGR5[CG12] (UART1_CLK_ENABLE)	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
			CCGR0[CG14] (UART2_CLK_ENABLE) CCGR1[CG5] (UART3_CLK_ENABLE) CCGR1[CG12] (UART4_CLK_ENABLE) CCGR3[CG1] (UART5_CLK_ENABLE) CCGR3[CG3] (UART6_CLK_ENABLE) CCGR5[CG13] (UART7_CLK_ENABLE) CCGR6[CG7] (UART8_CLK_ENABLE)	
USB	IPG_AHB_CLK	AHB_CLK_ROOT	CCGR6[CG0] (USBOH3_CLK_ENABLE)	
	IPG_CLK_32KHZ	CKIL_SYNC_CLK_ROOT		
	IPG_CLK_S	IPG_CLK_ROOT	CCGR6[CG0] (USBOH3_CLK_ENABLE)	
	IPG_CLK_S_PL301	IPG_CLK_ROOT	CCGR6[CG0] (USBOH3_CLK_ENABLE)	
	TEST_CLK_240M	IPG_CLK_ROOT	CCGR6[CG0] (USBOH3_CLK_ENABLE)	
	TEST_CLK_480M	IPG_CLK_ROOT	CCGR6[CG0] (USBOH3_CLK_ENABLE)	
	TEST_CLK_60M	IPG_CLK_ROOT	CCGR6[CG0] (USBOH3_CLK_ENABLE)	
USDHC*n*	HCLK	AHB_CLK_ROOT	CCGR6[CG1] (USDHC1_CLK_ENABLE) CCGR6[CG2] (USDHC2_CLK_ENABLE)	CMEOR[MOD_EN_OV_USDHC]
	IPG_CLK	IPG_CLK_ROOT	CCGR6[CG1] (USDHC1_CLK_ENABLE) CCGR6[CG2] (USDHC2_CLK_ENABLE)	CMEOR[MOD_EN_OV_USDHC]
	IPG_CLK_PERCLK	USDHC*n*_CLK_ROOT	CCGR6[CG1] (USDHC1_CLK_ENABLE) CCGR6[CG2] (USDHC2_CLK_ENABLE)	CMEOR[MOD_EN_OV_USDHC]
	IPG_CLK_S	IPG_CLK_ROOT	CCGR6[CG1] (USDHC1_CLK_ENABLE) CCGR6[CG2] (USDHC2_CLK_ENABLE)	
WDOG*n*	IPG_CLK	IPG_CLK_ROOT	CCGR3[CG8] WDOG1_CLK_ENABLE	

Table continues on the next page...

Table 18-3. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
			CCGR5[CG5] WDOG2_CLK_ENABLE CCGR6[CG10] WDOG3_CLK_ENABLE	
	IPG_CLK_32K	CKIL_SYNC_CLK_ROOT		
	IPG_CLK_S	IPG_CLK_ROOT	CCGR3[CG8] WDOG1_CLK_ENABLE CCGR5[CG5] WDOG2_CLK_ENABLE CCGR6[CG10] WDOG3_CLK_ENABLE	

Table 18-4. System Clock Frequency Values

Clock Root	Default Frequency (MHz)	Maximum Frequency (MHz)
ARM_CLK_ROOT	12	528
MMDC_CLK_ROOT FABRIC_CLK_ROOT	24	396
AXI_CLK_ROOT	12	264
AHB_CLK_ROOT	6	132
PERCLK_CLK_ROOT	3	66
IPG_CLK_ROOT	3	66
USDHCn_CLK_ROOT	12	198
ACLK_EIM_SLOW_CLK_ROOT	6	132
SPDIF0_CLK_ROOT	1.5	66.6
SAIn_CLK_ROOT	3	66.6
LCDIF_CLK_ROOT	6	150
SIM_CLK_ROOT	12	264
QSPI_CLK_ROOT	12	396
ENFC_CLK_ROOT	12	198
CAN_CLK_ROOT	1.5	80
ECSPI_CLK_ROOT	3	60
UART_CLK_ROOT	4	80

NOTE

The default frequency is reset value after system power on reset. It is different with the value initialized by ROM code, which depends on boot CFG pins/fuse setting.

18.5 Functional Description

This section provides a complete functional description of the block.

18.5.1 Clock Generation

18.5.1.1 External Low Frequency Clock - CKIL

The chip can use a 32 kHz or 32.768 kHz crystal as the external low-frequency source (XTALOSC). Throughout this chapter, the low-frequency crystal is referred to as the 32 kHz crystal.

This clock source should always be active when the chip is powered on. The 32 kHz entering the CCM are referred to as CKIL. CKIL is synchronized to IPG_CLK and supplied to modules that need it.

18.5.1.1.1 CKIL synchronizing to IPG_CLK

CKIL is synchronized to ipg_clk when the system is in functional mode. When the system is in STOP mode (when there is no IPG_CLK) the CKIL synchronizer is bypassed, and raw CKIL is supplied to the system.

18.5.1.2 External High Frequency Clock - CKIH and internal oscillator

The chip uses an internal oscillator to generate the reference clock (OSC). The internal oscillator is connected to the external crystal (XTALOSC) which generates the 24 MHz reference clock.

18.5.1.3 PLL reference clock

There are several PLLs in this chip.

PLL1 - ARM PLL (typical functional frequency)

PLL2 - System PLL (functional frequency 528 MHz)

PLL3 - USB1 PLL (functional frequency 480 MHz)

PLL4 - Audio PLL

PLL5 - Video PLL

PLL6 - ENET PLL

PLL7 - USB2 PLL (functional frequency 480 MHz)

Some of the PLLs are described in the sections below. See CCM Analog Memory Map/ Register Definition for register information.

18.5.1.3.1 ARM PLL

This PLL synthesizes a low jitter clock from a 24 MHz reference clock. The clock output frequency for this PLL ranges from 650 MHz to 1.3 GHz. The output frequency is selected by a 7-bit register field CCM_ANALOG_PLL_ARM[DIV_SELECT].

PLL output frequency = Fref * DIV_SEL/2

NOTE

The upper frequency range may exceed the maximum frequency supported. Please see the datasheet for more information.

18.5.1.3.2 USB PLLs

These PLLs synthesize a low jitter clock from the 24 MHz reference clock. USB1 PLL has 4 frequency-programmable PFD (phase fractional divider) outputs.

The output frequency of USB1 PLL is 480 MHz. Even though USB1 PLL has a DIV_SELECT register field, this PLL should always be set to 480 MHz in normal operation. USB2 PLL is only used by the USB UTM interface through a direct connection.

18.5.1.3.3 System PLL

This PLL synthesizes a low jitter clock from the 24 MHz reference clock. The PLL has one output clock, plus 4 PFD outputs. The System PLL supports spread spectrum modulation for use in applications to minimize radiated emissions. The spread spectrum PLL output clock is frequency modulated so that the energy is spread over a wider bandwidth, thereby reducing peak radiated emissions. Due to this feature support, the associated lock time of this PLL is longer than other PLLs in the SoC that do not support spread spectrum modulation.

Spread spectrum operation is controlled by configuring the CCM_ANALOG_PLL_SYS_SS register. When enabled, the PLL output frequency will decrease by the amount defined in the STEP field, until it reaches the limiting frequency in the STOP field. The frequency will then similarly return to the original nominal frequency. The following equations control the spread-spectrum operation:

$$\text{Spread spectrum range} = \text{Fref} \times \frac{\text{CCM_ANALOG_PLL_SYS_SS[STOP]}}{\text{CCM_ANALOG_PLL_SYS_DENOM[B]}}$$

$$\text{Modulation frequency} = \text{Fref} \times \frac{\text{CCM_ANALOG_PLL_SYS_SS[STEP]}}{2 \times \text{CCM_ANALOG_PLL_SYS_SS[STOP]}}$$

Although this PLL does have a DIV_SELECT register field, it is intended that this PLL will only be run at the default frequency of 528 MHz.

18.5.1.3.4 Audio / Video PLL

The audio PLL and video PLL each synthesize a low jitter clock from a 24 MHz reference clock. The clock output frequency range for this PLL is from 650 MHz to 1.3 GHz. It has a Fractional-N synthesizer.

There are /1, /2, /4, /8, /16 post dividers for the Video PLL and /1, /2, /4 , /8, /16 post dividers for the Audio PLL. The output frequency can be set by programming the fields in the CCM_ANALOG_PLL_AUDIO, CCM_ANALOG_PLL_VIDEO, and CCM_ANALOG_MISC2 register sets according to the following equation.

PLL output frequency = Fref * (DIV_SELECT + NUM/DENOM)

18.5.1.3.5 Ethernet PLL

This PLL synthesizes a low jitter clock from the 24 MHz reference clock.

The PLL outputs a clock. The reference clocks generated by this PLL are:

- ref_enetpll0 programmable to 25, 50, 100 and 125 MHz by setting CCM_ANALOG_PLL_ENET[DIV_SELECT] bitfield
- ref_enetpll1 programmable to 25, 50, 100 and 125 MHz by setting CCM_ANALOG_PLL_ENET[DIV_SELECT] bitfield
- ref_enetpll2 fixed at 25 MHz

18.5.1.4 Phase Fractional Dividers (PFD)

There are several PFD outputs from the System PLL and USB1 PLL.

Each PFD output generates a fractional multiplication of the associated PLL's VCO frequency. Where the output frequency is equal to Fvco*18/N, N can range from 12-35. The PFDs allow for clock frequency changes without forcing the relock of the root PLL. This feature is useful in support of dynamic voltage and frequency scaling (DVFS). See CCM Analog Memory Map/Register Definition.

When the related PLL is powered up from the power down state or made to go through a relock cycle due to PLL regrogramming, it is required that the related PFDx_CLKGATE bit in CCM_ANALOG_PFD_480n or CCM_ANALOG_PFD_528n, be cycled on and off (1 to 0) after PLL lock. The PFDs can be in the clock gated state during PLL relock but must be un-clock gated only after lock is achieved. See the engineering bulletin, *Configuration of Phase Fractional Dividers (EB790)* at www.freescale.com for procedure details.

18.5.1.5 CCM internal clock generation

The clock generation is comprised of two sub-modules:

CCM_CLK_SWITCHER

CCM_CLK_ROOT_GEN

18.5.1.5.1 Clock Switcher

The Clock Switcher (CCM_CLK_SWITCHER) sub-module receives the PLL output clocks and the PLL bypass clocks.

Figure 18-4 describes the generation of the three switcher clocks.

The figure also includes the Frequency Switch Control sub-module responsible for frequency change.

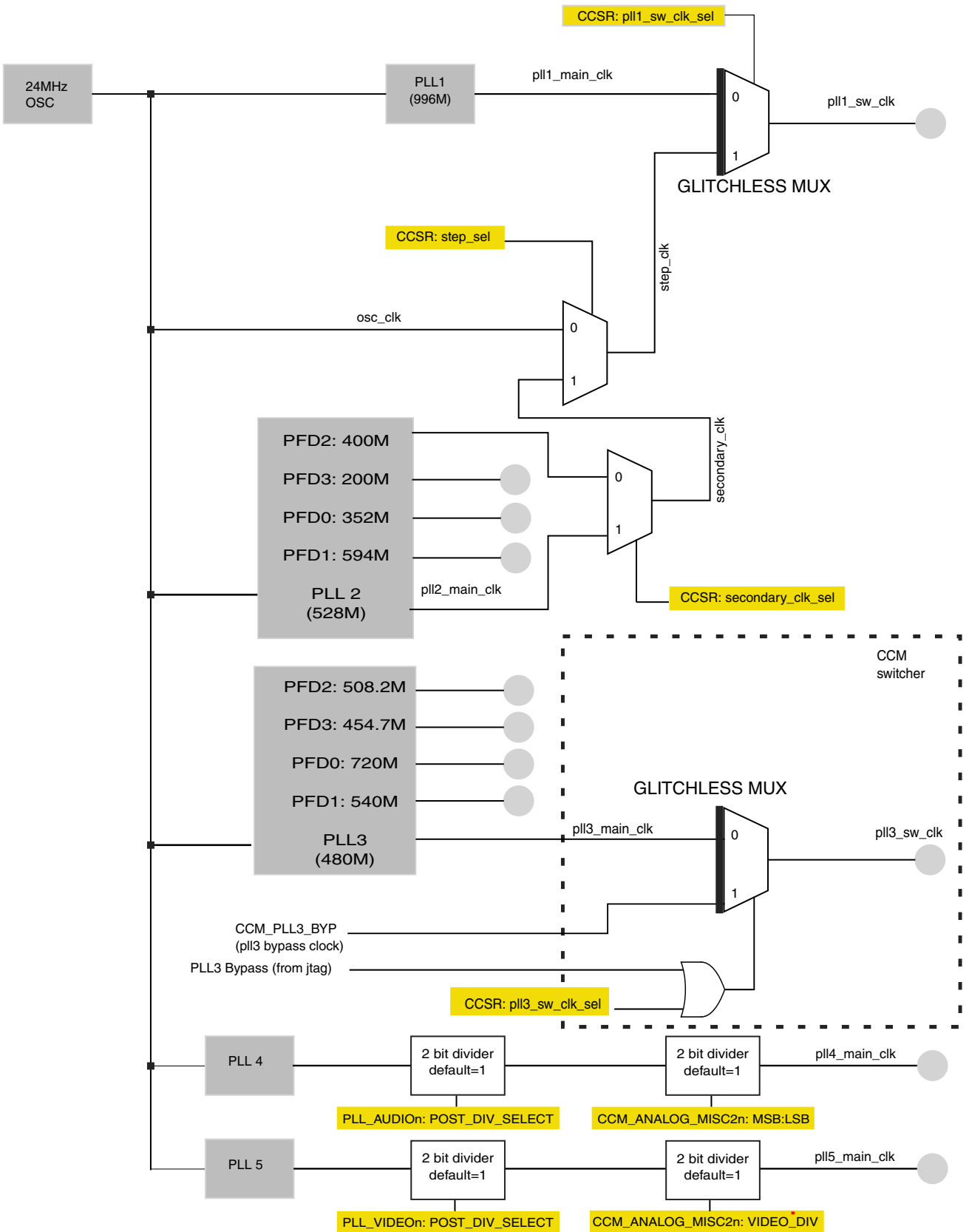


Figure 18-4. Switcher clock generation

18.5.1.5.2 PLL bypass procedure

In addition to PLL bypass options in CCM_ANALOG module, switcher and clk_root_gen sub-modules includes capability for each of the PLL clocks to be bypassed with an external bypass clock.

18.5.1.5.3 PLL clock change

In order to modify or stop the clock output of a specific PLL, all the clocks generated from the current PLL must be transitioned to the new PLL whose frequency is not being modified.

For clocks which can't be stopped (core and bus clocks), this should be done via the glitchless mux. Before changing the PLL setting, power it down. Power up the PLL after the change. See Disabling / Enabling PLLs for more information.

18.5.1.5.4 Clock Root Generator

The Clock Root Generator (CCM_CLK_ROOT_GEN) sub-module generates the root clocks to be delivered to LPCG.

The following figures describe clock generation. The frequencies in parantheses are the default typical frequencies.

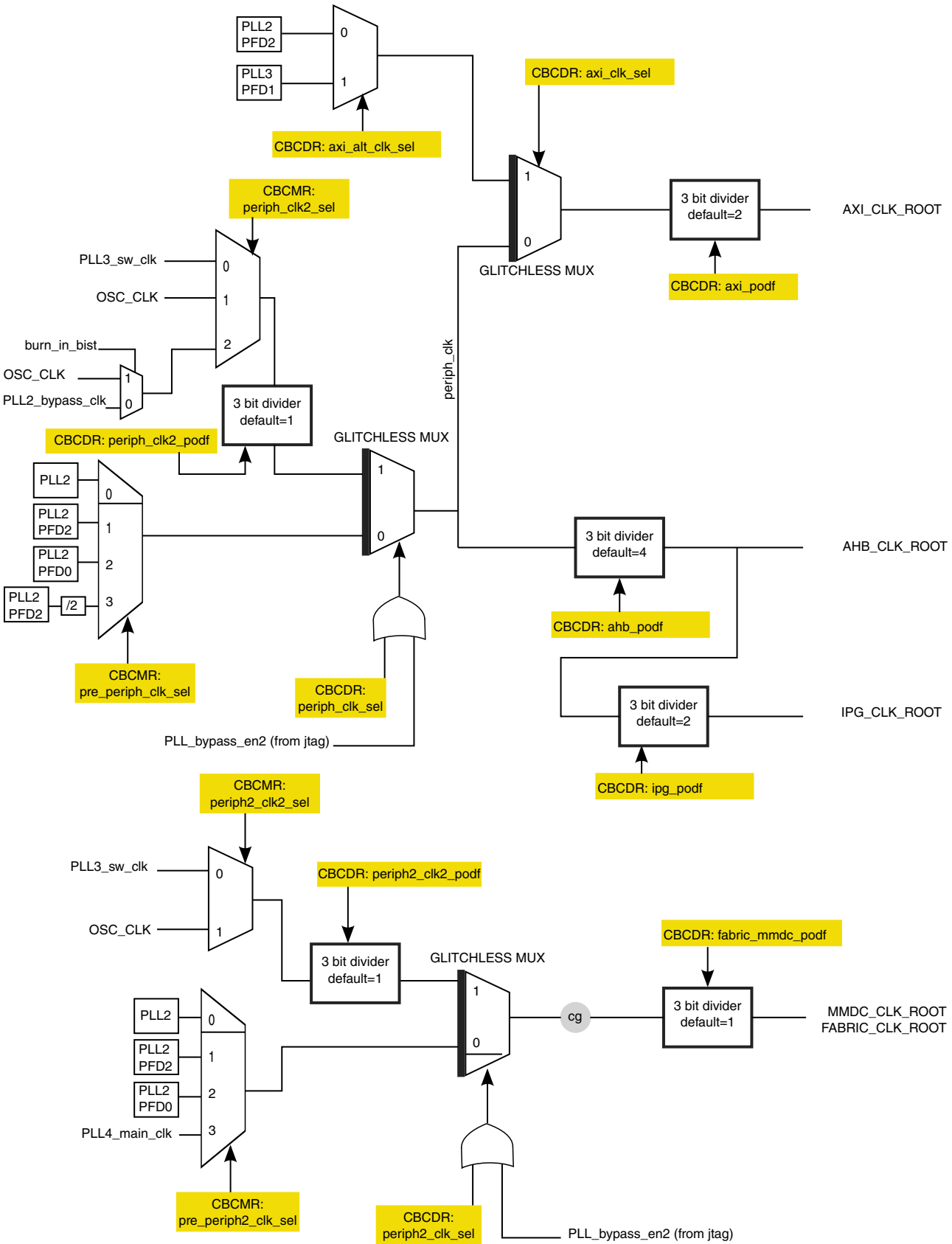


Figure 18-5. BUS clock generation

NOTE

All 6-bit PODF dividers found in the diagrams above can operate on low frequency.

18.5.1.5.5 Initial values controlled by the System JTAG Controller (SJC).

The initial values of the following dividers and muxes can be controlled by SJC.

In regular functional mode, the SJC will drive the reset values stated in the CCM register memory map. If SJC is programmed to change those values, then the reset value for those dividers/muxes will be taken from the SJC programability.

Software can update the changed reset value after reset sequence. The control signals and the dividers/muxes are listed below:

- [2:0] init_periph2_clk2_podf
- [1:0] init_ipg_podf
- [2:0] init_ahb_podf
- [2:0] init_axi_podf
- [2:0] init_periph_clk2_podf
- init_periph_clk_sel
- init_periph2_clk_sel

18.5.1.5.6 Divider change handshake

Modifying the following dividers will start the handshake with MMDC.

- mmdc_podf
- periph_clk_sel
- periph2_clk_sel
- arm_podf
- ahb_podf

18.5.1.6 Disabling / Enabling PLLs

PLL disabling and enabling is done via analog module.

Before disabling a PLL using the analog registers, software should first move all the clocks generated from that specific PLL to another source. This alternate source could be another PLL, or a PFD driven by another PLL. Alternatively, software can bypass the PLL and use the PLL reference clock (usually 24 MHz) as the output clock. Bypassing the PLL is done by setting the analog BYPASS bit in the control register for that PLL.

18.5.1.7 Clock Switching Multiplexers

There are a multitude of multiplexers available throughout the clock generation logic that provide alternate clock sources for the system clocks controlled by the CCM. The CCM utilizes several synchronous glitchless clock multiplexers as well as asynchronous glitchy clock multiplexers.

Synchronous muxes ensure there are no glitches between the transition of two asynchronous clocks and that there will be no pulses that are of a frequency higher than either input clock. In order for the synchronous multiplexer to work properly, both the current clock and the clock to be selected must remain active during the entire selection process.

There are five glitchless (synchronous) muxes used in the CCM. The table below lists the muxes and the respective control bits.

Table 18-5. Glitchless Multiplexers

Glitchless Mux	Mux Select Bit	Handshake Bit
periph_clk_mux	CBCDR[periph_clk_sel]	CDHIPR[periph_clk_sel_busy]
periph2_clk_mux	CBCDR[periph2_clk_sel]	CDHIPR[periph2_clk_sel_busy]
axi_alt_clk_mux	CBCDR[axi_sel]	
pll3_sw_clk_mux	CCSR[pll3_sw_clk_sel]	
pll1_sw_clk_mux	CCSR[pll1_sw_clk_sel]	

NOTE

Any change of the periph_clk_sel and periph2_clk_sel sync mux select will involve handshake with the MMDC. Refer to the CCDR and CDHIPR registers for the handshake bypass and busy bits.

For critical system bus clocks, changing the clock source can be done in the CCM using the glitchless clock muxes in . In the figure, the thick bar on the input side indicates the glitchless muxes. Those without the thick bar are regular muxes (not glitchless).

For example, before disabling PLL2, software can switch the FABRIC_CLK_ROOT away from the PLL2 or one of its PFDs by programming CBCMR[PERIPH2_CLK2_SEL] and CBCDR[PERIPH2_CLK2_PODF] to provide an appropriate frequency clock, then glitchlessly switch to it by programming CBCDR[PERIPH2_CLK_SEL].

Asynchronous multiplexers or glitchy multiplexers, allow the clock to switch immediately after the multiplexer select changed. This immediate switch of two asynchronous clock domains can cause the output clock to glitch. Since both clock sources to the mux are asynchronous, switching the clocks from one source to the other can cause a glitch to be generated, regardless of the input clock source.

The input clocks to the mux are required to be gated before switching the source clock in the CCM clock mux and the output should also be gated. If the input and output clocks are not gated, clock glitches can propagate to the logic that follows the clock mux, causing the logic to behave unpredictably.

For serial clocks, software should first disable the module, then gate its clock in the LPCG. Then it should move the mux controlling the source of the clocks to another PLL, and reset the module and its clocks. Only then is it safe to disable the PLL. The mux for the serial clocks is not glitchless.

18.5.1.8 Low Power Clock Gating module (LPCG)

The LPCG module receives the root clocks and splits them to clock branches for each module. The clock branches are gated clocks.

The enables for those gates can come from four sources:

1. Clock enable signal from CCM - this signal is generated by configuring of the CGR bits in the CCM. It is based on the low power mode.
2. Clock enable signal from the module - this signal is generated by the module based on internal logic of the module. Not every enable signal from the module is used. For used clock enable signals from the module, CCM will generate an override signal based on a programable bit in CCM (CMEOR).
3. Clock enable signal from Reset controller (SRC) - this signal will enable the clock during the reset procedure. Please see the SRC chapter for details on the clock enable signal during reset procedure.
4. Hard-coded enable from fuse box.

These enable signals are ANDed to generate the enable signal for the gating cell.

The enable signal for the gating cell is synchronized with the clock it needs to gate in order to prevent glitches on the gated clock.

Notifications are generated for CCM to indicate when clock roots should be opened and closed. All notifications that correspond to the same clock root will be ORed to generate one notification signal to CCM for clock root gating.

The following figure describes the clock split inside the LPCG module. It describes the case of two modules; one module is without an enable signal and one is shown with an enable signal. SRC enable signals and sync flip flops are omitted from this figure.

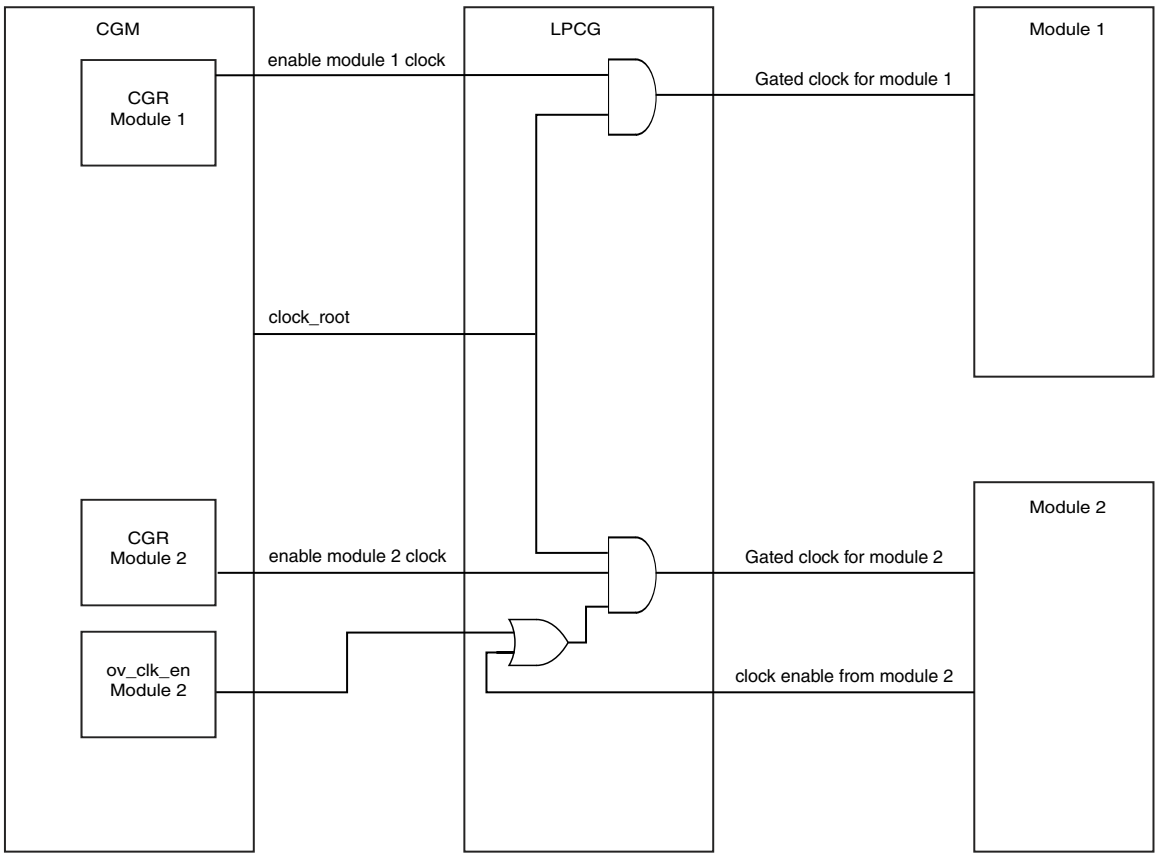


Figure 18-6. Clock split in LPCG

18.5.1.8.1 MMDC handshake

CCM will assert the mmdc_freq_change_req signal.

MMDC will assert the signals to acknowledge that the frequency change request has been received and that the frequency can now be changed safely.

CCM will commence the actual change of division ratio of mmdc dividers or apply mux change on root clocks once both of the non-masked acknowledges are asserted.

NOTE

MMDC handshakes can be masked.

18.5.2 DVFS support

When performing DVFS, the frequency shift procedure for the ARM core clock domain can be performed by software.

CPU PLL frequency and CCM ARM clock divider is controlled by CCM and CPU power domain supply voltage value is controlled by CCM_ANALOG module.

NOTE

The frequency should be shifted down first and then voltage value reduced, and vice-versa, when shifting the frequency up.

NOTE

CCM_ANALOG will not control the voltage value in Bypass mode

18.5.3 Power modes

The chip supports 3 low power modes: RUN mode, WAIT mode, STOP mode.

18.5.3.1 RUN mode

This is the normal/functional operating mode. In this mode, the CPU runs in its normal operational mode. Clocks to the modules can be gated by configuring the corresponding CCGRx bits.

18.5.3.2 WAIT mode

In this mode the CPU clock is gated. All other clocks are functional and can be gated by programming their CGR bits when all ARM cores are in WFI, and L2 cache and SCU are idle.

18.5.3.2.1 Entering WAIT mode

If the CLPCR[LPM] bit is set by software to WAIT mode, when CPU executes the next wait for interrupt (WFI) instruction, WAIT mode sequence will start.

As part of the WFI routine, alternative interrupt controller in GPC should be updated; the CPU platform interrupt controller will be disabled first by software and will be not functional, due to clock gating. Interrupts during WAIT mode are monitored by alternative interrupt controller.

After execution of the WFI routine, the CPU platform will assert idle signals for each component of the platform and CCM will gate clock to the platform.

The next actions can be programmed during WAIT mode:

1. CCM requests an acknowledge to close clocks to MMDC if its CGR bits indicate to close its clocks on WAIT mode, and if those clocks are not already closed in run mode. The request will be issued if the handshake is not bypassed by programing the CLPCR register. If the corresponding bits are set, the request signal will not be issued to the corresponding module and CCM will not wait for its acknowledge in the process of entering low power mode. Once CCM receives all the acknowledge signals needed, then it will enter WAIT mode.
2. Close the clocks to the modules which were defined to be shut at WAIT mode in the CCGR bits.
3. Observability to indicate WAIT mode.

NOTE

Setting MMDC CGR bits to 01 can hang the entire system since the MMDC clock and fabric clock share the same clock root.

Any enabled interrupt assertion will start the exit from WAIT mode.

18.5.3.2.2 Exiting WAIT mode

As soon as enabled interrupt is asserted, CPU supply will be restored if CPU SRPG was applied and clocks are enabled to CPU and other modules.

18.5.3.3 STOP mode

In this mode all system clocks are stopped, along with the CPU, system buses and all PLLs. Power gating can be applied for ARM platform. External supply voltage can be reduced to decrease leakage.

18.5.3.3.1 Entering STOP mode

Procedure entering STOP mode is the same, as entering WAIT mode until the moment of disabling clocks to modules. (LPM bit should be configured to STOP mode.)

After clocks to modules are gated, the following actions will be taken:

- PLLs are disabled
- CCM_PMIC_STBY_REQ asserted, if vstby bit is set
- osc_en signal is negated
- osc_pwrdn is asserted, if sbyos bit is set

Counter will be triggered after CCM_PMIC_STBY_REQ assertion to allow to external regulator or PMIC to decrease voltage until valid voltage range. On counter completion, stop_mode signal will be asserted, that will trigger disabling analog elements in anatop.

CCM's low power state machine will remain in state STOP_GPC until STOP mode is exited.

18.5.3.3.2 Exiting STOP mode

As soon as an enabled interrupt is asserted, the CCM will begin the process of exiting STOP mode.

The following will take place:

1. If vstby bit was set, deassert PMIC_STBY_REQ to notify power management IC to change voltage from standby voltage to functional voltage.
2. If sbyos was set, and CCM closed either external oscillator or on board oscillator, then CCM will start oscillator by asserting ref_en_b signal and deasserting cosc_pwrdown signal respectively.
3. After the number of cycles of CKILs defined in stby_count bits, wait until PMIC functional voltage is ready. This is the notification from power management IC that the voltage is ready at its functional value. Only then will CCM continue the steps.
4. Start osc. If oscillator was started, wait until oscnt has finished its counting to make sure that oscillator is ready.
5. Start PLLs. Only the PLLs that were configured to be on prior to the entrance to STOP mode will be started.
6. CCM will request GPC to restore ARM power by GPC_PUP_REQ. If power was removed from the ARM platform, GPC will notify CCM by asserting signal GPC_PUP_ACK that power to ARM is back on, and its safe to exit from STOP mode. Only then will the CCM progress to the next step.
7. Once assertion of notification from src that the resets for the power gated modules has been finished, (src_power_gating_reset_done is set) negate the low power request signals to all modules and enable all module clocks including ARM clocks and CKIL sync, and return to run mode. (Clocks whose CCGR bits are not to be opened in RUN mode will not be opened; they will continued to be gated.)

Once the system is in run mode, negate signals ccm_ipg_stop and system_in_stop_mode.

18.6 CCM Memory Map/Register Definition

NOTE

The register reset values for CCM change depending on the boot configuration. See Clocks at Boot Time for more information.

CCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_4000	CCM Control Register (CCM_CCR)	32	R/W	0401_167Fh	18.6.1/639
20C_4004	CCM Control Divider Register (CCM_CCDR)	32	R/W	0000_0000h	18.6.2/640
20C_4008	CCM Status Register (CCM_CSR)	32	R	0000_0010h	18.6.3/642
20C_400C	CCM Clock Switcher Register (CCM_CCSR)	32	R/W	0000_0100h	18.6.4/643
20C_4010	CCM Arm Clock Root Register (CCM_CACRR)	32	R/W	0000_0000h	18.6.5/644
20C_4014	CCM Bus Clock Divider Register (CCM_CBCDR)	32	R/W	0001_8D00h	18.6.6/645
20C_4018	CCM Bus Clock Multiplexer Register (CCM_CBCMR)	32	R/W	2486_0324h	18.6.7/648
20C_401C	CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)	32	R/W	0490_0080h	18.6.8/650
20C_4020	CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)	32	R/W	0319_2F06h	18.6.9/652
20C_4024	CCM Serial Clock Divider Register 1 (CCM_CSCDR1)	32	R/W	0049_0B00h	18.6.10/ 654
20C_4028	CCM SAI1 Clock Divider Register (CCM_CS1CDR)	32	R/W	0EC1_02C1h	18.6.11/ 656
20C_402C	CCM SAI2 Clock Divider Register (CCM_CS2CDR)	32	R/W	0003_36C1h	18.6.12/ 657
20C_4030	CCM D1 Clock Divider Register (CCM_CDCDR)	32	R/W	33F7_1F92h	18.6.13/ 659
20C_4034	CCM HSC Clock Divider Register (CCM_CHSCCDR)	32	R/W	0002_48A4h	18.6.14/ 660
20C_4038	CCM Serial Clock Divider Register 2 (CCM_CSCDR2)	32	R/W	0002_9B48h	18.6.15/ 662
20C_403C	CCM Serial Clock Divider Register 3 (CCM_CSCDR3)	32	R/W	0001_4841h	18.6.16/ 663
20C_4044	CCM Wakeup Detector Register (CCM_CWDR)	32	R/W	0000_0000h	18.6.17/ 664
20C_4048	CCM Divider Handshake In-Process Register (CCM_CDHIPR)	32	R	0000_0000h	18.6.17/ 665
20C_4054	CCM Low Power Control Register (CCM_CLPCR)	32	R/W	0000_0079h	18.6.18/ 668
20C_4058	CCM Interrupt Status Register (CCM_CISR)	32	w1c	0000_0000h	18.6.19/ 670

Table continues on the next page...

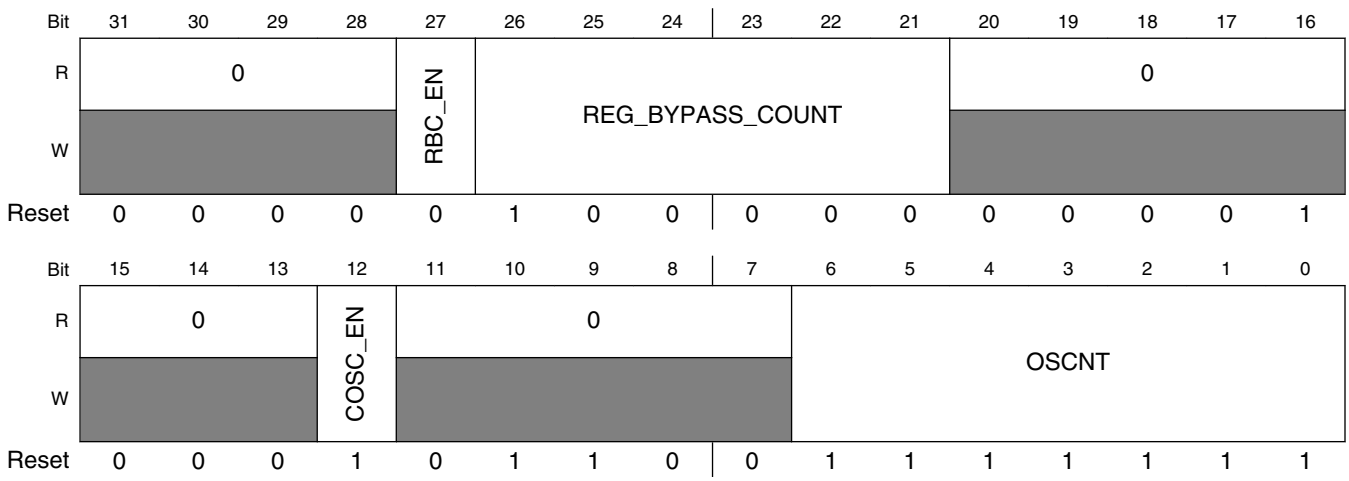
CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_405C	CCM Interrupt Mask Register (CCM_CIMR)	32	R/W	FFFF_FFFFh	18.6.20/ 673
20C_4060	CCM Clock Output Source Register (CCM_CCOSR)	32	R/W	000A_0001h	18.6.21/ 675
20C_4064	CCM General Purpose Register (CCM_CGPR)	32	R/W	0000_FE62h	18.6.22/ 677
20C_4068	CCM Clock Gating Register 0 (CCM_CCGR0)	32	R/W	FFFF_FFFFh	18.6.23/ 678
20C_406C	CCM Clock Gating Register 1 (CCM_CCGR1)	32	R/W	FFFF_FFFFh	18.6.24/ 680
20C_4070	CCM Clock Gating Register 2 (CCM_CCGR2)	32	R/W	FC3F_FFFFh	18.6.25/ 682
20C_4074	CCM Clock Gating Register 3 (CCM_CCGR3)	32	R/W	FFFF_FFFFh	18.6.26/ 683
20C_4078	CCM Clock Gating Register 4 (CCM_CCGR4)	32	R/W	FFFF_FFFFh	18.6.27/ 684
20C_407C	CCM Clock Gating Register 5 (CCM_CCGR5)	32	R/W	FFFF_FFFFh	18.6.28/ 686
20C_4080	CCM Clock Gating Register 6 (CCM_CCGR6)	32	R/W	FFFF_FFFFh	18.6.29/ 687
20C_4088	CCM Module Enable Overide Register (CCM_CMEOR)	32	R/W	FFFF_FFFFh	18.6.30/ 689

18.6.1 CCM Control Register (CCM_CCR)

The figure below represents the CCM Control Register (CCR), which contains bits to control general operation of CCM. The table below provides its field descriptions.

Address: 20C_4000h base + 0h offset = 20C_4000h



CCM_CCR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27 RBC_EN	Enable for REG_BYPASS_COUNTER. If enabled, analog_reg_bypass signal will be asserted after REG_BYPASS_COUNT clocks of CKIL, after standby voltage is requested. If standby voltage is not requested analog_reg_bypass won't be asserted, event if counter is enabled. 1 REG_BYPASS_COUNTER enabled. 0 REG_BYPASS_COUNTER disabled
26–21 REG_BYPASS_ COUNT	Counter for analog_reg_bypass signal assertion after standby voltage request by PMIC_STBY_REQ. Should be zeroed and reconfigured after exit from low power mode. REG_BYPASS_COUNT can also be used for holding off interrupts when the PGC unit is sending signals to power gate the core. 000000 no delay 000001 1 CKIL clock period delay 111111 63 CKIL clock periods delay
20–13 Reserved	This read-only field is reserved and always has the value 0.
12 COSC_EN	On chip oscillator enable bit - this bit value is reflected on the output cosc_en. The system will start with on chip oscillator enabled to supply source for the PLLs. Software can change this bit if a transition to the bypass PLL clocks was performed for all the PLLs. In cases that this bit is changed from '0' to '1' then CCM will enable the on chip oscillator and after counting oscnt ckil clock cycles it will notify that on chip

Table continues on the next page...

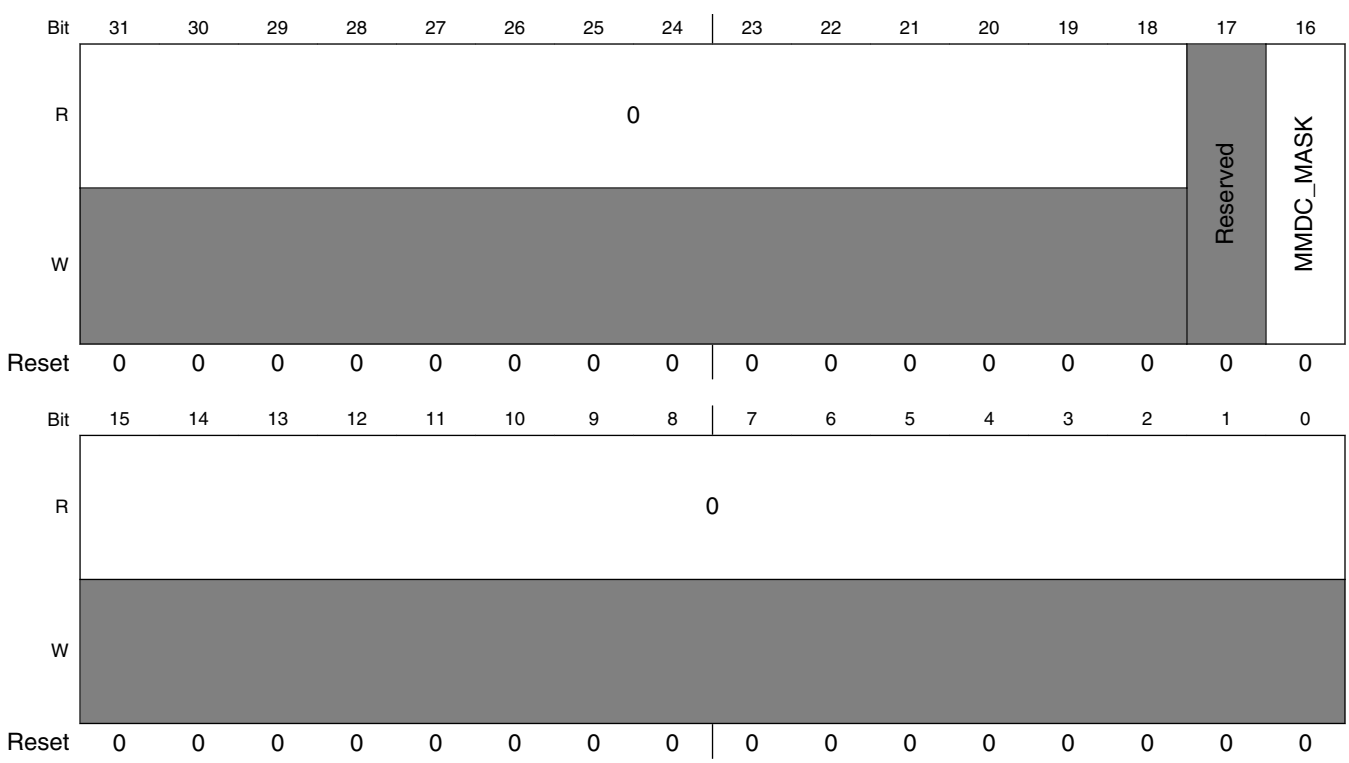
CCM_CCR field descriptions (continued)

Field	Description
	oscillator is ready by a interrupt cosc_ready and by status bit cosc_ready. The cosc_en bit should be changed only when on chip oscillator is not chosen as the clock source. 0 disable on chip oscillator 1 enable on chip oscillator
11–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
OSCNT	Oscillator ready counter value. These bits define value of 32KHz counter, that serve as counter for oscillator lock time. This is used for oscillator lock time. Current estimation is ~5ms. This counter will be used in ignition sequence and in wake from stop sequence if sbyos bit was defined, to notify that on chip oscillator output is ready for the dpll_ip to use and only then the gate in dpll_ip can be opened. 0000000 count 1 ckil 1111111 count 128 ckil's

18.6.2 CCM Control Divider Register (CCM_CCDR)

The figure below represents the CCM Control Divider Register (CCDR), which contains bits that control the loading of the dividers that need handshake with the modules they affect. The table below provides its field descriptions.

Address: 20C_4000h base + 4h offset = 20C_4004h



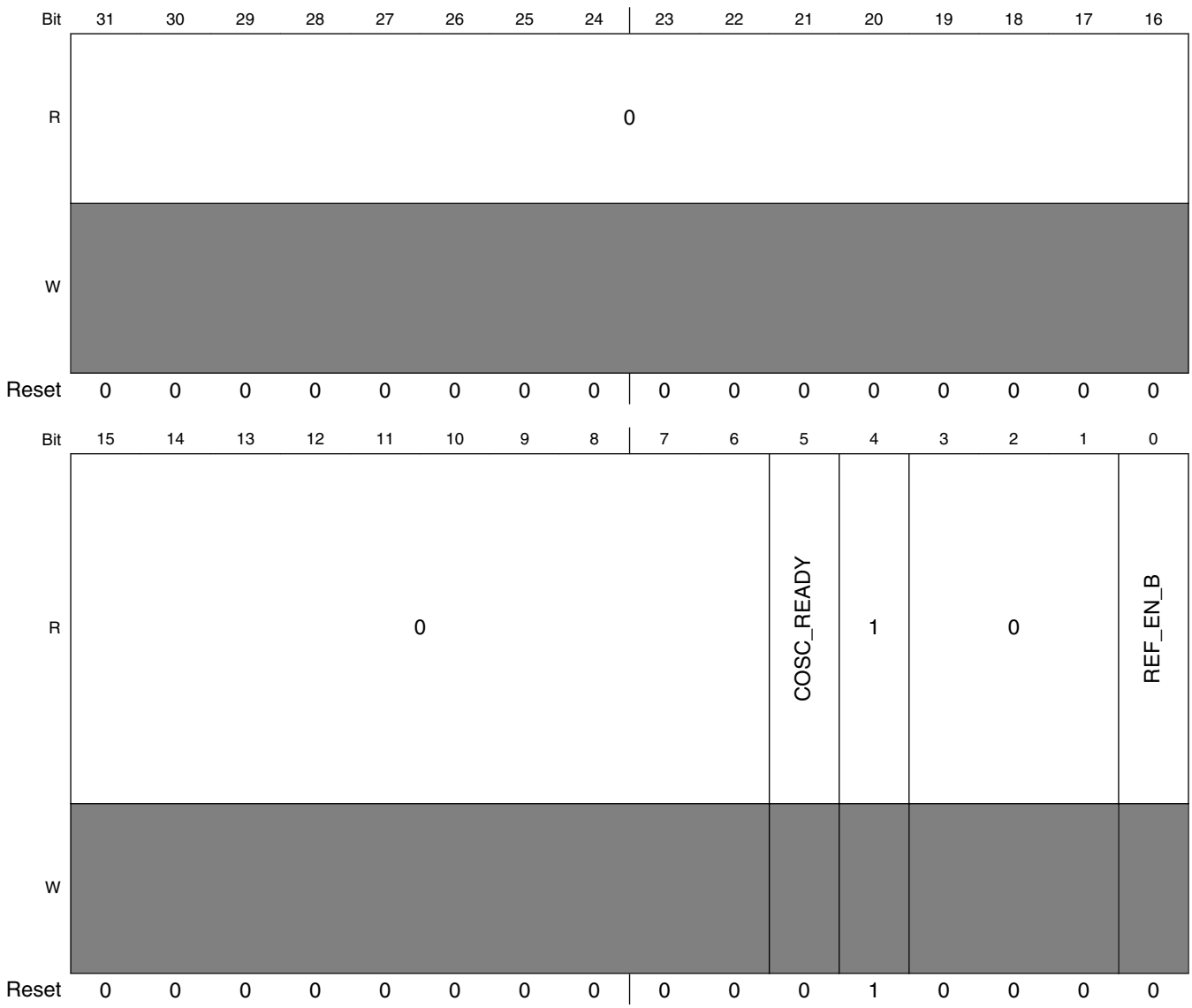
CCM_CCDR field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value 0.
17 -	This field is reserved. Reserved
16 MMDC_MASK	During divider ratio mmdc_axi_podf change or sync mux periph2_clk_sel change (but not jtag) or SRC request during warm reset, mask handshake with mmdc module. 0 allow handshake with mmdc module 1 mask handshake with mmdc. Request signal will not be generated.
Reserved	This read-only field is reserved and always has the value 0.

18.6.3 CCM Status Register (CCM_CSR)

The figure below represents the CCM status Register (CSR). The status bits are read-only bits. The table below provides its field descriptions.

Address: 20C_4000h base + 8h offset = 20C_4008h



CCM_CSR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

CCM_CSR field descriptions (continued)

Field	Description
5 COSC_READY	Status indication of on board oscillator. This bit will be asserted if on chip oscillator is enabled and on chip oscillator is not powered down, and if oscnt counter has finished counting. 0 on board oscillator is not ready. 1 on board oscillator is ready.
4 Reserved	This read-only field is reserved and always has the value 1.
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 REF_EN_B	Status of the value of CCM_REF_EN_B output of ccm 0 value of CCM_REF_EN_B is '0' 1 value of CCM_REF_EN_B is '1'

18.6.4 CCM Clock Switcher Register (CCM_CCSR)

The figure below represents the CCM Clock Switcher register (CCSR). The CCSR register contains bits to control the switcher sub-module dividers and multiplexers. The table below provides its field descriptions.

Address: 20C_4000h base + Ch offset = 20C_400Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							STEP_SEL	0				SECONDARY_CLK_SEL	PLL1_SW_CLK_SEL	Reserved	PLL3_SW_CLK_SEL
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

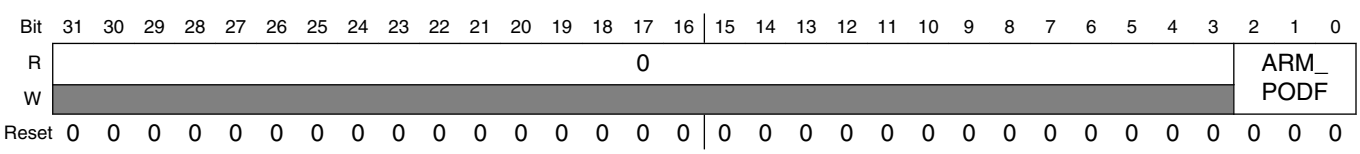
CCM_CCSR field descriptions

Field	Description
31–9 Reserved	This read-only field is reserved and always has the value 0.
8 STEP_SEL	Selects the option to be chosen for the step frequency when shifting ARM frequency. This will control the step_clk. **NOTE:** This mux is allowed to be changed only if its output is not used, i.e. ARM uses the output of pll1, and step_clk is not used. 0 derive clock from osc_clk (24M) - source for lp_apm. 1 derive clock from secondary_clk
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 SECONDARY_CLK_SEL	Select source to generate secondary_clk 0 PLL2 PFD2 (400 M) 1 PLL2 (528 M)
2 PLL1_SW_CLK_SEL	Selects source to generate pll1_sw_clk. 0 pll1_main_clk 1 step_clk
1 -	This field is reserved. Reserved
0 PLL3_SW_CLK_SEL	Selects source to generate pll3_sw_clk. This bit should only be used for testing purposes. 0 pll3_main_clk 1 pll3 bypass clock

18.6.5 CCM Arm Clock Root Register (CCM_CACRR)

The figure below represents the CCM Arm Clock Root register (CACRR). The CACRR register contains bits to control the ARM clock root generation. The table below provides its field descriptions.

Address: 20C_4000h base + 10h offset = 20C_4010h



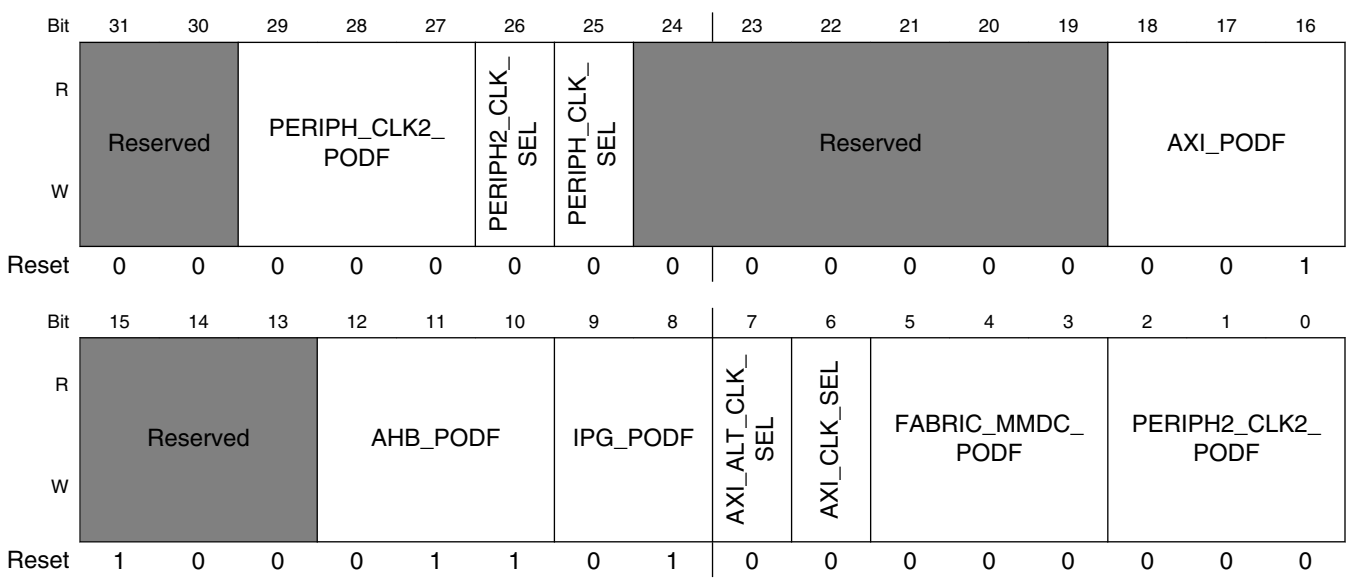
CCM_CACRR field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
ARM_PODF	Divider for ARM clock root. **NOTE:** If arm_freq_shift_divider is set to '1' then any new write to arm_podf will be held until arm_clk_switch_req signal is asserted. **NOTE:** arm_podf should be >= 3 if fuse bit CPU_SPEED_LIMIT is set. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

18.6.6 CCM Bus Clock Divider Register (CCM_CBCDR)

The figure below represents the CCM Bus Clock Divider Register (CBCDR). The CBCDR register contains bits to control the clock generation sub module dividers. The table below provides its field descriptions.

Address: 20C_4000h base + 14h offset = 20C_4014h



CCM_CBCDR field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–27 PERIPH_CLK2_PODF	Divider for periph2 clock podf. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
26 PERIPH2_CLK_SEL	Selector for peripheral2 main clock (source of mmdc_clk_root). **NOTE:** Any change of this mux select will involve handshake with the MMDC. Refer to the CCDR and CDHIPR registers for the handshake bypass and busy bits. 0 PLL2 (pll2_main_clk) 1 derive clock from periph2_clk2_clk clock source.
25 PERIPH_CLK_SEL	Selector for peripheral main clock (source of MMDC_CLK_ROOT). **NOTE:** Alternative clock source should be used when PLL is relocked. For PLL relock procedure pls refer to the PLL chapter. **NOTE:** Any change of this sync mux select will involve handshake with the MMDC. Refer to the CCDR and CDHIPR registers for the handshake bypass and busy bits. 0 PLL2 (pll2_main_clk) 1 derive clock from periph_clk2_clk clock source.
24–19 -	This field is reserved. Reserved
18–16 AXI_PODF	Post divider for axi clock. **NOTE:** Any change of this divider might involve handshake with EMI. See CDHIPR register for the handshake busy bits. 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15–13 -	This field is reserved. Reserved
12–10 AHB_PODF	Divider for AHB PODF. **NOTE:** Any change of this divider might involve handshake with EMI. See CDHIPR register for the handshake busy bits.

Table continues on the next page...

CCM_CBCDR field descriptions (continued)

Field	Description
	000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
9–8 IPG_PODF	Divider for ipg podf. **NOTE:** SDMA module will not support ratio of 1:3 and 1:4 for ahb_clk:ipg_clk. In case SDMA is used, then those ratios should not be used. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
7 AXI_ALT_CLK_SEL	AXI alternative clock select 0 PLL2 PFD2 will be selected as alternative clock for AXI root clock 1 PLL3 PFD1 will be selected as alternative clock for AXI root clock
6 AXI_CLK_SEL	AXI clock source select 0 Periph_clk output will be used as AXI clock root 1 AXI alternative clock will be used as AXI clock root
5–3 FABRIC_MMDC_PODF	Post divider for fabric / mmdc clock. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
PERIPH2_CLK2_PODF	Divider for periph2_clk2 podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

18.6.7 CCM Bus Clock Multiplexer Register (CCM_CBCMR)

The figure below represents the CCM Bus Clock Multiplexer Register (CBCMR). The CBCMR register contains bits to control the multiplexers that generate the bus clocks. The table below provides its field descriptions.

NOTE

Any change on the above multiplexer will have to be done while the module that its clock is affected is not functional and the respective clock is gated in LPCG. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

The change for arm_clk_sel should be done through sdma so that ARM will not use this clock during the change and the clock will be gated in LPCG.

Address: 20C_4000h base + 18h offset = 20C_4018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			Reserved			LCDIF1_PODF			PRE_ PERIPH2_ CLK_SEL		PERIPH2_CLK2_SEL	PRE_ PERIPH_ CLK_SEL		Reserved	
Reset	0	0	1	0	0	1	0	0	1	0	0	0	0	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		PERIPH_ CLK2_SEL		Reserved	Reserved	Reserved		Reserved		Reserved		Reserved			
Reset	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	0

CCM_CBCMR field descriptions

Field	Description
31–29 -	This field is reserved. Reserved

Table continues on the next page...

CCM_CBCMR field descriptions (continued)

Field	Description
28–26 -	This field is reserved. Reserved
25–23 LCDIF1_PODF	Post-divider for lcdif1 clock. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
22–21 PRE_PERIPH2_ CLK_SEL	Selector for pre_periph2 clock multiplexer 00 derive clock from PLL2 01 derive clock from PLL2 PFD2 10 derive clock from PLL2 PFD0 11 derive clock from PLL4
20 PERIPH2_CLK2_ SEL	Selector for periph2_clk2 clock multiplexer 0 derive clock from pll3_sw_clk 1 derive clock fromOSC
19–18 PRE_PERIPH_ CLK_SEL	Selector for pre_periph clock multiplexer 00 derive clock from PLL2 01 derive clock from PLL2 PFD2 10 derive clock from PLL2 PFD0 11 derive clock from divided (/2) PLL2 PFD2
17–14 -	This field is reserved. Reserved
13–12 PERIPH_CLK2_ SEL	Selector for peripheral clk2 clock multiplexer 00 derive clock from pll3_sw_clk 01 derive clock from osc_clk (pll1_ref_clk) 10 derive clock from pll2_bypass_clk 11 reserved
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 -	This field is reserved. Reserved
7–6 -	This field is reserved. Reserved
5–4 -	This field is reserved. Reserved
-	This field is reserved. Reserved

18.6.8 CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)

The figure below represents the CCM Serial Clock Multiplexer Register 1 (CSCMR1). The CSCMR1 register contains bits to control the multiplexers that generate the serial clocks. The table below provides its field descriptions.

NOTE

Any change on the above multiplexer will have to be done while the module that its clock is affected is not functional and the clock is gated. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

Address: 20C_4000h base + 1Ch offset = 20C_401Ch

Bit	31	30 29	28 27 26	25 24	23	22 21 20	19	18	17	16
R	0	ACLK_EIM_SLOW_SEL	QSPI1_PODF	ACLK_EIM_SLOW_PODF		Reserved	GPMI_CLK_SEL	BCH_CLK_SEL	USDHC2_CLK_SEL	USDHC1_CLK_SEL
W										
Reset	0	0 0	0 0 0	0 0	1	0 0 1	0	0	0	0

Bit	15 14	13 12	11 10	9 8	7	6	5 4 3 2 1 0
R	SAI3_CLK_SEL	SAI2_CLK_SEL	SAI1_CLK_SEL	QSPI1_CLK_SEL		PERCLK_CLK_SEL	PERCLK_PODF
W							
Reset	0 0	0 0	0 0	0 0	1	0	0 0 0 0 0 0

CCM_CSCMR1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–29 ACLK_EIM_SLOW_SEL	Selector for aclk_eim_slow root clock multiplexer 00 derive clock from AXI 01 derive clock from pll3_sw_clk 10 derive clock from PLL2 PFD2 11 derive clock from PLL3 PFD0
28–26 QSPI1_PODF	Divider for QSPI1 clock root 000 divide by 1

Table continues on the next page...

CCM_CSCMR1 field descriptions (continued)

Field	Description
	001 divide by 2 111 divide by 8
25–23 ACLK_EIM_SLOW_PODF	Divider for aclk_eim_slow clock root. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
22–20 -	This field is reserved. Reserved
19 GPMI_CLK_SEL	Selector for gpmi clock multiplexer 0 derive clock from PLL2 PFD2 1 derive clock from PLL2 PFD0
18 BCH_CLK_SEL	Selector for bch clock multiplexer 0 derive clock from PLL2 PFD2 1 derive clock from PLL2 PFD0
17 USDHC2_CLK_SEL	Selector for usdhc2 clock multiplexer 0 derive clock from PLL2 PFD2 1 derive clock from PLL2 PFD0
16 USDHC1_CLK_SEL	Selector for usdhc1 clock multiplexer 0 derive clock from PLL2 PFD2 1 derive clock from PLL2 PFD0
15–14 SAI3_CLK_SEL	Selector for sai3 clock multiplexer 00 derive clock from PLL3 PFD2 01 derive clock from PLL5 10 derive clock from PLL4 11 Reserved
13–12 SAI2_CLK_SEL	Selector for sai2 clock multiplexer 00 derive clock from PLL3 PFD2 01 derive clock from PLL5 10 derive clock from PLL4 11 Reserved
11–10 SAI1_CLK_SEL	Selector for sai1 clock multiplexer 00 derive clock from PLL3 PFD2 01 derive clock from PLL5 10 derive clock from PLL4 11 Reserved

Table continues on the next page...

CCM_CSCMR1 field descriptions (continued)

Field	Description
9–7 QSPI1_CLK_SEL	QSPI1 clock select 000 Derive clock from PLL3 001 Derive clock from PLL2 PFD0 010 Derive clock from PLL2 PFD2 011 Derive clock from PLL2 100 Derive clock from PLL3 PFD3 101 Derive clock from PLL3 PFD2
6 PERCLK_CLK_SEL	Selector for the perclk clock multiplexor 0 derive clock from ipg clk root 1 derive clock from osc_clk
PERCLK_PODF	Divider for perclk podf. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

18.6.9 CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)

The figure below represents the CCM Serial Clock Multiplexer Register 2 (CSCMR2). The CSCMR2 register contains bits to control the multiplexers that generate the serial clocks. The table below provides its field descriptions.

NOTE

Any change on the above multiplexer will have to be done while the module that its clock is affected is not functional and the clock is gated. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

Address: 20C_4000h base + 20h offset = 20C_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			VID_CLK_PODF			VID_CLK_ PRE_PODF		VID_CLK_SEL			Reserved		Reserved		
Reset	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				LDB_DI1_DIV	LDB_DI0_DIV	CAN_CLK_SEL		CAN_CLK_PODF						Reserved	
Reset	0	0	1	0	1	1	1	1	0	0	0	0	0	1	1	0

CCM_CSCMR2 field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28–26 VID_CLK_PODF	Post-divider for vid clock root 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
25–24 VID_CLK_PRE_PODF	Post-divider for vid clock root 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
23–21 VID_CLK_SEL	Selector for vid clock multiplexer 000 PLL3 PFD1 001 PLL3 010 PLL3 PFD3 011 PLL4 100 PLL5
20–19 -	This field is reserved. Reserved
18–12 -	This field is reserved. Reserved
11 LDB_DI1_DIV	Control for divider of ldb clock for di1 0 divide by 3.5 1 divide by 7
10 LDB_DI0_DIV	Control for divider of ldb clock for di0 0 divide by 3.5 1 divide by 7
9–8 CAN_CLK_SEL	Selector for FlexCAN clock multiplexer 00 derive clock from pll3_sw_clk divided clock (60M) 01 derive clock from osc_clk (24M)

Table continues on the next page...

CCM_CSCMR2 field descriptions (continued)

Field	Description
	10 derive clock from pll3_sw_clk divided clock (80M) 11 Disable FlexCAN clock
7–2 CAN_CLK_PODF	Divider for can clock podf. 000000 divide by 1 000111 divide by 8 111111 divide by 2^6
-	This field is reserved. Reserved

18.6.10 CCM Serial Clock Divider Register 1 (CCM_CSCDR1)

The figure below represents the CCM Serial Clock Divider Register 1 (CSCDR1). The CSCDR1 register contains bits to control the clock generation sub-module dividers. The table below provides its field descriptions.

NOTE

Any change on the above dividers will have to be done while the module that its clock is affected is not functional and the affected clock is gated. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

Address: 20C_4000h base + 24h offset = 20C_4024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				Reserved				GPMI_PODF		BCH_PODF			USDHC2_PODF		
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		USDHC1_PODF			Reserved			UART_CLK_SEL		UART_CLK_PODF					
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

CCM_CSCDR1 field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27–25 -	This field is reserved. Reserved
24–22 GPMI_PODF	Divider for gpmi clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
21–19 BCH_PODF	Divider for bch clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
18–16 USDHC2_PODF	Divider for usdhc2 clock. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
15–14 -	This field is reserved. Reserved
13–11 USDHC1_PODF	Divider for usdhc1 clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4

Table continues on the next page...

CCM_CSCDR1 field descriptions (continued)

Field	Description
	100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
10–7 -	This field is reserved. Reserved.
6 UART_CLK_SEL	Selector for the UART clock multiplexor 0 derive clock from pll3_80m 1 derive clock from osc_clk
UART_CLK_ PODF	Divider for uart clock podf. 000000 divide by 1 111111 divide by 2^6

18.6.11 CCM SAI1 Clock Divider Register (CCM_CS1CDR)

The figure below represents the CCM SAI1, and SAI3 Clock Divider Register (CS1CDR). The CS1CDR register contains bits to control the SAI1 and SAI3 clock generation dividers. The table below provides its field descriptions.

Address: 20C_4000h base + 28h offset = 20C_4028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Reserved			SAI3_CLK_PRED			SAI3_CLK_PODF						0				Reserved			SAI1_CLK_PRED			SAI1_CLK_PODF					
W																																
Reset	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1

CCM_CS1CDR field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–25 -	This field is reserved. Reserved
24–22 SAI3_CLK_ PRED	Divider for sai3 clock pred. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5

Table continues on the next page...

CCM_CS1CDR field descriptions (continued)

Field	Description
	101 divide by 6 110 divide by 7 111 divide by 8
21–16 SAI3_CLK_ PODF	Divider for sai3 clock podf. The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6
15–12 Reserved	This read-only field is reserved and always has the value 0.
11–9 -	This field is reserved. Reserved
8–6 SAI1_CLK_ PRED	Divider for sai1 clock pred. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
SAI1_CLK_ PODF	Divider for sai1 clock podf. The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6

18.6.12 CCM SAI2 Clock Divider Register (CCM_CS2CDR)

The figure below represents the CCM SAI2, LDB Clock Divider Register (CS2CDR). The CS2CDR register contains bits to control the SAI2 clock generation dividers, and ldb serial clocks select. The table below provides its field descriptions.

Address: 20C_4000h base + 2Ch offset = 20C_402Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					ENFC_CLK_PODF						ENFC_ CLK_ PRED			ENFC_ CLK_SEL		Reserved				LDB_ DI0_ CLK_SEL			SAI2_ CLK_ PRED			SAI2_CLK_PODF					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	1	1	0	0	0	0	0	1

CCM_CS2CDR field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26–21 ENFC_CLK_ PODF	Divider for enfc clock divider. 000000 divide by 1 000001 divide by 2 111111 divide by 2^6
20–18 ENFC_CLK_ PRED	Divider for enfc clock pred divider. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
17–15 ENFC_CLK_SEL	Selector for enfc clock multiplexer **NOTE:** Multiplexor should be updated when output clock is gated. 000 derive clock from PLL2 PFD0 001 derive clock from PLL2 010 derive clock from pll3_sw_clk 011 derive clock from PLL2 PFD2 100 derive clock from PLL3 PFD3 101-111 Reserved
14–12 -	This field is reserved. Reserved
11–9 LDB_DI0_CLK_ SEL	Selector for ldb_di0 clock multiplexer **NOTE:** Multiplexor should be updated when both input and output clocks are gated. 000 PLL5 clock 001 PLL2 PFD0 010 PLL2 PFD2 011 PLL2 PFD3 100 PLL2 PFD1 101 PLL3 PFD3 110 Reserved 111 Reserved
8–6 SAI2_CLK_ PRED	Divider for sai2 clock pred. **NOTE:** Divider should be updated when output clock is gated.

Table continues on the next page...

CCM_CS2CDR field descriptions (continued)

Field	Description
	000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
SAI2_CLK_PODF	Divider for sai2 clock podf. The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. **NOTE:** Divider should be updated when output clock is gated. 000000 divide by 1 111111 divide by 2^6

18.6.13 CCM D1 Clock Divider Register (CCM_CDCDR)

The figure below represents the CCM DI Clock Divider Register (CDCDR). The table below provides its field descriptions.

Address: 20C_4000h base + 30h offset = 20C_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				SPDIF0_CLK_PRED				SPDIF0_CLK_PODF		SPDIF0_CLK_SEL		Reserved			
Reset	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	1

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

CCM_CDCDR field descriptions

Field	Description
31–28 -	This field is reserved. Reserved 0 derive from pll3_120M clock 1 derive clock from PLL2 PFD2
27–25 SPDIF0_CLK_PRED	Divider for spdif0 clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 (do not use with high input frequencies)

Table continues on the next page...

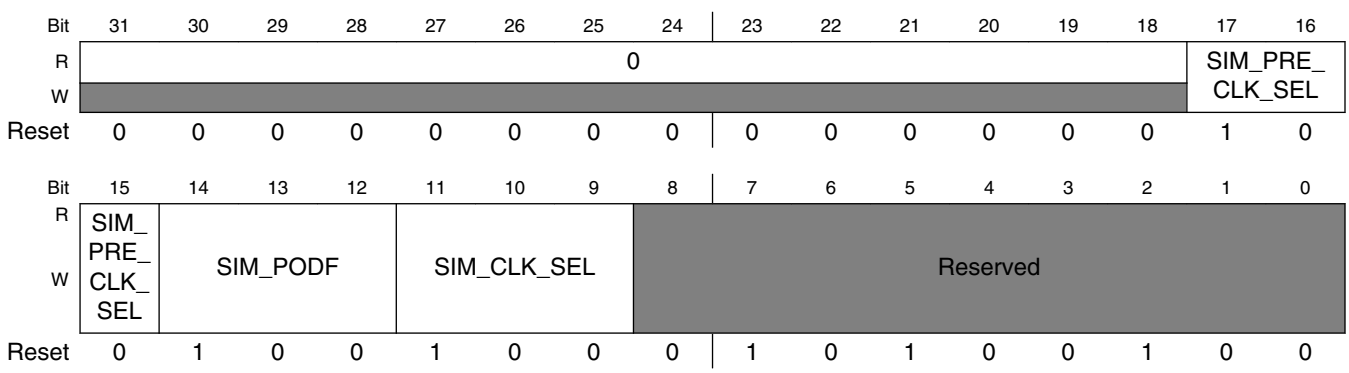
CCM_CDCDR field descriptions (continued)

Field	Description
	001 divide by 2 010 divide by 3 111 divide by 8
24–22 SPDIF0_CLK_PODF	Divider for spdif0 clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 111 divide by 8
21–20 SPDIF0_CLK_SEL	Selector for spdif0 clock multiplexer 00 derive clock from PLL4 01 derive clock from PLL3 PFD2 10 derive clock from PLL5 11 derive clock from pll3_sw_clk
-	This field is reserved. Reserved

18.6.14 CCM HSC Clock Divider Register (CCM_CHSCCDR)

The figure below represents the CCM HSC Clock Divider Register (CHSCCDR). The CHSCCDR register contains bits to control the clock generation dividers. The table below provides its field descriptions.

Address: 20C_4000h base + 34h offset = 20C_4034h



CCM_CHSCCDR field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

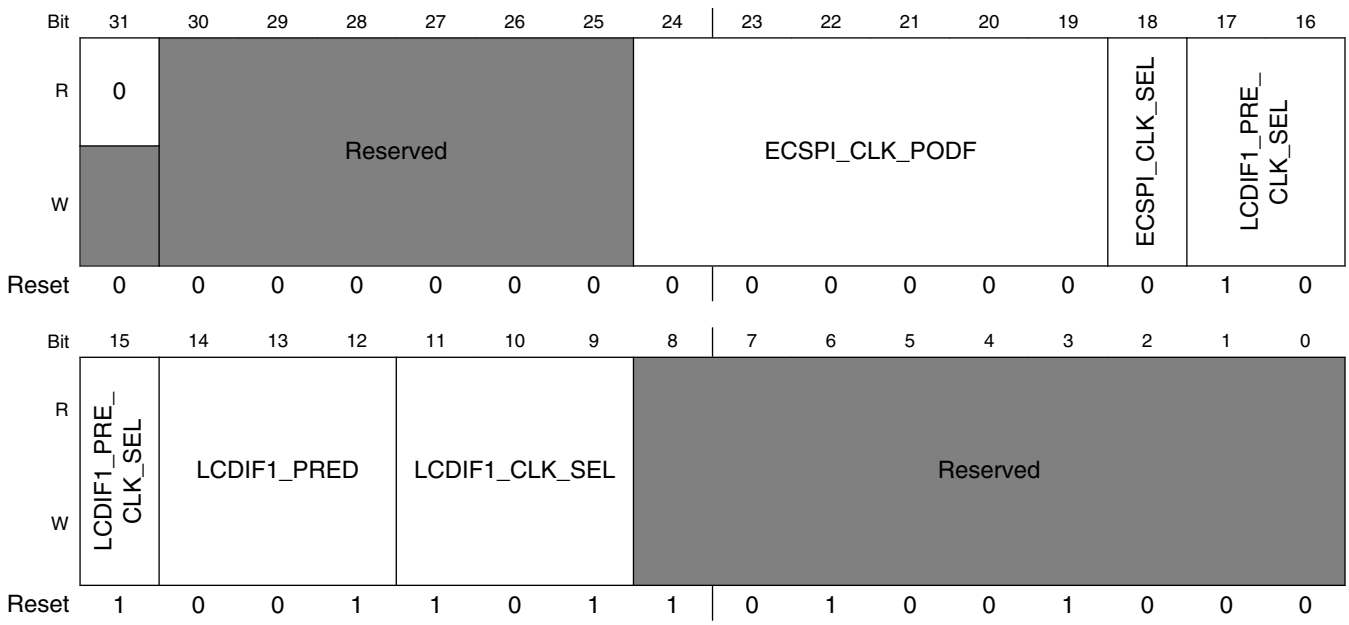
CCM_CHSCCDR field descriptions (continued)

Field	Description
17–15 SIM_PRE_CLK_SEL	Selector for SIM root clock pre-multiplexer 000 derive clock from PLL2 001 derive clock from pll3_sw_clk 010 derive clock from PLL5 011 derive clock from PLL2 PFD0 100 derive clock from PLL2 PFD2 101 derive clock from PLL3 PFD2 110-111 Reserved
14–12 SIM_PODF	Divider for SIM clock divider. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
11–9 SIM_CLK_SEL	Selector for SIM root clock multiplexer 000 derive clock from divided pre-muxed SIM clock 001 derive clock from ipp_di0_clk 010 derive clock from ipp_di1_clk 011 derive clock from ldb_di0_clk 100 derive clock from ldb_di1_clk 101-111 Reserved
-	This field is reserved. Reserved. Always set to 0.

18.6.15 CCM Serial Clock Divider Register 2 (CCM_CSCDR2)

The figure below represents the CCM Serial Clock Divider Register 2(CSCDR2). The CSCDR2 register contains bits to control the clock generation sub-module dividers. The table below provides its field descriptions.

Address: 20C_4000h base + 38h offset = 20C_4038h



CCM_CSCDR2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–25 -	This field is reserved. Reserved
24–19 ECSPI_CLK_PODF	Divider for ecspi clock podf. **NOTE:** Divider should be updated when output clock is gated. **NOTE:** The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6
18 ECSPI_CLK_SEL	Selector for the ECSPI clock multiplexor 0 derive clock from pll3_60m 1 derive clock from osc_clk

Table continues on the next page...

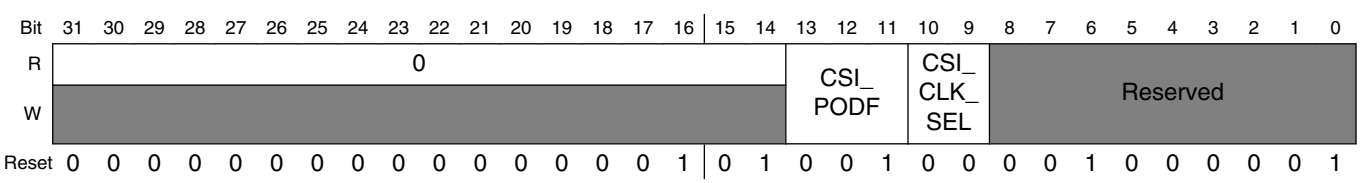
CCM_CSCDR2 field descriptions (continued)

Field	Description
17–15 LCDIF1_PRE_CLK_SEL	Selector for lcdif1 root clock pre-multiplexer 000 derive clock from PLL2 001 derive clock from PLL3 PFD3 010 derive clock from PLL5 011 derive clock from PLL2 PFD0 100 derive clock from PLL2 PFD1 101 derive clock from PLL3 PFD1 110-111 Reserved
14–12 LCDIF1_PRED	Pre-divider for lcdif1 clock. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
11–9 LCDIF1_CLK_SEL	Selector for lcdif1 root clock multiplexer 000 derive clock from divided pre-muxed lcdif1 clock 001 derive clock from ipp_di0_clk 010 derive clock from ipp_di1_clk 011 derive clock from ldb_di0_clk 100 derive clock from ldb_di1_clk 101-111 Reserved
-	This field is reserved. Reserved

18.6.16 CCM Serial Clock Divider Register 3 (CCM_CSCDR3)

The figure below represents the CCM Serial Clock Divider Register 3(CSCDR3). The CSCDR3 register contains bits to control the clock generation sub-module dividers. The table below provides its field descriptions.

Address: 20C_4000h base + 3Ch offset = 20C_403Ch



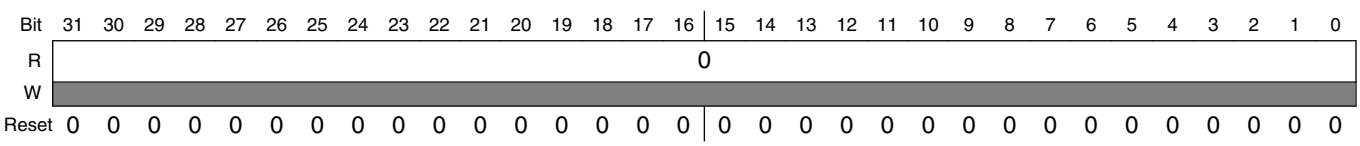
CCM_CSCDR3 field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–11 CSI_PODF	Post divider for csi_core clock. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
10–9 CSI_CLK_SEL	Selector for csi clock multiplexer 00 derive clock from osc_clk (24M) 01 derive clock from PLL2 PFD2 10 derive clock from pll3_120M 11 derive clock from PLL3 PFD1
-	This field is reserved. Reserved

18.6.17 CCM Wakeup Detector Register (CCM_CWDR)

The figure below represents the CCM Wakeup Detector Register (CWDR). The CWDR register contains reserved, read-only bits.The table below provides its field descriptions.

Address: 20C_4000h base + 44h offset = 20C_4044h



CCM_CWDR field descriptions

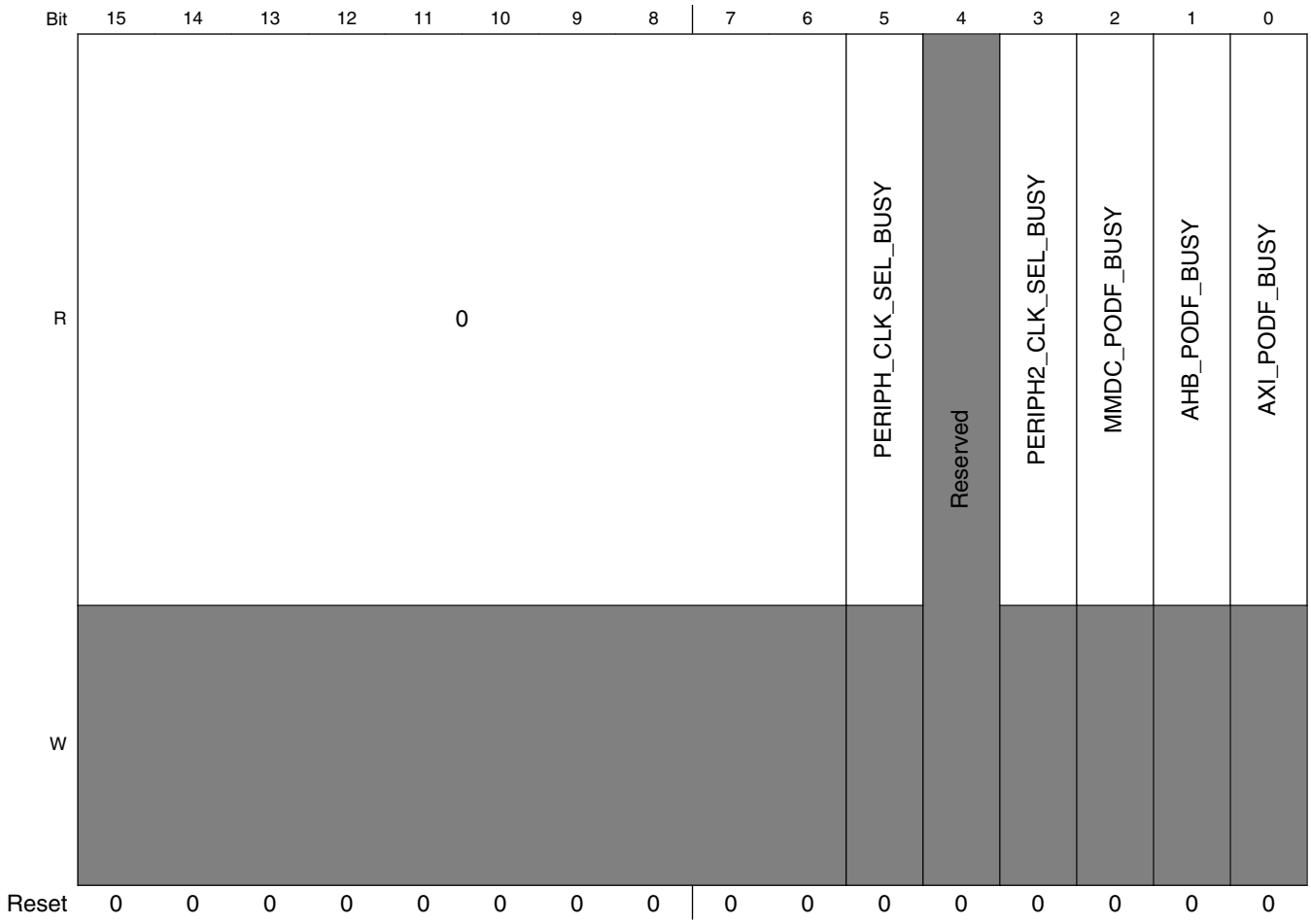
Field	Description
Reserved	This read-only field is reserved and always has the value 0.

18.6.17 CCM Divider Handshake In-Process Register (CCM_CDHIPR)

The figure below represents the CCM Divider Handshake In-Process Register (CDHIPR). The CDHIPR register contains read-only bits that indicate that CCM is in the process of updating dividers or muxes that might need handshake with modules.

Address: 20C_4000h base + 48h offset = 20C_4048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															ARM_PODF_BUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



CCM_CDHIPR field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 ARM_PODF_ BUSY	Busy indicator for arm_podf. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the previous value of the division factor, and after the handshake the written value of the arm_podf will be applied.
15–6 Reserved	This read-only field is reserved and always has the value 0.
5 PERIPH_CLK_ SEL_BUSY	Busy indicator for periph_clk_sel mux control. 0 mux is not busy and its value represents the actual division. 1 mux is busy with handshake process with module. The value read in the periph_clk_sel represents the previous value of select, and after the handshake periph_clk_sel value will be applied.
4 -	This field is reserved. Reserved

Table continues on the next page...

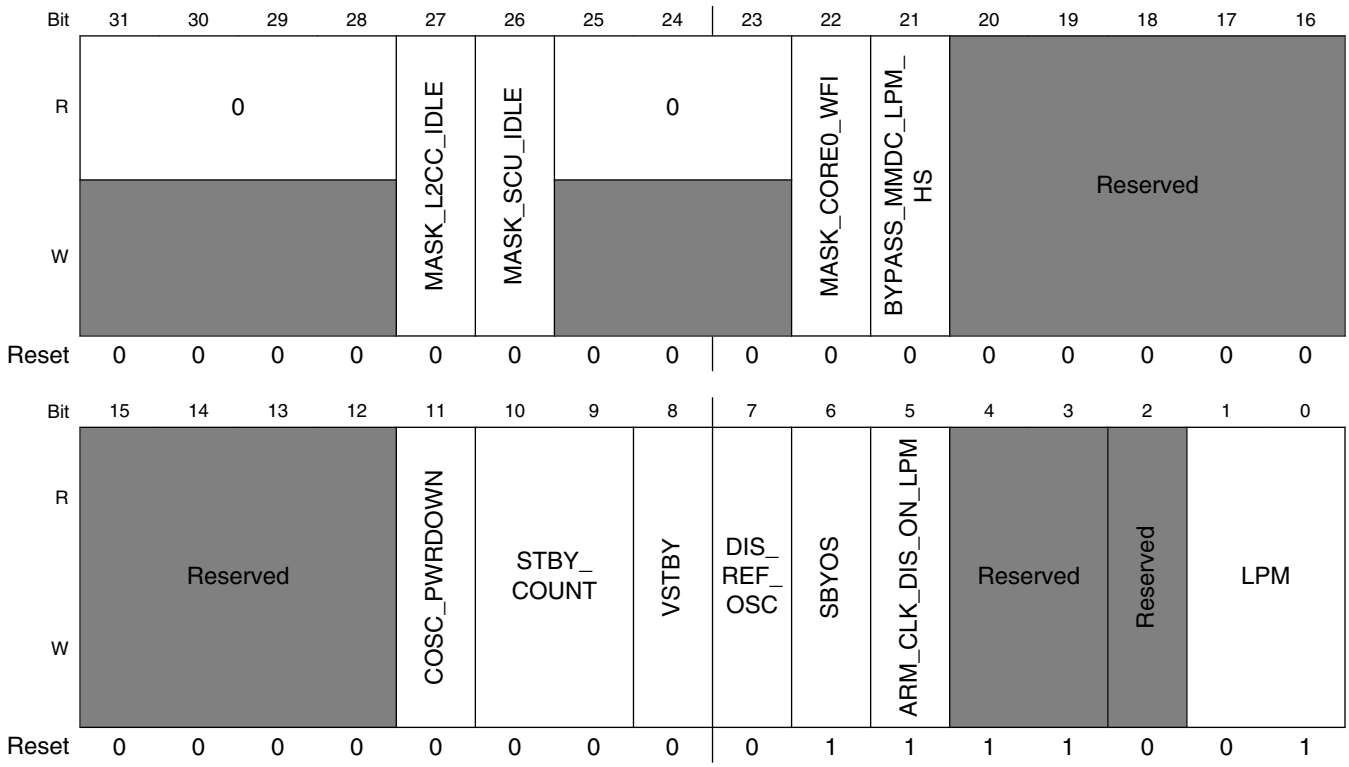
CCM_CDHIPR field descriptions (continued)

Field	Description
3 PERIPH2_CLK_SEL_BUSY	Busy indicator for periph2_clk_sel mux control. 0 mux is not busy and its value represents the actual division. 1 mux is busy with handshake process with module. The value read in the periph2_clk_sel represents the previous value of select, and after the handshake periph2_clk_sel value will be applied.
2 MMDC_PODF_BUSY	Busy indicator for mmdc_axi_podf. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the previous value of the division factor, and after the handshake the written value of the mmdc_axi_podf will be applied.
1 AHB_PODF_BUSY	Busy indicator for ahb_podf. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the previous value of the division factor, and after the handshake the written value of the ahb_podf will be applied.
0 AXI_PODF_BUSY	Busy indicator for axi_podf. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the previous value of the division factor, and after the handshake the written value of the axi_podf will be applied.

18.6.18 CCM Low Power Control Register (CCM_CLPCR)

The figure below represents the CCM Low Power Control Register (CLPCR). The CLPCR register contains bits to control the low power modes operation.The table below provides its field descriptions.

Address: 20C_4000h base + 54h offset = 20C_4054h



CCM_CLPCR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27 MASK_L2CC_IDLE	Mask L2CC IDLE for entering low power mode. **NOTE:** Assertion of all bits[27:22] will generate low power mode request 1 L2CC IDLE is masked 0 L2CC IDLE is not masked
26 MASK_SCU_IDLE	Mask SCU IDLE for entering low power mode **NOTE:** Assertion of all bits[27:22] will generate low power mode request 1 SCU IDLE is masked 0 SCU IDLE is not masked

Table continues on the next page...

CCM_CLPCR field descriptions (continued)

Field	Description
25–23 Reserved	This read-only field is reserved and always has the value 0.
22 MASK_CORE0_WFI	Mask WFI of core0 for entering low power mode **NOTE:** Assertion of all bits[27:22] will generate low power mode request 0 WFI of core0 is not masked 1 WFI of core0 is masked
21 BYPASS_MMDC_LPM_HS	Bypass handshake with mmdc on next entrance to low power mode (STOP or WAIT). CCM doesn't wait for the module's acknowledge. Handshake also will be bypassed, if CGR3 CG10 is set to gate fast mmdc clock. 0 handshake with mmdc on next entrance to low power mode will be performed. . 1 handshake with mmdc on next entrance to low power mode will be bypassed.
20–12 -	This field is reserved. Reserved
11 COSC_PWRDOWN	In run mode, software can manually control powering down of on chip oscillator, i.e. generating '1' on cosc_pwrdown signal. If software manually powered down the on chip oscillator, then sbyos functionality for on chip oscillator will be bypassed. The manual closing of onchip oscillator should be performed only in case the reference oscilator is not the source of all the clocks generation. 0 On chip oscillator will not be powered down, i.e. cosc_pwrdown = '0'. 1 On chip oscillator will be powered down, i.e. cosc_pwrdown = '1'.
10–9 STBY_COUNT	Standby counter definition. These two bits define, in the case of stop exit (if VSTBY bit was set). **NOTE:** Clock cycles ratio depends on pmic_delay_scaler, defined by CGPR[0] bit. 00 CCM will wait (1*pmic_delay_scaler)+1 ckil clock cycles 01 CCM will wait (3*pmic_delay_scaler)+1 ckil clock cycles 10 CCM will wait (7*pmic_delay_scaler)+1 ckil clock cycles 11 CCM will wait (15*pmic_delay_scaler)+1 ckil clock cycles
8 VSTBY	Voltage standby request bit. This bit defines if PMIC_STBY_REQ pin, which notifies external power management IC to move from functional voltage to standby voltage, will be asserted in STOP mode. 0 Voltage will not be changed to standby voltage after next entrance to STOP mode. (PMIC_STBY_REQ will remain negated - '0') 1 Voltage will be requested to change to standby voltage after next entrance to stop mode. (PMIC_STBY_REQ will be asserted - '1').
7 DIS_REF_OSC	dis_ref_osc - in run mode, software can manually control closing of external reference oscillator clock, i.e. generating '1' on CCM_REF_EN_B signal. If software closed manually the external reference clock, then sbyos functionality will be bypassed. The manual closing of external reference oscilator should be performed only in case the reference oscilator is not the source of any clock generation. **NOTE:** When returning from stop mode, the PMIC_STBY_REQ will be deasserted (if it was asserted when entering stop mode). See stby_count bits. 0 external high frequency oscillator will be enabled, i.e. CCM_REF_EN_B = '0'. 1 external high frequency oscillator will be disabled, i.e. CCM_REF_EN_B = '1'

Table continues on the next page...

CCM_CLPCR field descriptions (continued)

Field	Description
6 SBYOS	Standby clock oscillator bit. This bit defines if cosc_pwrdown, which power down the on chip oscillator, will be asserted in STOP mode. This bit is discarded if cosc_pwrdown='1' for the on chip oscillator. 0 On-chip oscillator will not be powered down, after next entrance to STOP mode. (CCM_REF_EN_B will remain asserted - '0' and cosc_pwrdown will remain de asserted - '0') 1 On-chip oscillator will be powered down, after next entrance to STOP mode. (CCM_REF_EN_B will be deasserted - '1' and cosc_pwrdown will be asserted - '1'). When returning from STOP mode, external oscillator will be enabled again, on-chip oscillator will return to oscillator mode, and after oscnt count, CCM will continue with the exit from the STOP mode process.
5 ARM_CLK_DIS_ON_LPM	Define if ARM clocks (arm_clk, soc_mxclk, soc_pclk, soc_dbg_pclk, vl_wrck) will be disabled on wait mode. This is useful for debug mode, when the user still wants to simulate entering wait mode and still keep ARM clock functioning. **NOTE:** Software should not enable ARM power gating in wait mode if this bit is cleared. 0 ARM clock enabled on wait mode. 1 ARM clock disabled on wait mode. .
4–3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
LPM	Setting the low power mode that system will enter on next assertion of dsm_request signal. 00 Remain in run mode 01 Transfer to wait mode 10 Transfer to stop mode 11 Reserved

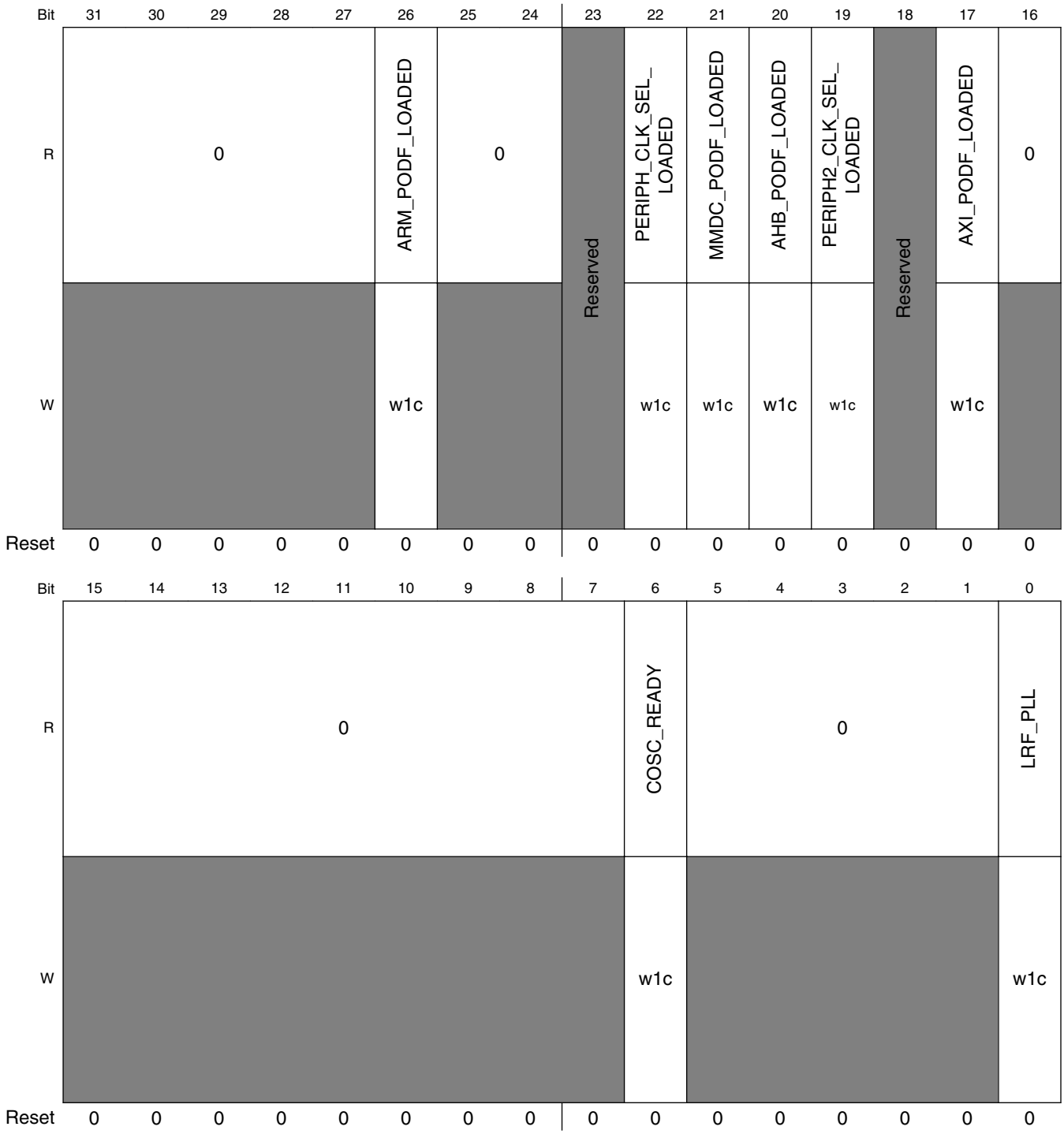
18.6.19 CCM Interrupt Status Register (CCM_CISR)

The figure below represents the CCM Interrupt Status Register (CISR). This is a write one to clear register. Once a interrupt is generated, software should write one to clear it. The table below provides its field descriptions.

NOTE

CCM interrupt request 1 can be masked by CCM interrupt request 1 mask bit. CCM interrupt request 2 can be masked by CCM interrupt request 2 mask bit.

Address: 20C_4000h base + 58h offset = 20C_4058h



CCM_CISR field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

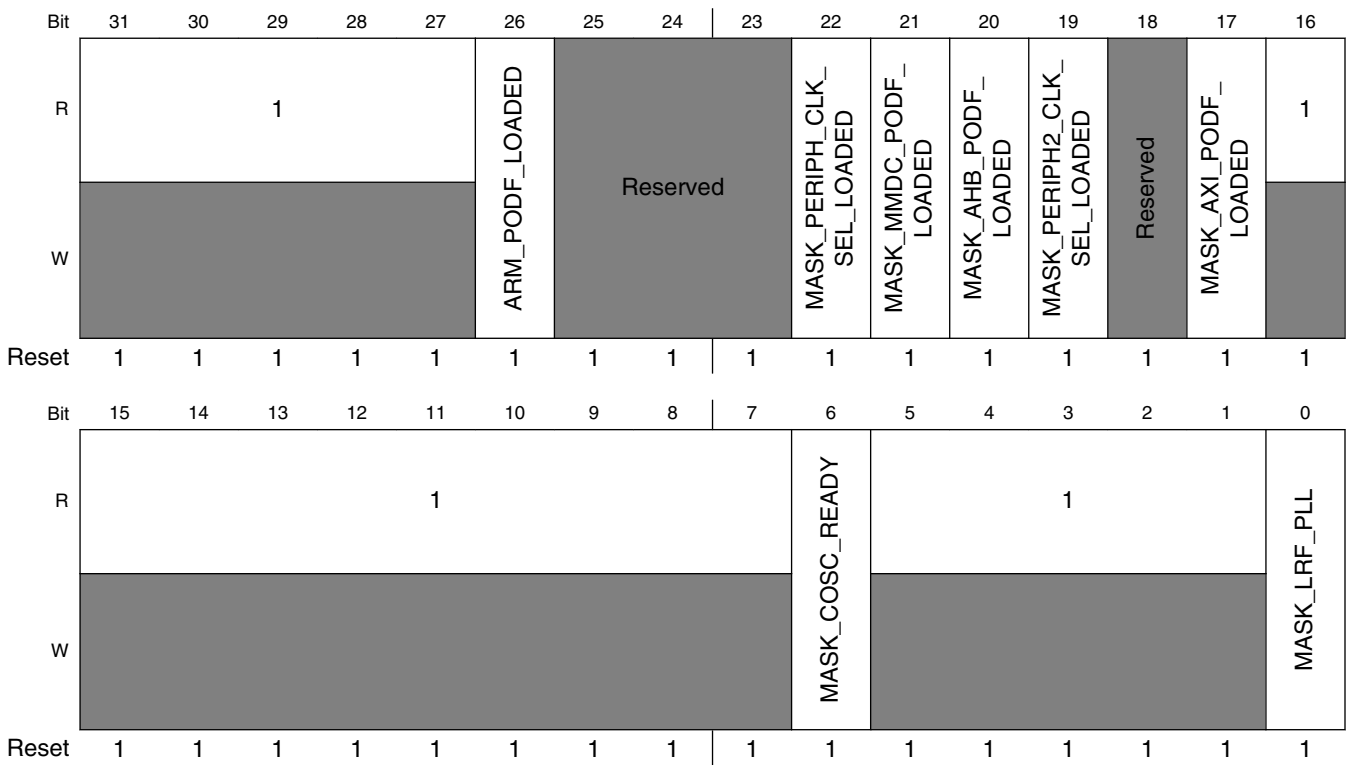
CCM_CISR field descriptions (continued)

Field	Description
26 ARM_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of arm_podf. The interrupt will commence only if arm_podf is loaded during a arm dvfs operation. 0 interrupt is not generated due to frequency change of arm_podf 1 interrupt generated due to frequency change of arm_podf
25–24 Reserved	This read-only field is reserved and always has the value 0.
23 -	This field is reserved. Reserved
22 PERIPH_CLK_ SEL_LOADED	CCM interrupt request 1 generated due to update of periph_clk_sel. 0 interrupt is not generated due to update of periph_clk_sel. 1 interrupt generated due to update of periph_clk_sel.
21 MMDC_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of mmdc_podf_ **loaded** 0 interrupt is not generated due to frequency change of mmdc_podf_ **loaded** 1 interrupt generated due to frequency change of mmdc_podf_ **loaded**
20 AHB_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of ahb_podf 0 interrupt is not generated due to frequency change of ahb_podf 1 interrupt generated due to frequency change of ahb_podf
19 PERIPH2_CLK_ SEL_LOADED	CCM interrupt request 1 generated due to frequency change of periph2_clk_sel 0 interrupt is not generated due to frequency change of periph2_clk_sel 1 interrupt generated due to frequency change of periph2_clk_sel
18 -	This field is reserved. Reserved
17 AXI_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of axi_podf 0 interrupt is not generated due to frequency change of axi_podf 1 interrupt generated due to frequency change of axi_podf
16–7 Reserved	This read-only field is reserved and always has the value 0.
6 COSC_READY	CCM interrupt request 2 generated due to on board oscillator ready, i.e. oscnt has finished counting. 0 interrupt is not generated due to on board oscillator ready 1 interrupt generated due to on board oscillator ready
5–1 Reserved	This read-only field is reserved and always has the value 0.
0 LRF_PLL	CCM interrupt request 2 generated due to lock of all enabled and not bypaseed PLLs 0 interrupt is not generated due to lock ready of all enabled and not bypaseed PLLs 1 interrupt generated due to lock ready of all enabled and not bypaseed PLLs

18.6.20 CCM Interrupt Mask Register (CCM_CIMR)

The figure below represents the CCM Interrupt Mask Register (CIMR). The table below provides its field descriptions.

Address: 20C_4000h base + 5Ch offset = 20C_405Ch



CCM_CIMR field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 1.
26 ARM_PODF_LOADED	mask interrupt generation due to frequency change of arm_podf 0 don't mask interrupt due to frequency change of arm_podf - interrupt will be created 1 mask interrupt due to frequency change of arm_podf
25–23 -	This field is reserved. Reserved
22 MASK_PERIPH_CLK_SEL_LOADED	mask interrupt generation due to update of periph_clk_sel. 0 don't mask interrupt due to update of periph_clk_sel - interrupt will be created 1 mask interrupt due to update of periph_clk_sel

Table continues on the next page...

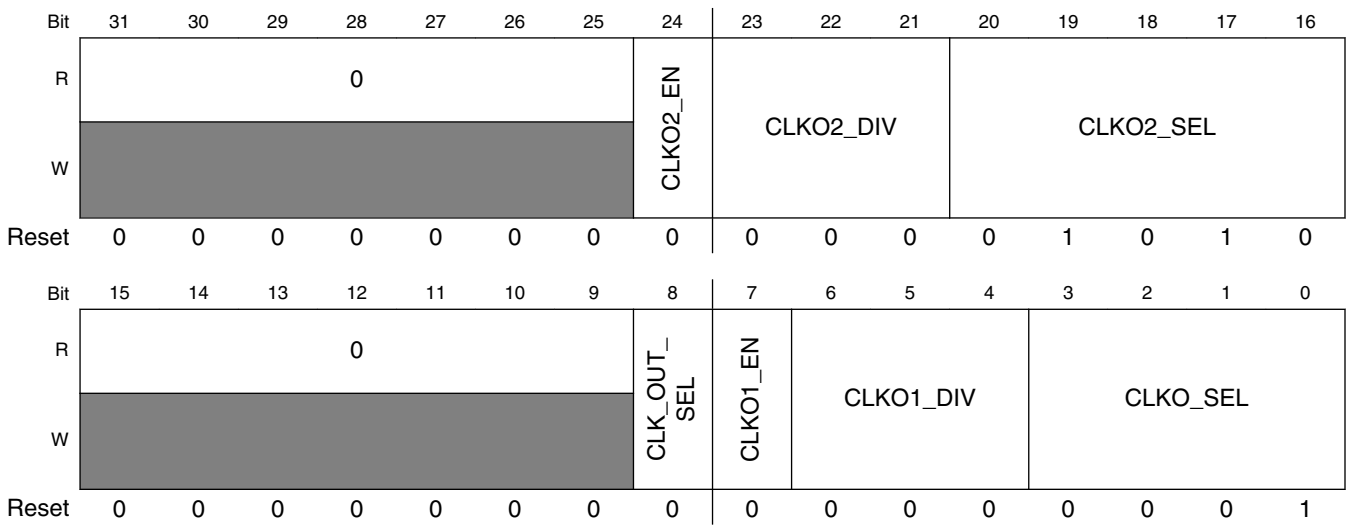
CCM_CIMR field descriptions (continued)

Field	Description
21 MASK_MMDC_ PODF_LOADED	mask interrupt generation due to update of mask_mmdc_podf 0 don't mask interrupt due to update of mask_mmdc_podf - interrupt will be created 1 mask interrupt due to update of mask_mmdc_podf
20 MASK_AHB_ PODF_LOADED	mask interrupt generation due to frequency change of ahb_podf 0 don't mask interrupt due to frequency change of ahb_podf - interrupt will be created 1 mask interrupt due to frequency change of ahb_podf
19 MASK_ PERIPH2_CLK_ SEL_LOADED	mask interrupt generation due to update of periph2_clk_sel. 0 don't mask interrupt due to update of periph2_clk_sel - interrupt will be created 1 mask interrupt due to update of periph2_clk_sel
18 -	This field is reserved. Reserved
17 MASK_AXI_ PODF_LOADED	mask interrupt generation due to frequency change of axi_podf 0 don't mask interrupt due to frequency change of axi_podf - interrupt will be created 1 mask interrupt due to frequency change of axi_podf
16–7 Reserved	This read-only field is reserved and always has the value 1.
6 MASK_COSC_ READY	mask interrupt generation due to on board oscillator ready 0 don't mask interrupt due to on board oscillator ready - interrupt will be created 1 mask interrupt due to on board oscillator ready
5–1 Reserved	This read-only field is reserved and always has the value 1.
0 MASK_LRF_PLL	mask interrupt generation due to lrf of PLLs 0 don't mask interrupt due to lrf of PLLs - interrupt will be created 1 mask interrupt due to lrf of PLLs

18.6.21 CCM Clock Output Source Register (CCM_CCOSR)

The figure below represents the CCM Clock Output Source Register (CCOSR). The CCOSR register contains bits to control the clock that will be generated on the output ipp_do_clko1 (CCM_CLKO).The table below provides its field descriptions.

Address: 20C_4000h base + 60h offset = 20C_4060h



CCM_CCOSR field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value 0.
24 CLKO2_EN	Enable of CCM_CLKO2 clock 0 CCM_CLKO2 disabled. 1 CCM_CLKO2 enabled.
23–21 CLKO2_DIV	Setting the divider of CCM_CLKO2 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
20–16 CLKO2_SEL	Selection of the clock to be generated on CCM_CLKO2 00001 mmdc_clk_root 00010 gpmi_clk_root

Table continues on the next page...

CCM_CCOSR field descriptions (continued)

Field	Description
	00011 usdhc1_clk_root 00101 wrck_clk_root 00110 ecspi_clk_root 01000 bch_clk_root 01010 Reserved 01011 csi_core 01110 osc_clk 10001 usdhc2_clk_root 10010 sai1_clk_root 10011 sai2_clk_root 10100 sai3_clk_root 10111 can_clk_root 11001 qspi1_clk_root 11011 aclk_eim_slow_clk_root 11100 uart_clk_root 11101 spdif0_clk_root 11111 Reserved
15–9 Reserved	This read-only field is reserved and always has the value 0.
8 CLK_OUT_SEL	CCM_CLKO1 output to reflect CCM_CLKO1 or CCM_CLKO2 clocks 0 CCM_CLKO1 output drives CCM_CLKO1 clock 1 CCM_CLKO1 output drives CCM_CLKO2 clock
7 CLKO1_EN	Enable of CCM_CLKO1 clock 0 CCM_CLKO1 disabled. 1 CCM_CLKO1 enabled.
6–4 CLKO1_DIV	Setting the divider of CCM_CLKO1 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
CLKO_SEL	Selection of the clock to be generated on CCM_CLKO1 0101 axi_clk_root 0110 enfc_clk_root 0111 1000 sim_clk_root 1001 1010 lcdif_pix_clk_root 1011 ahb_clk_root 1100 ipg_clk_root

Table continues on the next page...

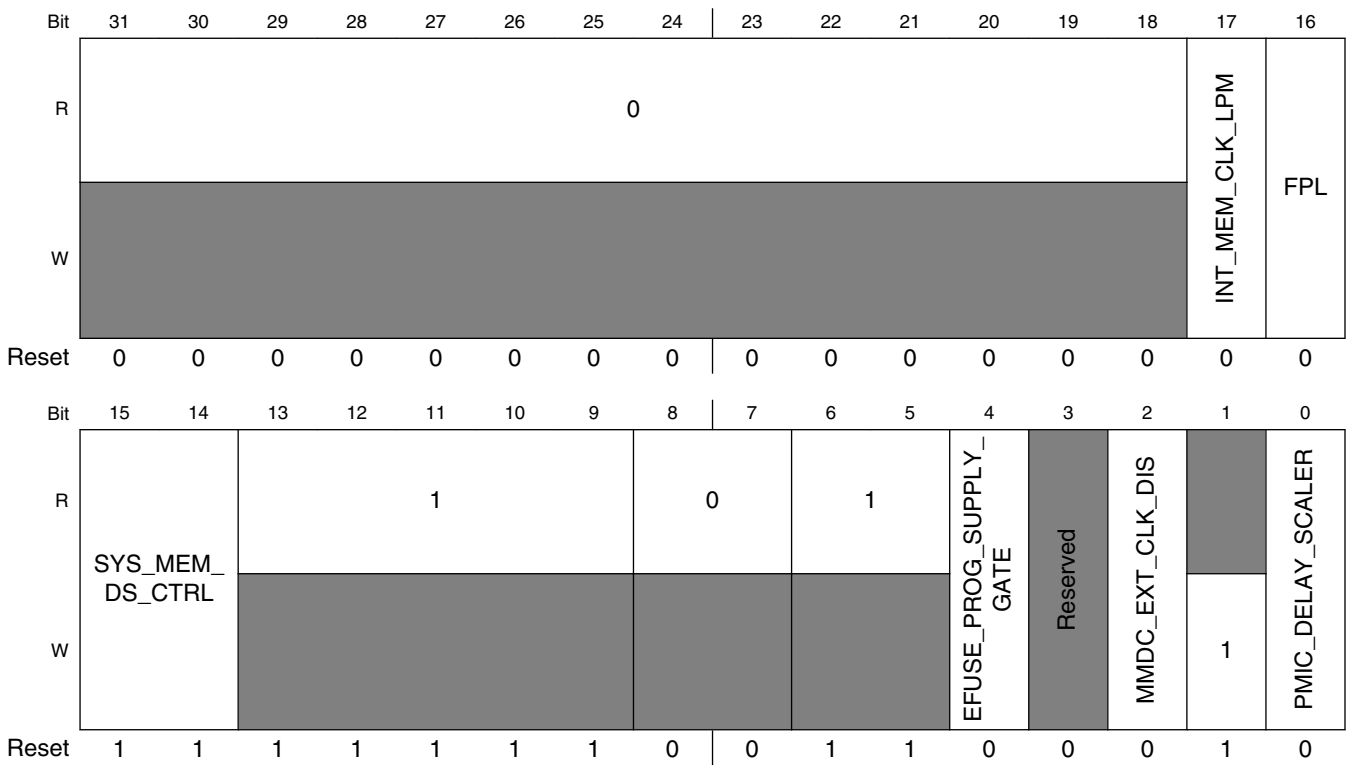
CCM_CCOSR field descriptions (continued)

Field	Description
	1101 perclk_root 1110 ckil_sync_clk_root 1111 pll4_main_clk

18.6.22 CCM General Purpose Register (CCM_CGPR)

Fast PLL enable. Can be used to engage PLL faster after STOP mode, if 24MHz OSC was active

Address: 20C_4000h base + 64h offset = 20C_4064h



CCM_CGPR field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value 0.
17 INT_MEM_CLK_LPM	Control for the Deep Sleep signal to the ARM Platform memories with additional control logic based on the ARM WFI signal. Used to keep the ARM Platform memory clocks enabled if an interrupt is pending when entering low power mode.

Table continues on the next page...

CCM_CGPR field descriptions (continued)

Field	Description
	NOTE: This bit should always bet set when the CCM_CLPCR_LPM bits are set to 01(WAIT Mode) or 10 (STOP mode) without power gating. This bit does not have to be set for STOP mode entry. 0 Disable the clock to the ARM platform memories when entering Low Power Mode 1 Keep the clocks to the ARM platform memories enabled only if an interrupt is pending when entering Low Power Modes (WAIT and STOP without power gating)
16 FPL	Fast PLL enable. 0 Engage PLL enable default way. 1 Engage PLL enable 3 CKIL clocks earlier at exiting low power mode (STOP). Should be used only if 24MHz OSC was active in low power mode.
15–14 SYS_MEM_DS_CTRL	System memory DS control 00 Disable memory DS mode always 01 Enable memory (outside ARM platform) DS mode when system STOP and PLL are disabled 1x enable memory (outside ARM platform) DS mode when system is in STOP mode
13–9 Reserved	This read-only field is reserved and always has the value 1.
8–7 Reserved	This read-only field is reserved and always has the value 0.
6–5 Reserved	This read-only field is reserved and always has the value 1.
4 EFUSE_PROG_SUPPLY_GATE	Defines the value of the output signal cgpr_dout[4]. Gate of program supply for efuse programing 0 fuse programing supply voltage is gated off to the efuse module 1 allow fuse programing.
3 -	This field is reserved. Reserved
2 MMDC_EXT_CLK_DIS	Disable external clock driver of MMDC during STOP mode 1 disable during stop mode 0 don't disable during stop mode.
1 -	Reserved. Keep default value set to '1' for proper operation.
0 PMIC_DELAY_SCALER	Defines clock dividion of clock for stby_count (pmic delay counter) 0 clock is not divided 1 clock is divided /8

18.6.23 CCM Clock Gating Register 0 (CCM_CCGR0)

CG(i) bits CCGR 0-6

These bits are used to turn on/off the clock to each module independently. The following table details the possible clock activity conditions for each module.

CGR value	Clock Activity Description
00	Clock is off during all modes. Stop enter hardware handshake is disabled.
01	Clock is on in run mode, but off in WAIT and STOP modes
10	Not applicable (Reserved).
11	Clock is on during all modes, except STOP mode.

Module should be stopped, before set its bits to "0"; clocks to the module will be stopped immediately.

The tables above show the register mapings for the different CGRs. The clock connectivity table should be used to match the "CCM output affected" to the actual clocks going into the modules.

The figure below represents the CCM Clock Gating Register 0 (CCM_CCGR0). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 7 CGR registers. The number of registers required is according to the number of peripherals in the system.

Address: 20C_4000h base + 68h offset = 20C_4068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR0 field descriptions

Field	Description
31–30 CG15	gpio2_clocks (gpio2_clk_enable)
29–28 CG14	uart2 clock (uart2_clk_enable)
27–26 CG13	gpt2 serial clocks (gpt2_serial_clk_enable)
25–24 CG12	dcic1 clocks (dcic1_clk_enable)gpt2 bus clocks (gpt2_bus_clk_enable)
23–22 CG11	CPU debug clocks (arm_dbg_clk_enable)
21–20 CG10	can2_serial clock (can2_serial_clk_enable)
19–18 CG9	can2 clock (can2_clk_enable)

Table continues on the next page...

CCM_CCGR0 field descriptions (continued)

Field	Description
17–16 CG8	can1_serial clock (can1_serial_clk_enable)
15–14 CG7	can1 clock (can1_clk_enable)
13–12 CG6	caam_wrapper_ipg clock (caam_wrapper_ipg_enable)
11–10 CG5	caam_wrapper_aclk clock (caam_wrapper_aclk_enable)
9–8 CG4	caam_secure_mem clock (caam_secure_mem_clk_enable)
7–6 CG3	asrc clock (asrc_clk_enable)
5–4 CG2	apbhdma hclk clock (apbhdma_hclk_enable)
3–2 CG1	aips_tz2 clocks (aips_tz2_clk_enable)
CG0	aips_tz1 clocks (aips_tz1_clk_enable)

18.6.24 CCM Clock Gating Register 1 (CCM_CCGR1)

The figure below represents the CCM Clock Gating Register 1(CCM_CCGR1). The clock gating registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 20C_4000h base + 6Ch offset = 20C_406Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR1 field descriptions

Field	Description
31–30 CG15	gpio5 clock (gpio5_clk_enable)
29–28 CG14	csu clock (csu_clk_enable)

Table continues on the next page...

CCM_CCGR1 field descriptions (continued)

Field	Description
27–26 CG13	gpio1 clock (gpio1_clk_enable)
25–24 CG12	uart4 clock (uart4_clk_enable)
23–22 CG11	gpt serial clock (gpt_serial_clk_enable)
21–20 CG10	gpt bus clock (gpt_clk_enable)
19–18 CG9	sim_s clock (sim_s_clk_enable)
17–16 CG8	adc1 clock (adc1_clk_enable)
15–14 CG7	epit2 clocks (epit2_clk_enable)
13–12 CG6	epit1 clocks (epit1_clk_enable)
11–10 CG5	uart3 clock (uart3_clk_enable)
9–8 CG4	adc2 clock (adc2_clk_enable)
7–6 CG3	ecspi4 clocks (ecspi4_clk_enable)
5–4 CG2	ecspi3 clocks (ecspi3_clk_enable)
3–2 CG1	ecspi2 clocks (ecspi2_clk_enable)
CG0	ecspi1 clocks (ecspi1_clk_enable)

18.6.25 CCM Clock Gating Register 2 (CCM_CCGR2)

The figure below represents the CCM Clock Gating Register 2 (CCM_CCGR2). The clock gating registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 20C_4000h base + 70h offset = 20C_4070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR2 field descriptions

Field	Description
31–30 CG15	pxp clocks (pxp_clk_enable)
29–28 CG14	lcd clocks (lcd_clk_enable)
27–26 CG13	gpio3 clock (gpio3_clk_enable)
25–24 CG12	Reserved
23–22 CG11	ipsync_ip2apb_tzasc1_ipg clocks (ipsync_ip2apb_tzasc1_ipg_master_clk_enable)
21–20 CG10	ipmux3 clock (ipmux3_clk_enable)
19–18 CG9	ipmux2 clock (ipmux2_clk_enable)
17–16 CG8	ipmux1 clock (ipmux1_clk_enable)
15–14 CG7	iomux_ipt_clk_io clock (iomux_ipt_clk_io_enable)
13–12 CG6	OCOTP_CTRL clock (iim_clk_enable)
11–10 CG5	i2c3_serial clock (i2c3_serial_clk_enable)
9–8 CG4	i2c2_serial clock (i2c2_serial_clk_enable)

Table continues on the next page...

CCM_CCGR2 field descriptions (continued)

Field	Description
7–6 CG3	i2c1_serial clock (i2c1_serial_clk_enable)
5–4 CG2	Reserved
3–2 CG1	csi clock (csi_clk_enable)
CG0	Reserved

18.6.26 CCM Clock Gating Register 3 (CCM_CCGR3)

The figure below represents the CCM Clock Gating Register 3 (CCM_CCGR3). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 20C_4000h base + 74h offset = 20C_4074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR3 field descriptions

Field	Description
31–30 CG15	Reserved
29–28 CG14	axi clock (axi_clk_enable)
27–26 CG13	mmdc_core_ipg_clk_p1 clock (mmdc_core_ipg_clk_p1_enable)
25–24 CG12	mmdc_core_ipg_clk_p0 clock (mmdc_core_ipg_clk_p0_enable)
23–22 CG11	Reserved
21–20 CG10	mmdc_core_aclk_fast_core_p0 clock (mmdc_core_aclk_fast_core_p0_enable)
19–18 CG9	a7 clkdiv patch clock (a7_clkdiv_patch_clk_enable)

Table continues on the next page...

CCM_CCGR3 field descriptions (continued)

Field	Description
17–16 CG8	wdog1 clock (wdog1_clk_enable)
15–14 CG7	qspi1 clock (qspi1_clk_enable)
13–12 CG6	gpio4 clock (gpio4_clk_enable)
11–10 CG5	lcdif1 pix clock (lcdif1_pix_clk_enable)
9–8 CG4	CA7 CCM DAP clock (ccm_dap_clk_enable)
7–6 CG3	uart6 clock (uart6_clk_enable)
5–4 CG2	enet clock (enet_clk_enable)
3–2 CG1	uart5 clock (uart5_clk_enable)
CG0	Reserved

18.6.27 CCM Clock Gating Register 4 (CCM_CCGR4)

The figure below represents the CCM Clock Gating Register 4 (CCM_CCGR4). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 20C_4000h base + 78h offset = 20C_4078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR4 field descriptions

Field	Description
31–30 CG15	rawnand_u_gpmi_input_apb clock (rawnand_u_gpmi_input_apb_clk_enable)
29–28 CG14	rawnand_u_gpmi_bch_input_gpmi_io clock (rawnand_u_gpmi_bch_input_gpmi_io_clk_enable)

Table continues on the next page...

CCM_CCGR4 field descriptions (continued)

Field	Description
27–26 CG13	rawnand_u_gpmi_bch_input_bch clock (rawnand_u_gpmi_bch_input_bch_clk_enable)
25–24 CG12	rawnand_u_bch_input_apb clock (rawnand_u_bch_input_apb_clk_enable)
23–22 CG11	pwm4 clocks (pwm4_clk_enable)
21–20 CG10	pwm3 clocks (pwm3_clk_enable)
19–18 CG9	pwm2 clocks (pwm2_clk_enable)
17–16 CG8	pwm1 clocks (pwm1_clk_enable)
15–14 CG7	pl301_mx6qper2_mainclk_enable (pl301_mx6qper2_mainclk_enable)
13–12 CG6	pl301_mx6qper1_bch clocks (pl301_mx6qper1_bchclk_enable) **NOTE:** This gates bch_clk_root to sim_m fabric.
11–10 CG5	tsc_dig clock (tsc_clk_enable)
9–8 CG4	cxapbsyncbridge slave clock (cxapbsyncbridge_slave_clk_enable)
7–6 CG3	sim_cpu clock (sim_cpu_clk_enable)
5–4 CG2	iomuxc gpr clock (iomuxc_gpr_clk_enable)
3–2 CG1	iomuxc clock (iomuxc_clk_enable)
CG0	Reserved

18.6.28 CCM Clock Gating Register 5 (CCM_CCGR5)

The figure below represents the CCM Clock Gating Register 5 (CCM_CCGR5). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 20C_4000h base + 7Ch offset = 20C_407Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR5 field descriptions

Field	Description
31–30 CG15	sai2 clock (sai2_clk_enable)
29–28 CG14	sai1 clock (sai1_clk_enable)
27–26 CG13	uart7 clock (uart7_clk_enable)
25–24 CG12	uart1 clock (uart1_clk_enable)
23–22 CG11	sai3 clock (sai3_clk_enable)
21–20 CG10	snvs_lp clock (snvs_lp_clk_enable)
19–18 CG9	snvs_hp clock (snvs_hp_clk_enable)
17–16 CG8	sim_main clock (sim_main_clk_enable)
15–14 CG7	spdif / audio clock (spdif_clk_enable, audio_clk_root)
13–12 CG6	spba clock (spba_clk_enable)
11–10 CG5	wdog2 clock (wdog2_clk_enable)
9–8 CG4	kpp clock (kpp_clk_enable)

Table continues on the next page...

CCM_CCGR5 field descriptions (continued)

Field	Description
7–6 CG3	sdma clock (sdma_clk_enable)
5–4 CG2	snvs dryice clock (snvs_dryice_clk_enable)
3–2 CG1	stcr clock (stcr_clk_enable)
CG0	rom clock (rom_clk_enable)

18.6.29 CCM Clock Gating Register 6 (CCM_CCGR6)

The figure below represents the CCM Clock Gating Register 6 (CCM_CCGR6). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 20C_4000h base + 80h offset = 20C_4080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR6 field descriptions

Field	Description
31–30 CG15	pwm7 clocks (pwm7_clk_enable)
29–28 CG14	pwm6 clocks (pwm6_clk_enable)
27–26 CG13	pwm5 clocks (pwm5_clk_enable)
25–24 CG12	i2c4 serial clock (i2c4_serial_clk_enable)
23–22 CG11	anadig clocks (anadig_clk_enable)
21–20 CG10	wdog3 clock (wdog3_clk_enable)
19–18 CG9	Reserved

Table continues on the next page...

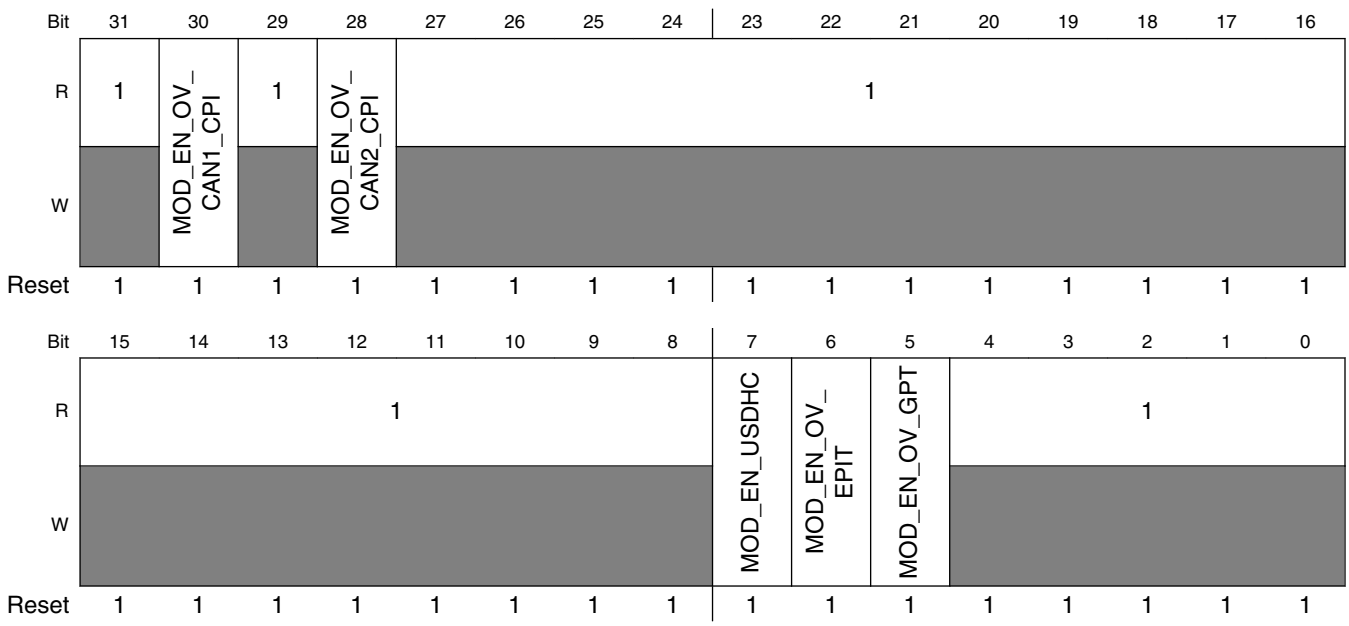
CCM_CCGR6 field descriptions (continued)

Field	Description
17–16 CG8	pwm8 clocks (pwm8_clk_enable)
15–14 CG7	uart8 clocks (uart8_clk_enable)
13–12 CG6	Reserved
11–10 CG5	eim_slow clocks (eim_slow_clk_enable)
9–8 CG4	sim2_clock (sim2_clk_enable)
7–6 CG3	sim1_clock (sim1_clk_enable)
5–4 CG2	usdhc2 clocks (usdhc2_clk_enable)
3–2 CG1	usdhc1 clocks (usdhc1_clk_enable)
CG0	usboh3 clock (usboh3_clk_enable)

18.6.30 CCM Module Enable Overide Register (CCM_CMEOR)

The follow figure represents the CCM Module Enable Override Register (CMEOR). The CMEOR register contains bits to override the clock enable signal from the module. This bit will be applicable only for modules whose clock enable signals are used. The following table provides its field descriptions.

Address: 20C_4000h base + 88h offset = 20C_4088h



CCM_CMEOR field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 1.
30 MOD_EN_OV_CAN1_CPI	Overide clock enable signal from CAN1 - clock will not be gated based on CAN's signal 'enable_clk_cpi'. 0 don't overide module enable signal 1 overide module enable signal
29 Reserved	This read-only field is reserved and always has the value 1.
28 MOD_EN_OV_CAN2_CPI	Overide clock enable signal from CAN2 - clock will not be gated based on CAN's signal 'enable_clk_cpi'. 0 don't override module enable signal 1 override module enable signal
27–8 Reserved	This read-only field is reserved and always has the value 1.

Table continues on the next page...

CCM_CMEOR field descriptions (continued)

Field	Description
7 MOD_EN_ USDHC	overide clock enable signal from USDHC. 0 don't override module enable signal 1 override module enable signal
6 MOD_EN_OV_ EPIT	Overide clock enable signal from EPIT - clock will not be gated based on EPIT's signal 'ipg_enable_clk' . 0 don't override module enable signal 1 override module enable signal
5 MOD_EN_OV_ GPT	Overide clock enable signal from GPT - clock will not be gated based on GPT's signal 'ipg_enable_clk' . 0 don't override module enable signal 1 override module enable signal
Reserved	This read-only field is reserved and always has the value 1.

18.7 CCM Analog Memory Map/Register Definition

This section describes the registers for the analog PLLs. The registers which have the same description are grouped within {}. The register offsets for the various PLLs are:

- ARM PLL: {0h000, 0h004, 0h008, 0h00C}.
- USB1 PLL: {0h010, 0h014, 0h018, 0h01C}, {0h0F0, 0h0F4, 0h0F8, 0h0FC}.
- System PLL: {0h030, 0h034, 0h038, 0h03C}, 0h040, 0h050, 0h060, {0h100, 0h104, 0h108, 0h10C}.
- Audio / Video PLL: {0h070, 0h074, 0h078, 0h07C}, 0h080, 0h090, {0h0A0, 0h0A4, 0h0A8, 0h0AC}, 0h0B0, 0h0C0

CCM_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8000	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM)	32	R/W	0001_3063h	18.7.1/694
20C_8004	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM_SET)	32	R/W	0001_3063h	18.7.1/694
20C_8008	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM_CLR)	32	R/W	0001_3063h	18.7.1/694
20C_800C	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM_TOG)	32	R/W	0001_3063h	18.7.1/694
20C_8010	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1)	32	R/W	0001_2000h	18.7.2/696

Table continues on the next page...

CCM_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8014	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_SET)	32	R/W	0001_2000h	18.7.2/696
20C_8018	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_CLR)	32	R/W	0001_2000h	18.7.2/696
20C_801C	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_TOG)	32	R/W	0001_2000h	18.7.2/696
20C_8020	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2)	32	R/W	0001_2000h	18.7.3/698
20C_8024	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2_SET)	32	R/W	0001_2000h	18.7.3/698
20C_8028	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2_CLR)	32	R/W	0001_2000h	18.7.3/698
20C_802C	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2_TOG)	32	R/W	0001_2000h	18.7.3/698
20C_8030	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS)	32	R/W	0001_3001h	18.7.4/700
20C_8034	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_SET)	32	R/W	0001_3001h	18.7.4/700
20C_8038	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_CLR)	32	R/W	0001_3001h	18.7.4/700
20C_803C	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_TOG)	32	R/W	0001_3001h	18.7.4/700
20C_8040	528MHz System PLL Spread Spectrum Register (CCM_ANALOG_PLL_SYS_SS)	32	R/W	0000_0000h	18.7.5/702
20C_8050	Numerator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_NUM)	32	R/W	0000_0000h	18.7.6/702
20C_8060	Denominator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_DENOM)	32	R/W	0000_0012h	18.7.7/703
20C_8070	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO)	32	R/W	0001_1006h	18.7.8/704
20C_8074	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_SET)	32	R/W	0001_1006h	18.7.8/704
20C_8078	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_CLR)	32	R/W	0001_1006h	18.7.8/704
20C_807C	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_TOG)	32	R/W	0001_1006h	18.7.8/704
20C_8080	Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)	32	R/W	05F5_E100h	18.7.9/706
20C_8090	Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)	32	R/W	2964_619Ch	18.7.10/ 707
20C_80A0	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO)	32	R/W	0001_100Ch	18.7.11/ 708
20C_80A4	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO_SET)	32	R/W	0001_100Ch	18.7.11/ 708

Table continues on the next page...

CCM_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_80A8	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO_CLR)	32	R/W	0001_100Ch	18.7.11/ 708
20C_80AC	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO_TOG)	32	R/W	0001_100Ch	18.7.11/ 708
20C_80B0	Numerator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_NUM)	32	R/W	05F5_E100h	18.7.12/ 710
20C_80C0	Denominator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_DENOM)	32	R/W	10A2_4447h	18.7.13/ 711
20C_80E0	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET)	32	R/W	0001_1001h	18.7.14/ 712
20C_80E4	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_SET)	32	R/W	0001_1001h	18.7.14/ 712
20C_80E8	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_CLR)	32	R/W	0001_1001h	18.7.14/ 712
20C_80EC	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_TOG)	32	R/W	0001_1001h	18.7.14/ 712
20C_80F0	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480)	32	R/W	1311_100Ch	18.7.15/ 714
20C_80F4	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_SET)	32	R/W	1311_100Ch	18.7.15/ 714
20C_80F8	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_CLR)	32	R/W	1311_100Ch	18.7.15/ 714
20C_80FC	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_TOG)	32	R/W	1311_100Ch	18.7.15/ 714
20C_8100	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528)	32	R/W	1018_101Bh	18.7.16/ 716
20C_8104	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_SET)	32	R/W	1018_101Bh	18.7.16/ 716
20C_8108	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_CLR)	32	R/W	1018_101Bh	18.7.16/ 716
20C_810C	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_TOG)	32	R/W	1018_101Bh	18.7.16/ 716
20C_8150	Miscellaneous Register 0 (CCM_ANALOG_MISC0)	32	R/W	0400_0000h	18.7.17/ 719
20C_8154	Miscellaneous Register 0 (CCM_ANALOG_MISC0_SET)	32	R/W	0400_0000h	18.7.17/ 719
20C_8158	Miscellaneous Register 0 (CCM_ANALOG_MISC0_CLR)	32	R/W	0400_0000h	18.7.17/ 719
20C_815C	Miscellaneous Register 0 (CCM_ANALOG_MISC0_TOG)	32	R/W	0400_0000h	18.7.17/ 719
20C_8160	Miscellaneous Register 1 (CCM_ANALOG_MISC1)	32	R/W	0000_0000h	18.7.18/ 723
20C_8164	Miscellaneous Register 1 (CCM_ANALOG_MISC1_SET)	32	R/W	0000_0000h	18.7.18/ 723

Table continues on the next page...

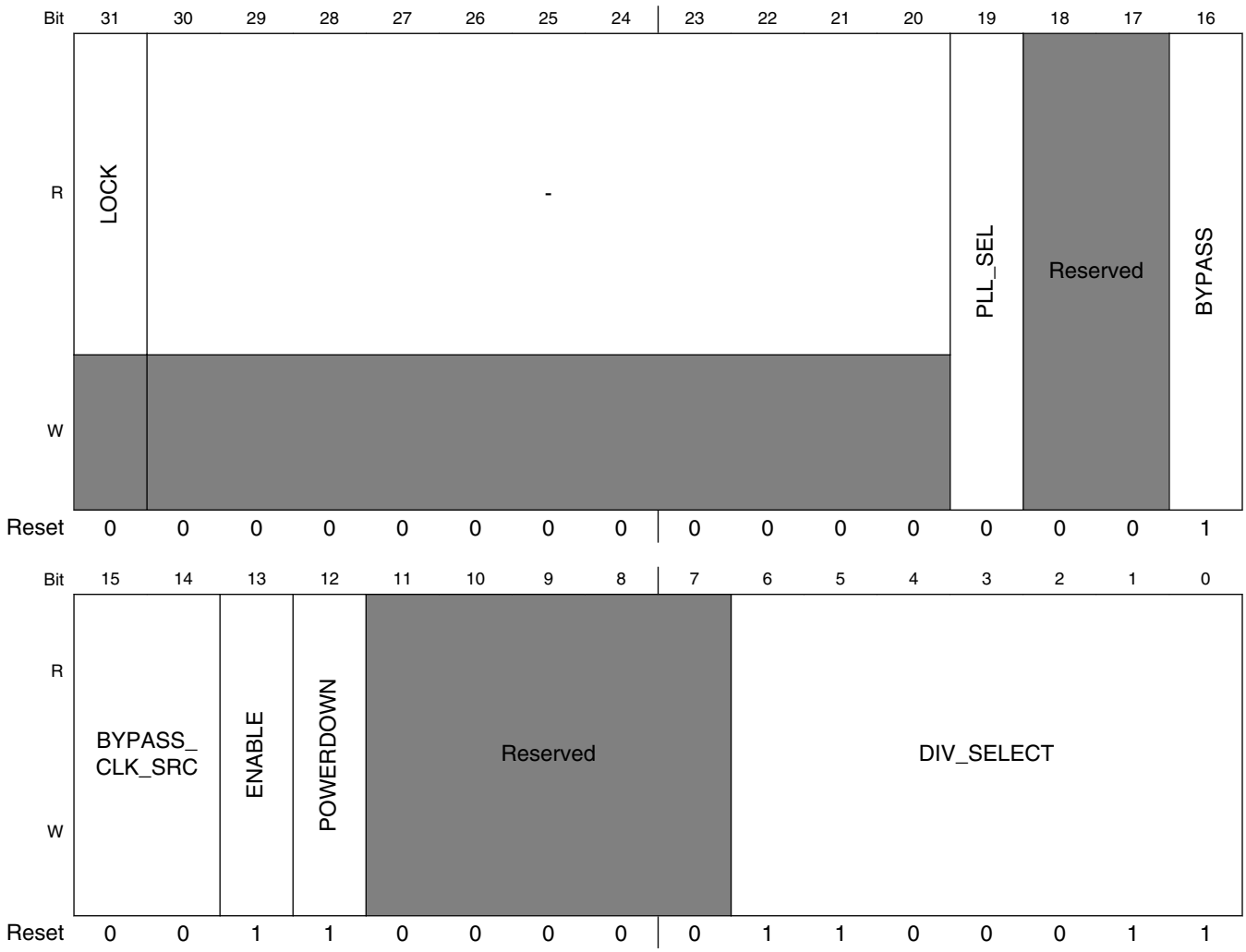
CCM_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8168	Miscellaneous Register 1 (CCM_ANALOG_MISC1_CLR)	32	R/W	0000_0000h	18.7.18/ 723
20C_816C	Miscellaneous Register 1 (CCM_ANALOG_MISC1_TOG)	32	R/W	0000_0000h	18.7.18/ 723
20C_8170	Miscellaneous Register 2 (CCM_ANALOG_MISC2)	32	R/W	0027_2727h	18.7.19/ 726
20C_8174	Miscellaneous Register 2 (CCM_ANALOG_MISC2_SET)	32	R/W	0027_2727h	18.7.19/ 726
20C_8178	Miscellaneous Register 2 (CCM_ANALOG_MISC2_CLR)	32	R/W	0027_2727h	18.7.19/ 726
20C_817C	Miscellaneous Register 2 (CCM_ANALOG_MISC2_TOG)	32	R/W	0027_2727h	18.7.19/ 726

18.7.1 Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM*n*)

The control register provides control for the system PLL.

Address: 20C_8000h base + 0h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PLL_ARM*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–20 -	Always set to zero (0).
19 PLL_SEL	Reserved

Table continues on the next page...

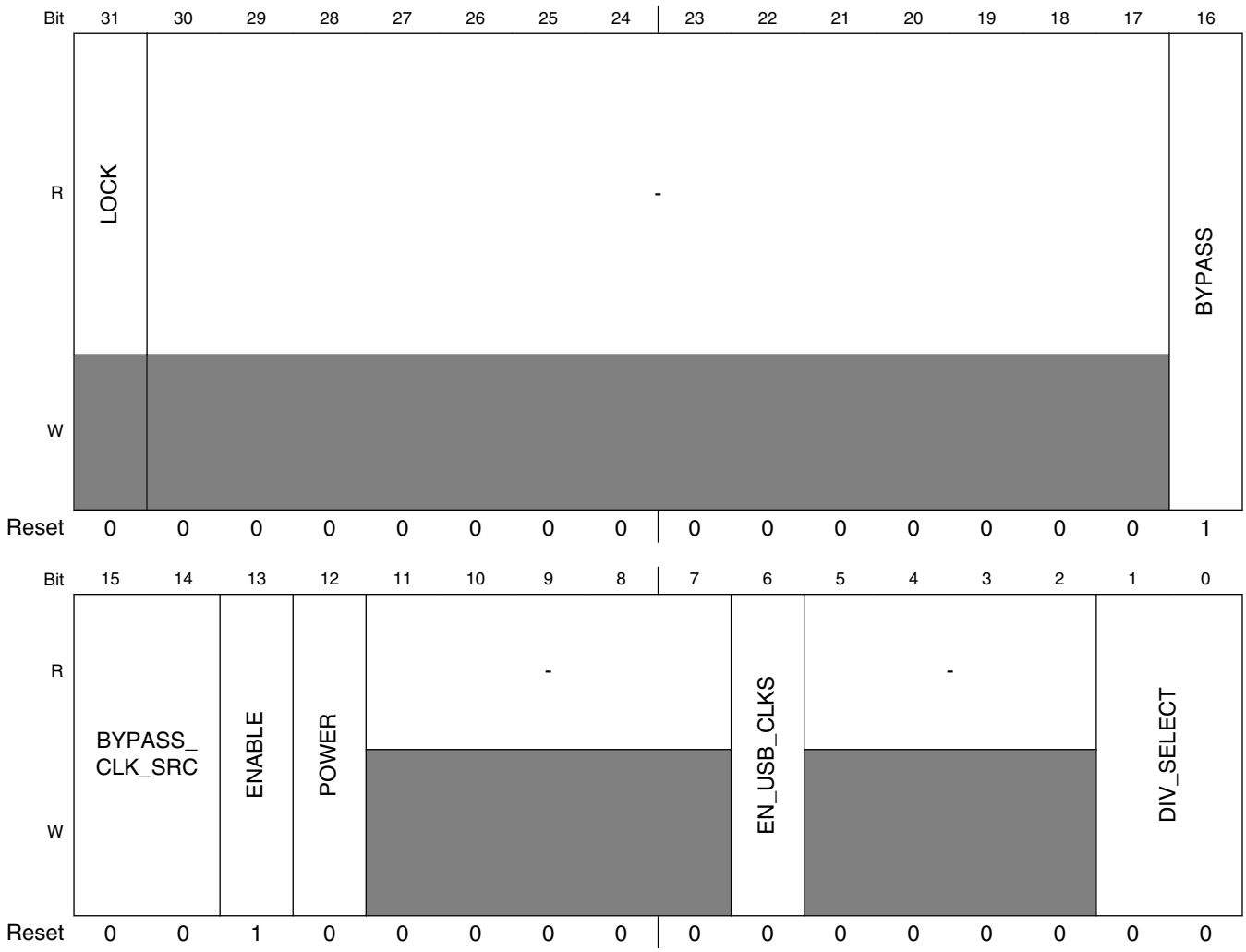
CCM_ANALOG_PLL_ARM*n* field descriptions (continued)

Field	Description
18–17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_SRC	Determines the bypass source. **NOTE:** Changing the Bypass clock source also changes the PLL reference clock source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved** — 0x3 **Reserved** —
13 ENABLE	Enable the clock output.
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.
DIV_SELECT	This field controls the PLL loop divider. Valid range for divider value: 54-108. Fout = Fin * div_select/2.0.

18.7.2 Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1*n*)

The control register provides control for USBPHY0 480MHz PLL.

Address: 20C_8000h base + 10h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PLL_USB1*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–17 -	Always set to zero (0).
16 BYPASS	Bypass the PLL.

Table continues on the next page...

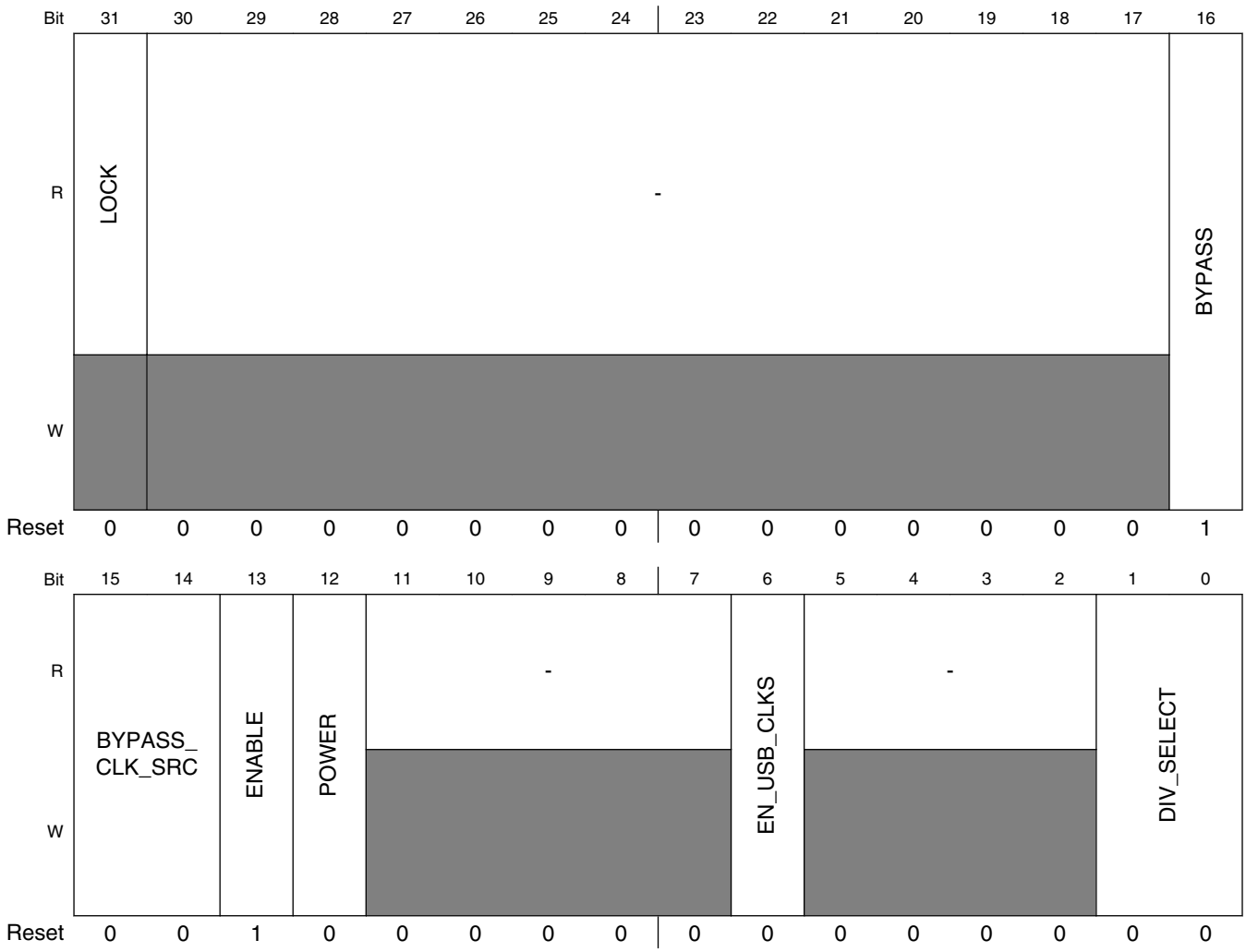
CCM_ANALOG_PLL_USB1*n* field descriptions (continued)

Field	Description
15–14 BYPASS_CLK_SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **GPANAIO** — 0x3 **CHRG_DET_B** —
13 ENABLE	Enable the PLL clock output.
12 POWER	Powers up the PLL. This bit will be set automatically when USBPHY0 remote wakeup event happens.
11–7 -	Always set to zero (0).
6 EN_USB_CLKS	Powers the 9-phase PLL outputs for USBPHYn. Additionally, the UTMI clock gate must be deasserted in the USBPHYn to enable USBn operation (clear CLKGATE bit in USBPHYn_CTRL). This bit will be set automatically when USBPHYn remote wakeup event occurs. 0 PLL outputs for USBPHYn off. 1 PLL outputs for USBPHYn on.
5–2 -	Always set to zero (0).
DIV_SELECT	This field controls the PLL loop divider. 0 - Fout=Fref*20; 1 - Fout=Fref*22.

18.7.3 Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2*n*)

The control register provides control for USBPHY1 480MHz PLL.

Address: 20C_8000h base + 20h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PLL_USB2*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–17 -	Always set to zero (0).
16 BYPASS	Bypass the PLL.

Table continues on the next page...

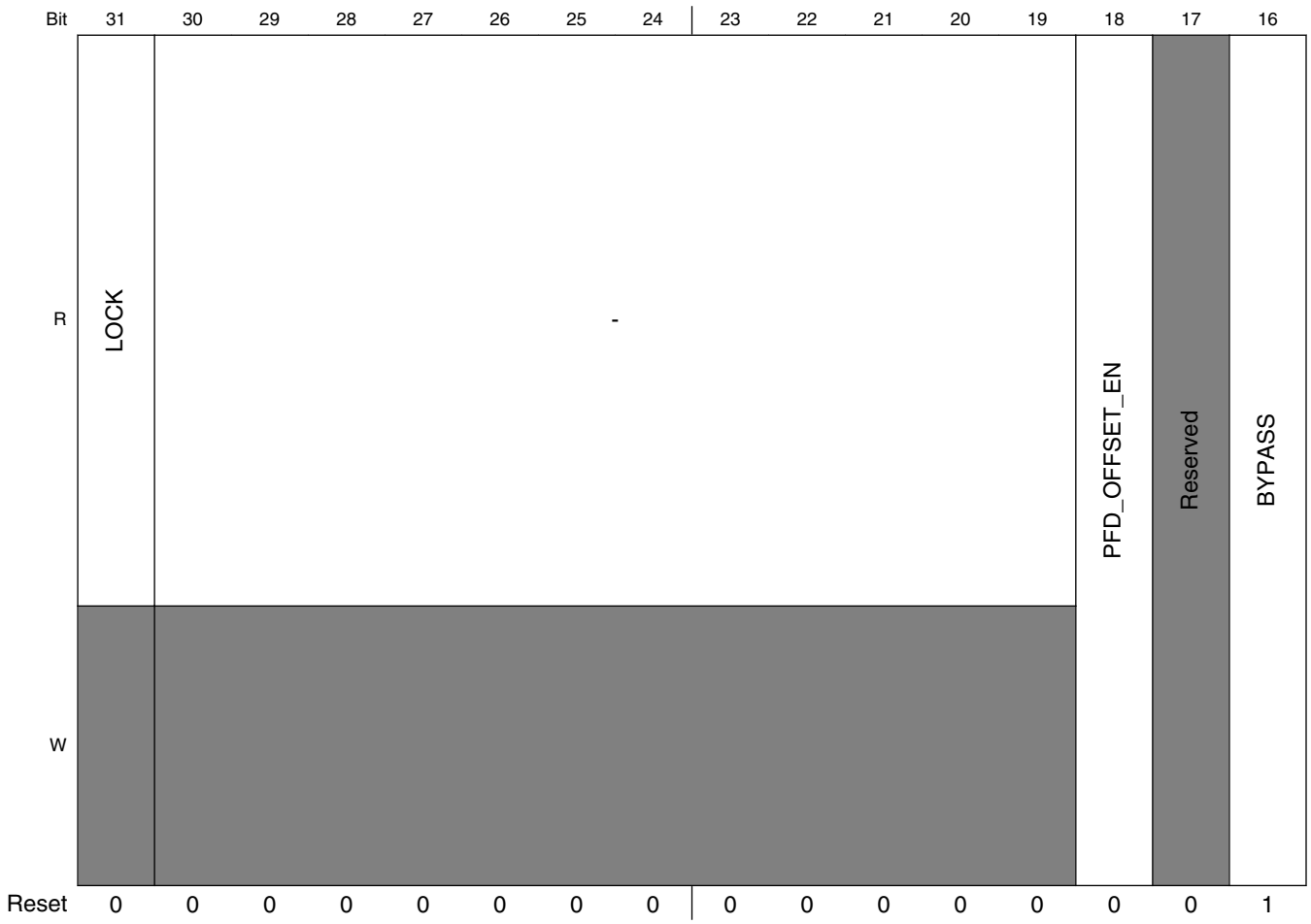
CCM_ANALOG_PLL_USB2*n* field descriptions (continued)

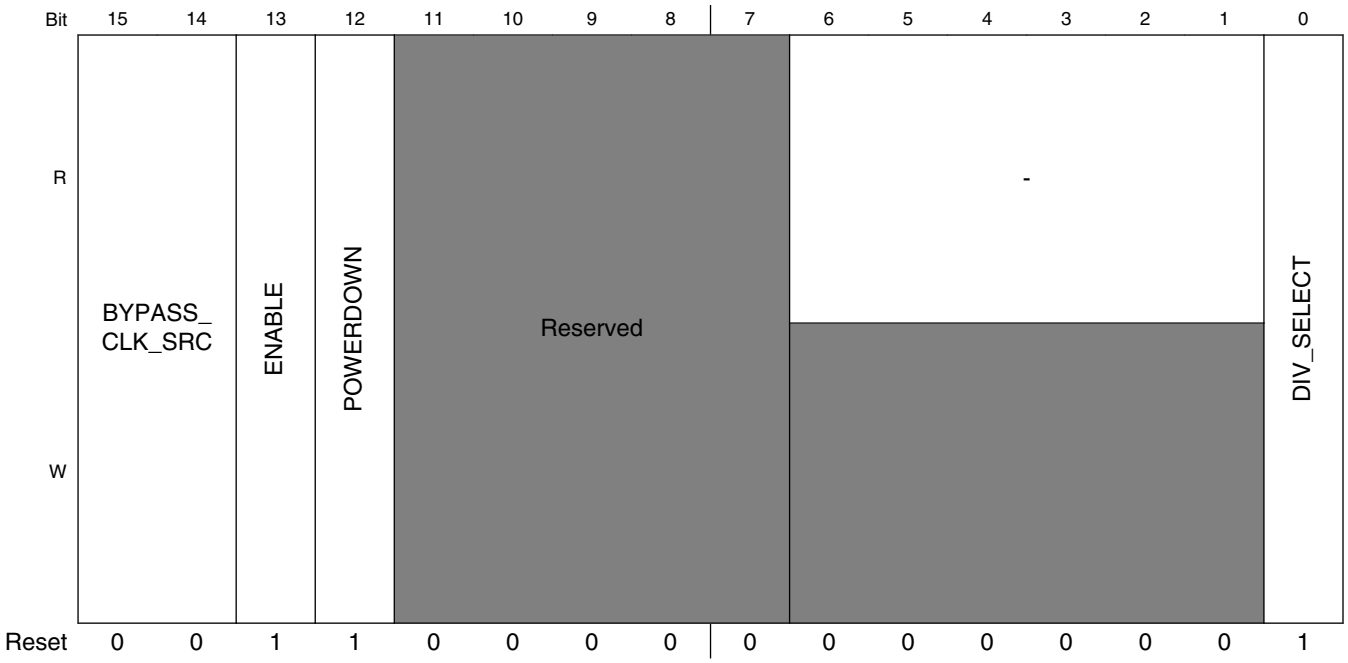
Field	Description
15–14 BYPASS_CLK_SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved** — 0x3 **Reserved** —
13 ENABLE	Enable the PLL clock output.
12 POWER	Powers up the PLL. This bit will be set automatically when USBPHY1 remote wakeup event happens.
11–7 -	Always set to zero (0).
6 EN_USB_CLKS	0: 8-phase PLL outputs for USBPHY1 are powered down. If set to 1, 8-phase PLL outputs for USBPHY1 are powered up. Additionally, the utmi clock gate must be deasserted in the USBPHY1 to enable USB0 operation (clear CLKGATE bit in USBPHY1_CTRL).This bit will be set automatically when USBPHY1 remote wakeup event happens.
5–2 -	Always set to zero (0).
DIV_SELECT	This field controls the PLL loop divider. 0 - Fout=Fref*20; 1 - Fout=Fref*22.

18.7.4 Analog System PLL Control Register (CCM_ANALOG_PLL_SYS*n*)

The control register provides control for the 528MHz PLL.

Address: 20C_8000h base + 30h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PLL_SYS*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–19 -	Always set to zero (0).
18 PFD_OFFSET_ EN	Enables an offset in the phase frequency detector.
17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **GPANAIO** — 0x3 **CHRG_DET_B** —
13 ENABLE	Enable PLL output
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.
6–1 -	Always set to zero (0).
0 DIV_SELECT	This field controls the PLL loop divider. 0 - Fout=Fref*20; 1 - Fout=Fref*22.

18.7.5 528MHz System PLL Spread Spectrum Register (CCM_ANALOG_PLL_SYS_SS)

This register contains the 528 PLL spread spectrum controls.

Address: 20C_8000h base + 40h offset = 20C_8040h



CCM_ANALOG_PLL_SYS_SS field descriptions

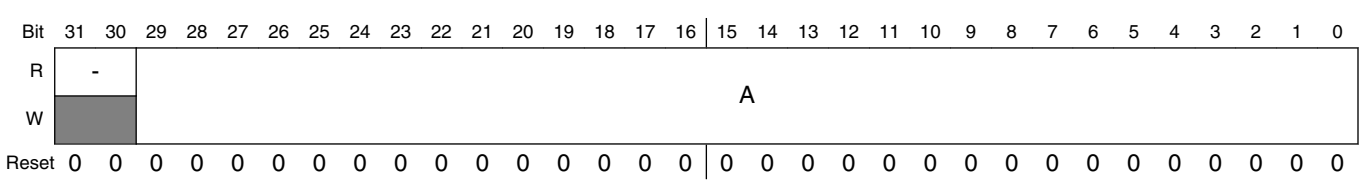
Field	Description
31–16 STOP	Frequency change = stop/CCM_ANALOG_PLL_SYS_DENOM[B]*24MHz.
15 ENABLE	0 — Spread spectrum modulation disabled 1 — Soread spectrum modulation enabled
STEP	Frequency change step = step/CCM_ANALOG_PLL_SYS_DENOM[B]*24MHz.

18.7.6 Numerator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_NUM)

This register contains the numerator of 528MHz PLL fractional loop divider (signed number).

Absoulte value should be less than denominator

Address: 20C_8000h base + 50h offset = 20C_8050h



CCM_ANALOG_PLL_SYS_NUM field descriptions

Field	Description
31–30 -	Always set to zero (0).
A	30 bit numerator (A) of fractional loop divider (signed integer).

18.7.7 Denominator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_DENOM)

This register contains the Denominator of 528MHz PLL fractional loop divider.

Address: 20C_8000h base + 60h offset = 20C_8060h

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

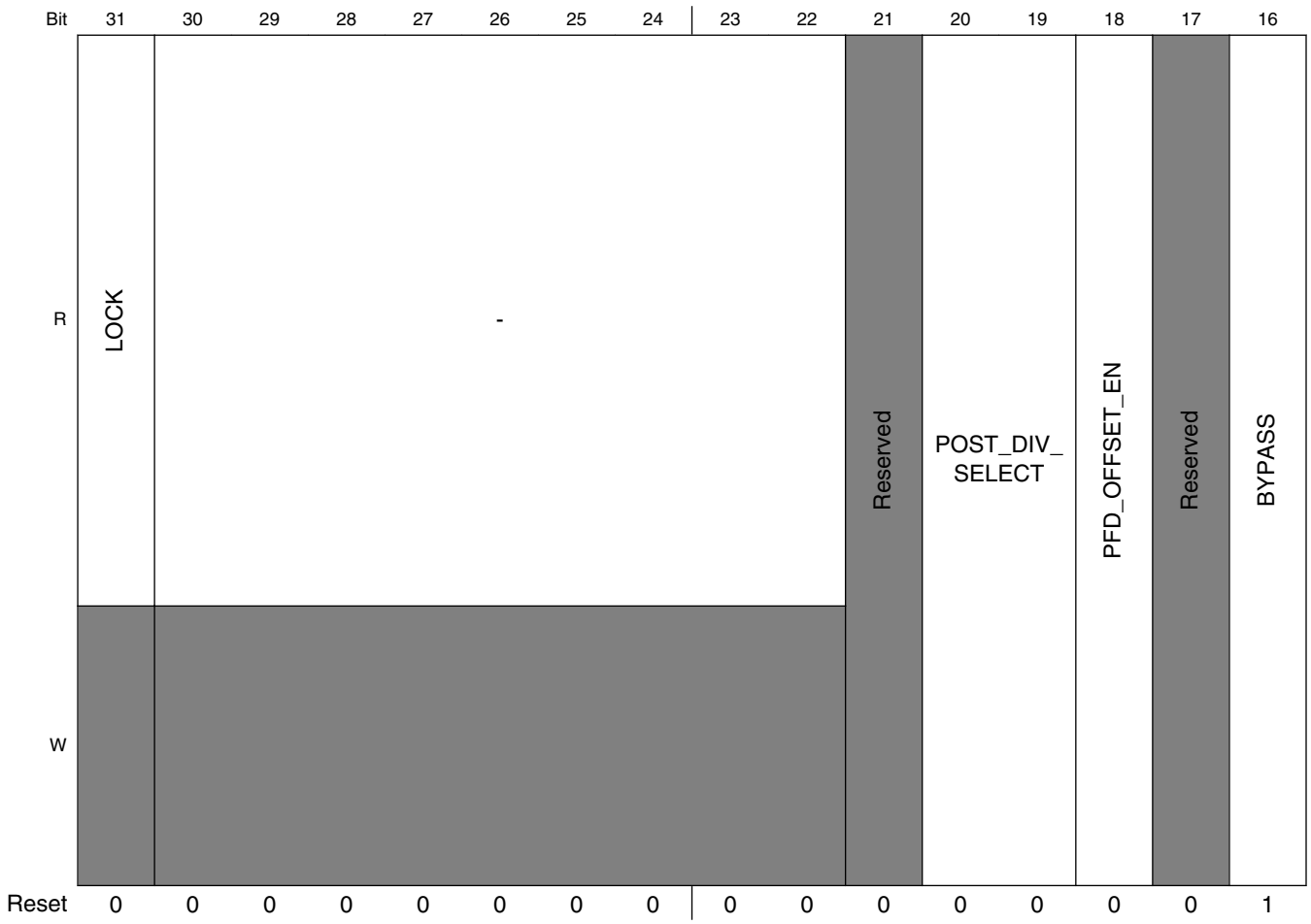
CCM_ANALOG_PLL_SYS_DENOM field descriptions

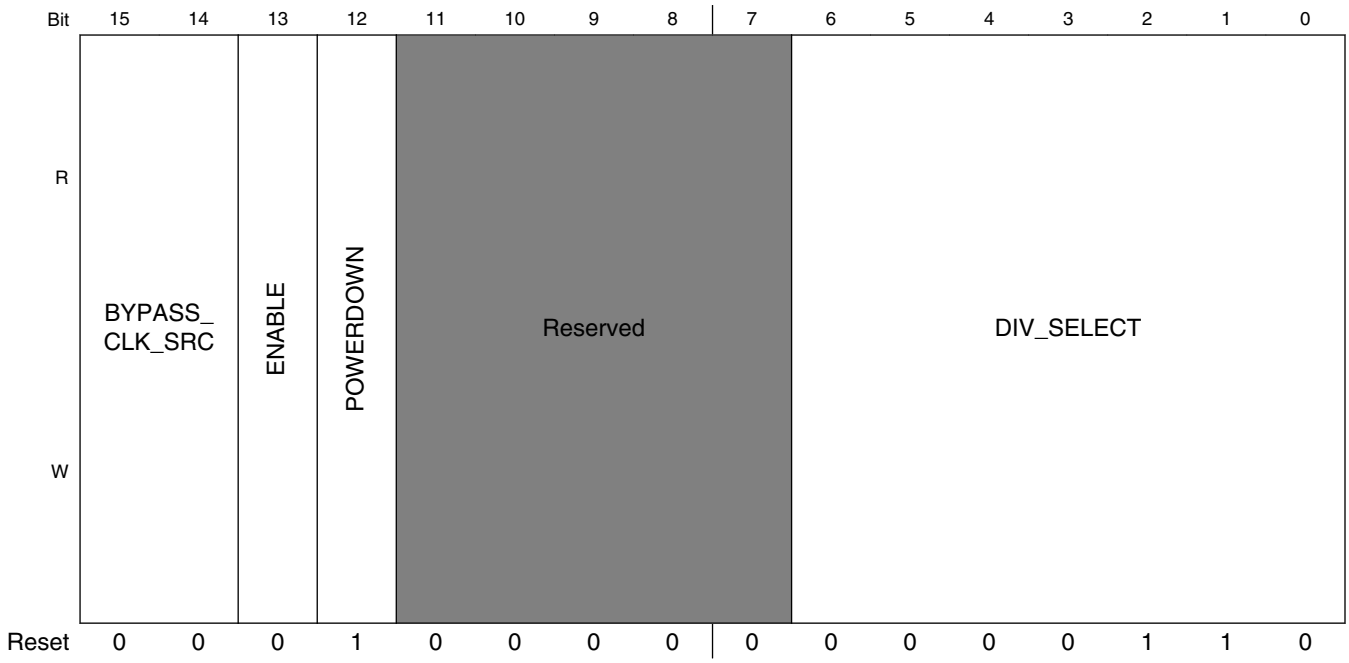
Field	Description
31–30 -	Always set to zero (0).
B	30 bit Denominator (B) of fractional loop divider (unsigned integer).

18.7.8 Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO*n*)

The control register provides control for the audio PLL.

Address: 20C_8000h base + 70h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PLL_AUDIO*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–22 -	Always set to zero (0).
21 -	This field is reserved. Reserved
20–19 POST_DIV_ SELECT	These bits implement a divider after the PLL, but before the enable and bypass mux. 00 — Divide by 4. 01 — Divide by 2. 10 — Divide by 1. 11 — Reserved
18 PFD_OFFSET_ EN	Enables an offset in the phase frequency detector.
17 -	This field is reserved. Revsered
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved** — 0x3 **Reserved** —

Table continues on the next page...

CCM_ANALOG_PLL_AUDIO*n* field descriptions (continued)

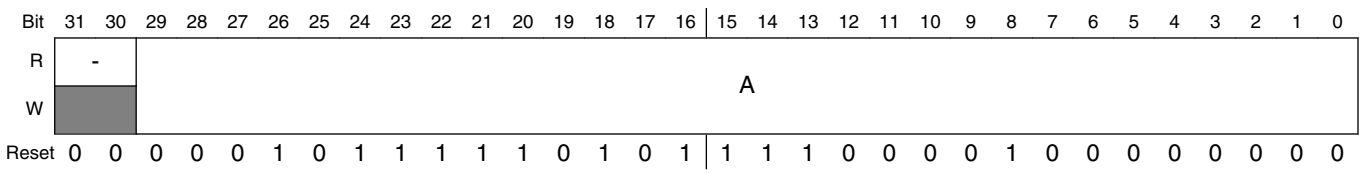
Field	Description
13 ENABLE	Enable PLL output
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.
DIV_SELECT	This field controls the PLL loop divider. Valid range for DIV_SELECT divider value: 27~54.

18.7.9 Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)

This register contains the numerator (A) of Audio PLL fractional loop divider.(Signed number), absolute value should be less than denominator

Absolute value should be less than denominator

Address: 20C_8000h base + 80h offset = 20C_8080h



CCM_ANALOG_PLL_AUDIO_NUM field descriptions

Field	Description
31–30 -	Always set to zero (0).
A	30 bit numerator of fractional loop divider.

18.7.10 Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)

This register contains the Denominator (B) of Audio PLL fractional loop divider. (unsigned number)

Address: 20C_8000h base + 90h offset = 20C_8090h

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

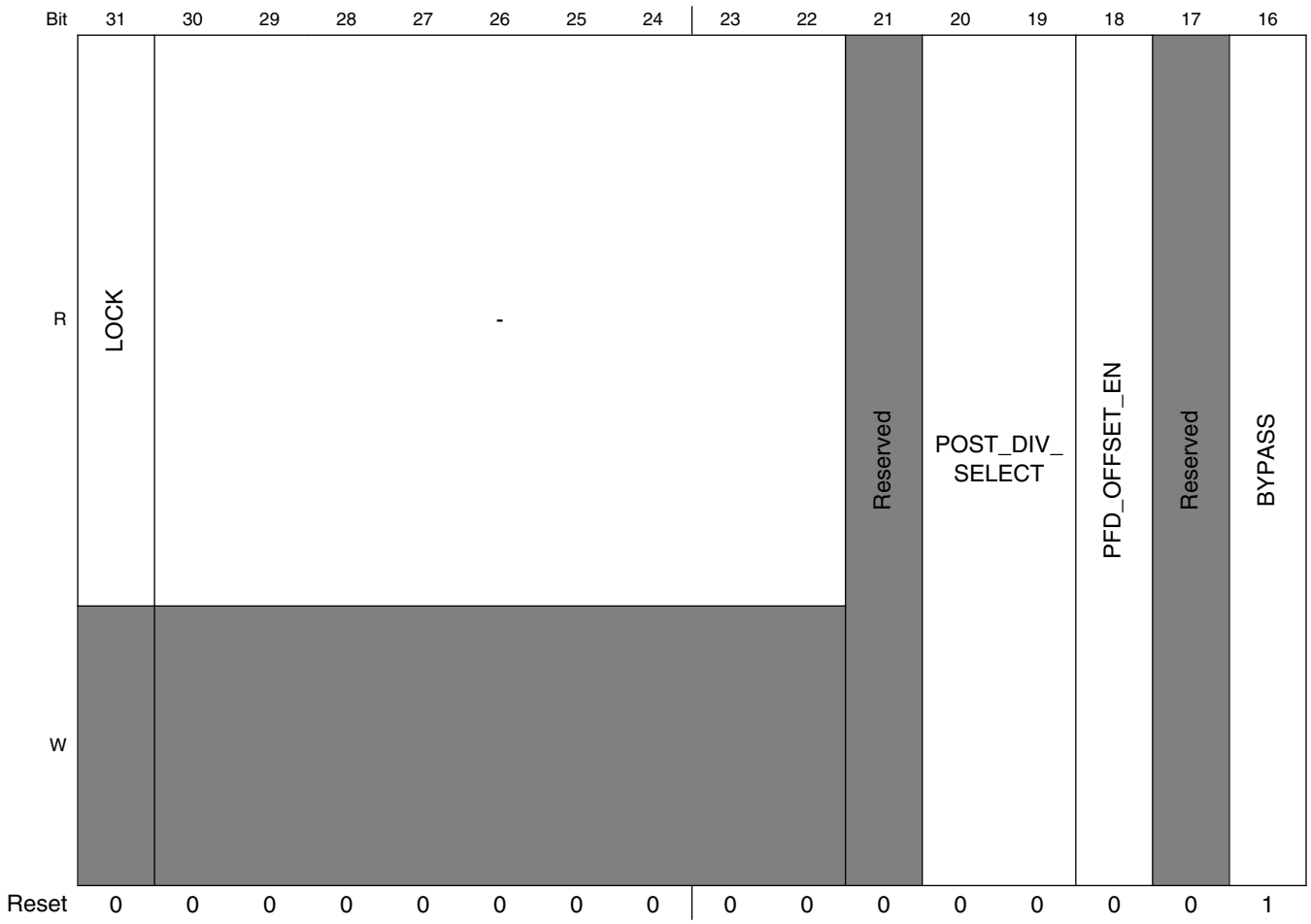
CCM_ANALOG_PLL_AUDIO_DENOM field descriptions

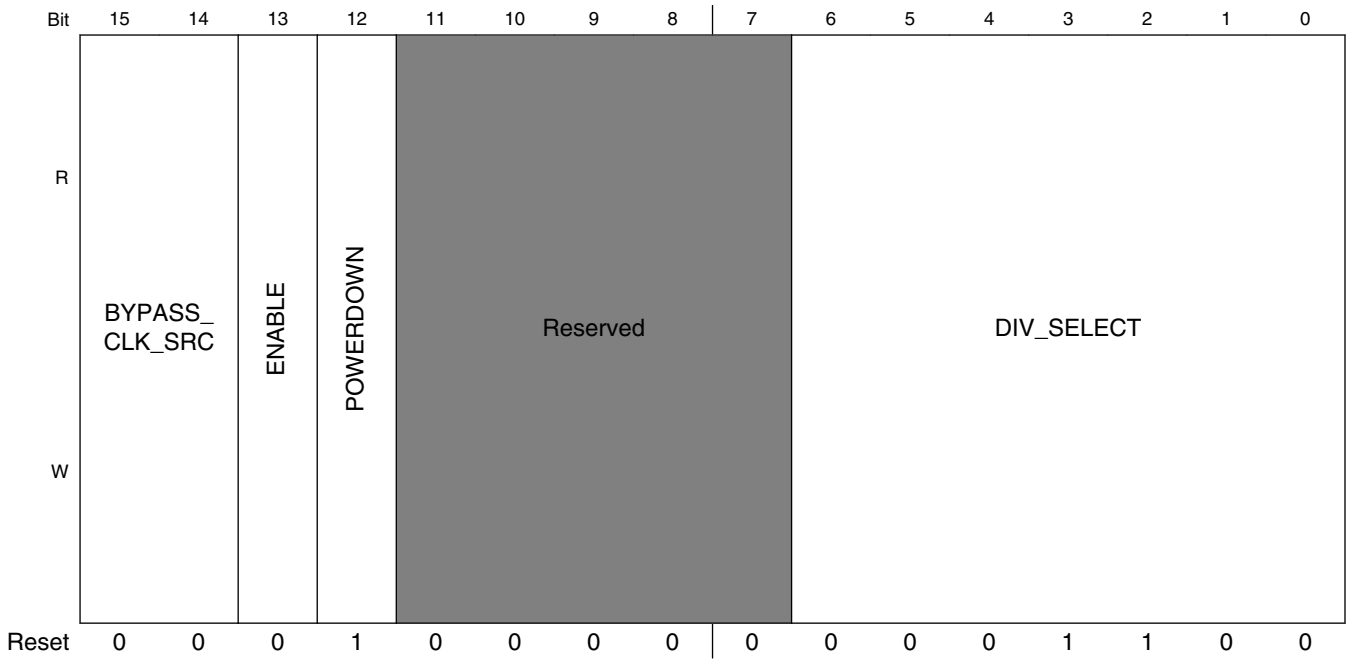
Field	Description
31–30 -	Always set to zero (0).
B	30 bit Denominator of fractional loop divider.

18.7.11 Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO*n*)

The control register provides control for the Video PLL.

Address: 20C_8000h base + A0h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PLL_VIDEO*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–22 -	Always set to zero (0).
21 -	This field is reserved. Revserved
20–19 POST_DIV_ SELECT	These bits implement a divider after the PLL, but before the enable and bypass mux. 00 — Divide by 4. 01 — Divide by 2. 10 — Divide by 1. 11 — Reserved
18 PFD_OFFSET_ EN	Enables an offset in the phase frequency detector.
17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved** — 0x3 **Reserved** —

Table continues on the next page...

CCM_ANALOG_PLL_VIDEO*n* field descriptions (continued)

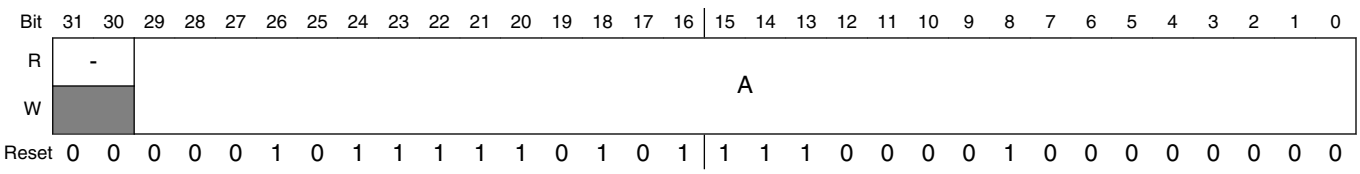
Field	Description
13 ENABLE	Enalbe PLL output
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.
DIV_SELECT	This field controls the PLL loop divider. Valid range for DIV_SELECT divider value: 27~54.

18.7.12 Numerator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_NUM)

This register contains the numerator (A) of Video PLL fractional loop divider.(Signed number)

Absolute value should be less than denominator

Address: 20C_8000h base + B0h offset = 20C_80B0h



CCM_ANALOG_PLL_VIDEO_NUM field descriptions

Field	Description
31–30 -	Always set to zero (0).
A	30 bit numerator of fractional loop divider(Signed number), absolute value should be less than denominator

18.7.13 Denominator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_DENOM)

This register contains the Denominator (B) of Video PLL fractional loop divider. (Unsigned number)

Address: 20C_8000h base + C0h offset = 20C_80C0h

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

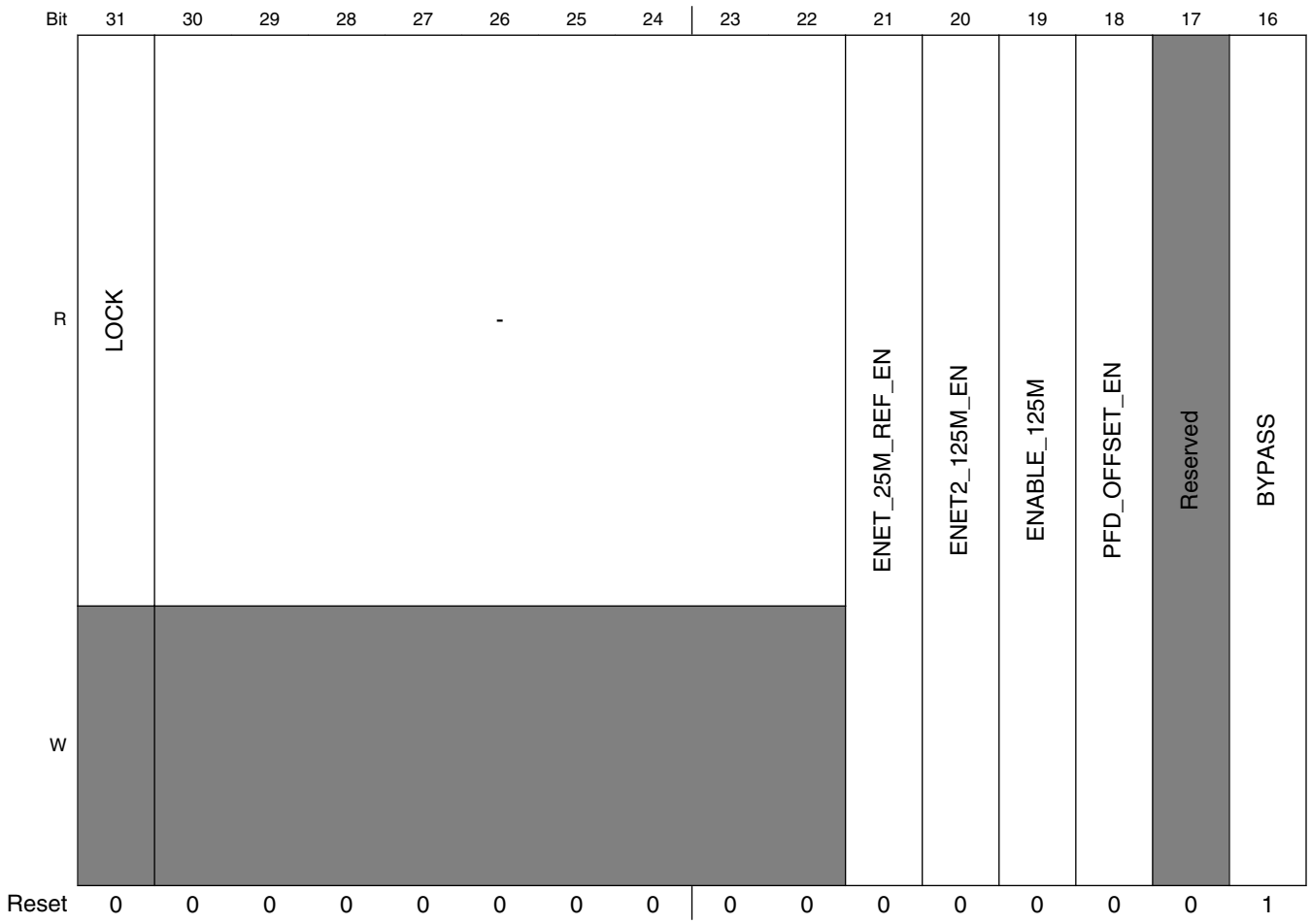
CCM_ANALOG_PLL_VIDEO_DENOM field descriptions

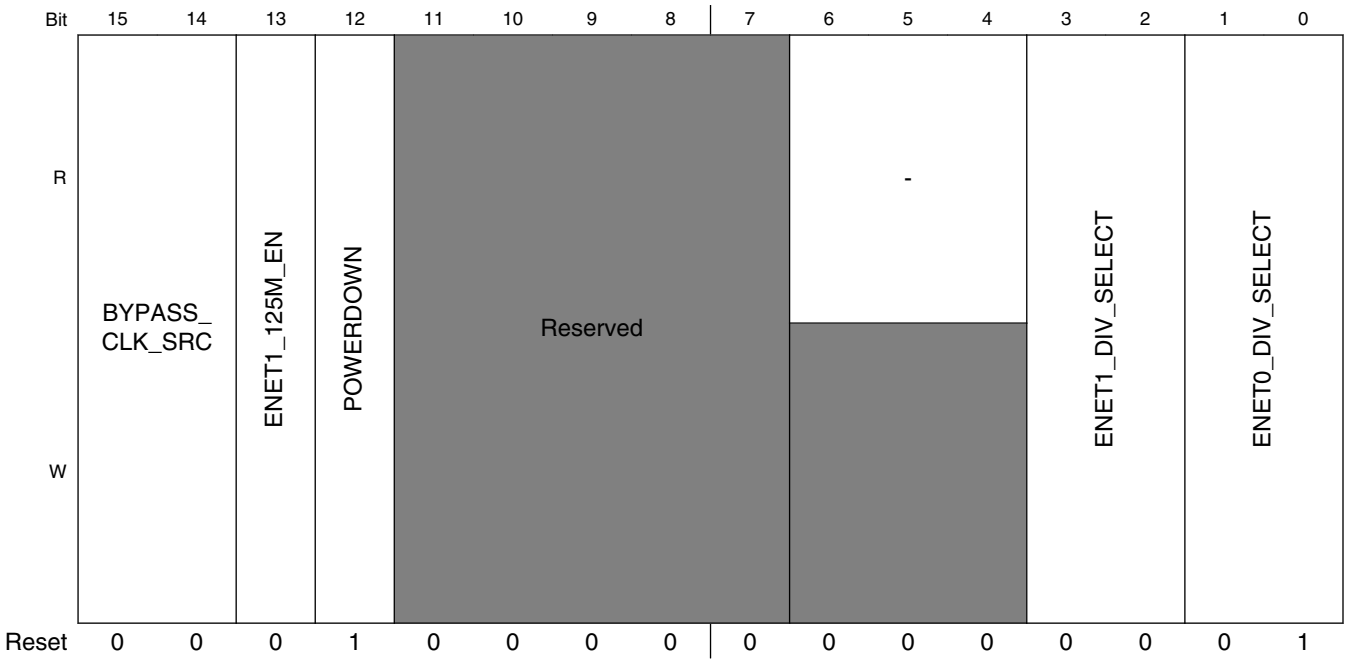
Field	Description
31–30 -	Always set to zero (0).
B	30 bit Denominator of fractional loop divider.

18.7.14 Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET*n*)

The control register provides control for the ENET PLL.

Address: 20C_8000h base + E0h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PLL_ENET*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–22 -	Always set to zero (0).
21 ENET_25M_ REF_EN	Enable the PLL providing ENET 25 MHz reference clock
20 ENET2_125M_ EN	Enable the PLL providing the ENET2 125 MHz reference clock
19 ENABLE_125M	Enables an offset in the phase frequency detector.
18 PFD_OFFSET_ EN	Enables an offset in the phase frequency detector.
17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved** — 0x3 **Reserved** —

Table continues on the next page...

CCM_ANALOG_PLL_ENET*n* field descriptions (continued)

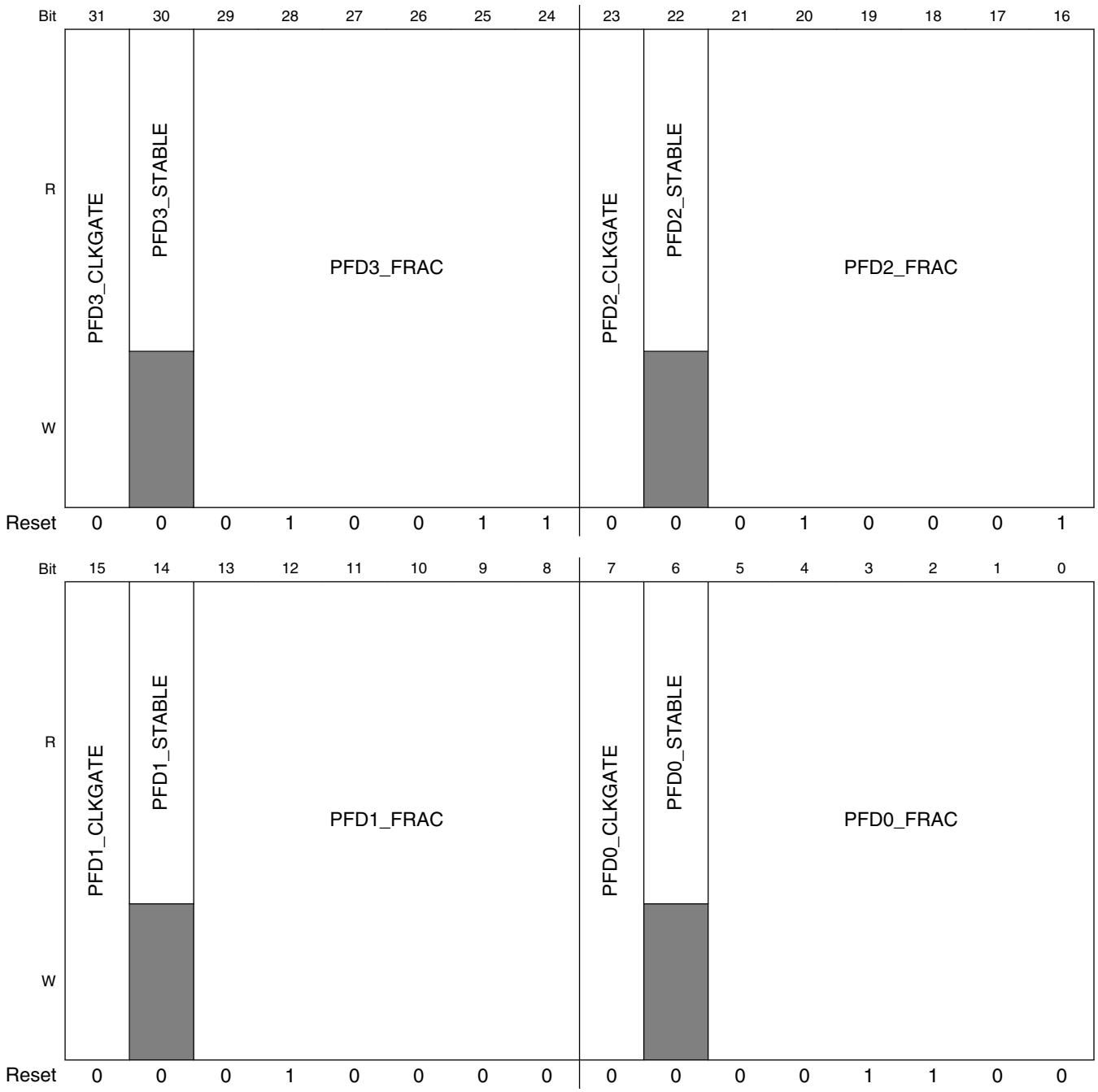
Field	Description
13 ENET1_125M_EN	Enable the PLL providing the ENET1 125 MHz reference clock.
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.
6–4 -	Always set to zero (0).
3–2 ENET1_DIV_SELECT	Controls the frequency of the ethernet1 reference clock. 00 25MHz 01 50MHz 10 100MHz (not 50% duty cycle) 11 125MHz
ENET0_DIV_SELECT	Controls the frequency of the ethernet0 reference clock. 00 25MHz 01 50MHz 10 100MHz (not 50% duty cycle) 11 125MHz

18.7.15 480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480*n*)

The PFD_480 control register provides control for PFD clock generation.

This register controls the 4-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Address: 20C_8000h base + F0h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PFD_480*n* field descriptions

Field	Description
31 PFD3_CLKGATE	IO Clock Gate. If set to 1, the 3rd fractional divider clock (reference ref_pfd3) is off (power savings). 0: ref_pfd3 fractional divider clock is enabled. Need to assert this bit before PLL is powered down
30 PFD3_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit,

Table continues on the next page...

CCM_ANALOG_PFD_480*n* field descriptions (continued)

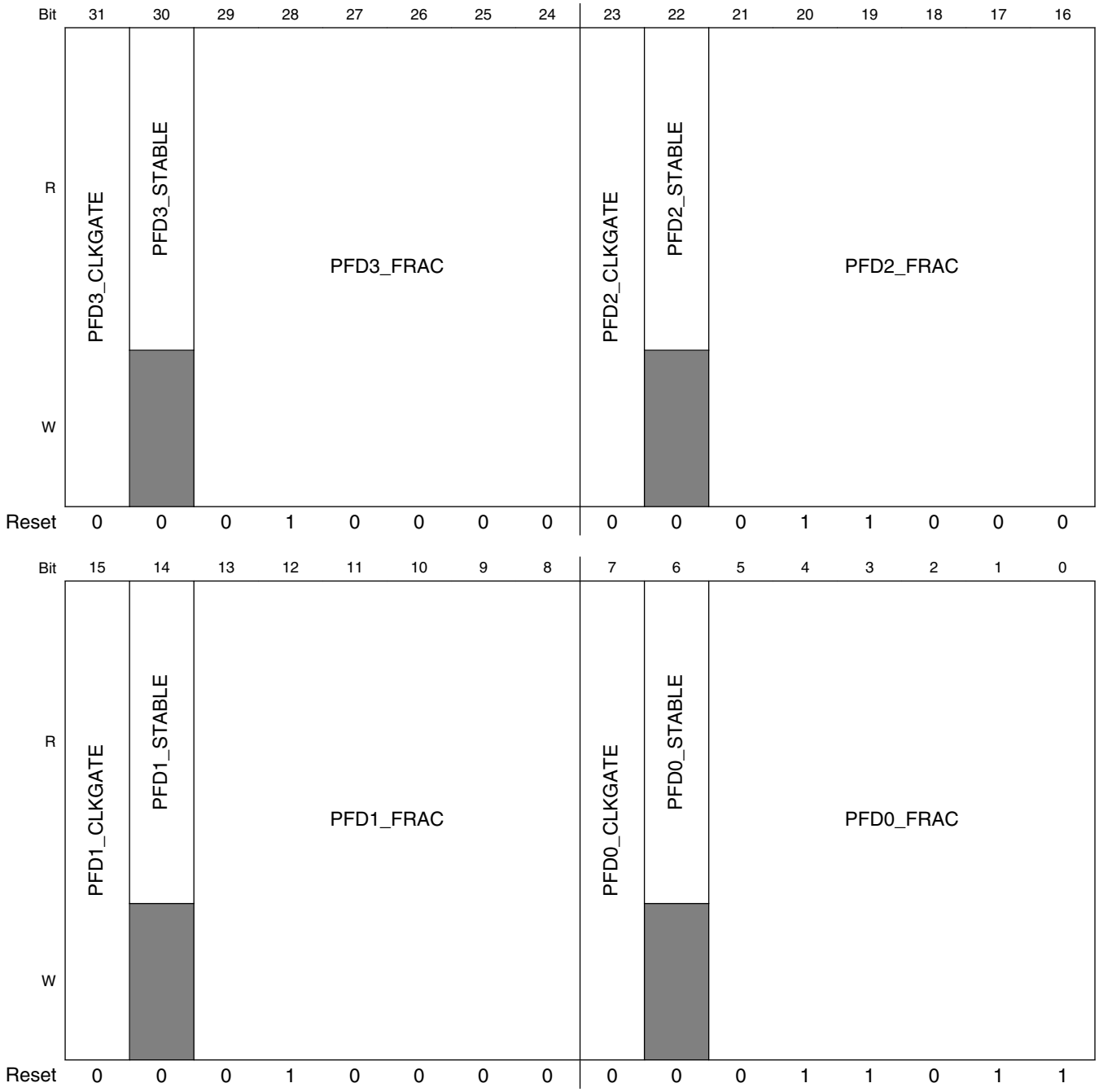
Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD3_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD3_FRAC where PFD3_FRAC is in the range 12-35.
23 PFD2_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd2) is off (power savings). 0: ref_pfd2 fractional divider clock is enabled. Need to assert this bit before PLL is powered down
22 PFD2_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD2_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD2_FRAC where PFD2_FRAC is in the range 12-35.
15 PFD1_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd1) is off (power savings). 0: ref_pfd1 fractional divider clock is enabled. Need to assert this bit before PLL is powered down
14 PFD1_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD1_FRAC where PFD1_FRAC is in the range 12-35.
7 PFD0_CLKGATE	If set to 1, the IO fractional divider clock (reference ref_pfd0) is off (power savings). 0: ref_pfd0 fractional divider clock is enabled. Need to assert this bit before PLL is powered down
6 PFD0_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
PFD0_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD0_FRAC where PFD0_FRAC is in the range 12-35.

18.7.16 528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528*n*)

The PFD_528 control register provides control for PFD clock generation.

This register controls the 3-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Address: 20C_8000h base + 100h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PFD_528*n* field descriptions

Field	Description
31 PFD3_CLKGATE	IO Clock Gate. If set to 1, the 3rd fractional divider clock (reference ref_pfd3) is off (power savings). 0: ref_pfd3 fractional divider clock is enabled. Need to assert this bit before PLL powered down
30 PFD3_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit,

Table continues on the next page...

CCM_ANALOG_PFD_528*n* field descriptions (continued)

Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD3_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD3_FRAC where PFD3_FRAC is in the range 12-35.
23 PFD2_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd2) is off (power savings). 0: ref_pfd2 fractional divider clock is enabled. Need to assert this bit before PLL powered down
22 PFD2_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD2_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD2_FRAC where PFD2_FRAC is in the range 12-35. **NOTE:** The maximum allowed frequency of PFD2 is 400MHz
15 PFD1_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd1) is off (power savings). 0: ref_pfd1 fractional divider clock is enabled. Need to assert this bit before PLL powered down
14 PFD1_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD1_FRAC where PFD1_FRAC is in the range 12-35.
7 PFD0_CLKGATE	If set to 1, the IO fractional divider clock (reference ref_pfd0) is off (power savings). 0: ref_pfd0 fractional divider clock is enabled. Need to assert this bit before PLL powered down
6 PFD0_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
PFD0_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD0_FRAC where PFD0_FRAC is in the range 12-35. **NOTE:** For QSPI boot at 76 MHz, this PFD is relocked to 307MHz, so the default value for this field (modified by ROM) would be 0x1f. Also for Low Freq Boot, ROM will relock this PFD to 307MHz, hence default would be 0x1f.

18.7.17 Miscellaneous Register 0 (CCM_ANALOG_MISC0*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 20C_8000h base + 150h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VID_PLL_PREDIV	XTAL_24M_PWD	RTC_XTAL_SOURCE	CLKGATE_DELAY			CLKGATE_CTRL	Reserved								OSC_XTALOK_EN
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OSC_XTALOK	OSC_I		DISCON_HIGH_SNVS	STOP_ MODE_ CONFIG		Reserved		REFTOP_VBGUP	REFTOP_VBGADJ			REFTOP_SELFBIASOFF	Reserved		REFTOP_PWD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_MISC0*n* field descriptions

Field	Description
31 VID_PLL_ PREDIV	Predivider for the source clock of the PLL's. 0 Divide by 1 1 Divide by 2
30 XTAL_24M_PWD	This field powers down the 24M crystal oscillator if set true. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC)
29 RTC_XTAL_ SOURCE	This field indicates which chip source is being used for the rtc clock. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 0 Internal ring oscillator 1 RTC_XTAL
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 000 0.5ms 001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms 101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–17 -	This field is reserved. Always set to zero.
16 OSC_XTALOK_ EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC)
15 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC)

Table continues on the next page...

CCM_ANALOG_MISC0*n* field descriptions (continued)

Field	Description
14–13 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%
12 DISCON_HIGH_SNVS	This bit controls a switch from VDD_HIGH_IN to VDD_SNVS_IN. 0 Turn on the switch 1 Turn off the switch
11–10 STOP_MODE_CONFIG	Configure the analog behavior in stop mode. 00 All analog except rtc powered down on stop mode assertion. XtalOsc=on, RCOsc=off; 01 Certain analog functions such as certain regulators left up. XtalOsc=on, RCOsc=off; 10 XtalOsc=off, RCOsc=on, Old BG=on, New BG=off. 11 XtalOsc=off, RCOsc=on, Old BG=off, New BG=on.
9–8 -	This field is reserved. Reserved
7 REFTOP_VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
6–4 REFTOP_VBGADJ	**NOTE:** Not related to CCM. See Power Management Unit (PMU) 000 Nominal VBG 001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2–1 -	This field is reserved.

Table continues on the next page...

CCM_ANALOG_MISC0*n* field descriptions (continued)

Field	Description
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, PLL, and other analog functions on the die. **NOTE:** Not related to CCM. See Power Management Unit (PMU)

18.7.18 Miscellaneous Register 1 (CCM_ANALOG_MISC1*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 20C_8000h base + 160h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IRQ_DIG_BO	IRQ_ANA_BO	IRQ_TEMPHIGH	IRQ_TEMPLOW	IRQ_TEMPPANIC	Reserved									PFD_528_AUTOGATE_EN	PFD_480_AUTOGATE_EN
W	w1c	w1c	w1c	w1c	w1c											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Reserved	LVDSCLK1_IBEN	Reserved	LVDSCLK1_OBEN	Reserved					LVDS1_CLK_SEL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_MISC1*n* field descriptions

Field	Description
31 IRQ_DIG_BO	This status bit is set to one when when any of the digital regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
30 IRQ_ANA_BO	This status bit is set to one when when any of the analog regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
29 IRQ_TEMPHIGH	This status bit is set to one when the temperature sensor high interrupt asserts for high temperature. **NOTE:** Not related to CCM. See Temperature Monitor (TEMPMON)
28 IRQ_TEMPLOW	This status bit is set to one when the temperature sensor low interrupt asserts for low temperature. **NOTE:** Not related to CCM. See Temperature Monitor (TEMPMON)
27 IRQ_ TEMPPANIC	This status bit is set to one when the temperature sensor panic interrupt asserts for a panic high temperature. **NOTE:** Not related to CCM. See Temperature Monitor (TEMPMON)
26–18 -	This field is reserved. Reserved
17 PFD_528_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_528 clocks anytime the PLL_528 is unlocked or powered off.
16 PFD_480_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_480 clocks anytime the USB1_PLL_480 is unlocked or powered off.
15–14 -	This field is reserved. Reserved
13 -	This field is reserved. Reserved
12 LVDSCLK1_ IBEN	This enables the LVDS input buffer for anaclk1/1b. Do not enable input and output buffers simultaneously.
11 -	This field is reserved. Reserved
10 LVDSCLK1_ OBEN	This enables the LVDS output buffer for anaclk1/1b. Do not enable input and output buffers simultaneously.
9–5 -	This field is reserved. Reserved
LVDS1_CLK_ SEL	This field selects the clk to be routed to anaclk1/1b. 00000 **ARM_PLL** — Arm PLL 00001 **SYS_PLL** — System PLL 00010 **PFD4** — ref_pfd4_clk == pll2_pfd0_clk

Table continues on the next page...

CCM_ANALOG_MISC1*n* field descriptions (continued)

Field	Description	
	00011	**PFD5** — ref_pfd5_clk == pll2_pfd1_clk
	00100	**PFD6** — ref_pfd6_clk == pll2_pfd2_clk
	00101	**PFD7** — ref_pfd7_clk == pll2_pfd3_clk
	00110	**AUDIO_PLL** — Audio PLL
	00111	**VIDEO_PLL** — Video PLL
	01001	**ETHERNET_REF** — ethernet ref clock (ENET_PLL)
	01100	**USB1_PLL** — USB1 PLL clock
	01101	**USB2_PLL** — USB2 PLL clock
	01110	**PFD0** — ref_pfd0_clk == pll3_pfd0_clk
	01111	**PFD1** — ref_pfd1_clk == pll3_pfd1_clk
	10000	**PFD2** — ref_pfd2_clk == pll3_pfd2_clk
	10001	**PFD3** — ref_pfd3_clk == pll3_pfd3_clk
	10010	**XTAL** — xtal (24M)
	10101 to 11111	ref_pfd7_clk == pll2_pfd3_clk

18.7.19 Miscellaneous Register 2 (CCM_ANALOG_MISC2*n*)

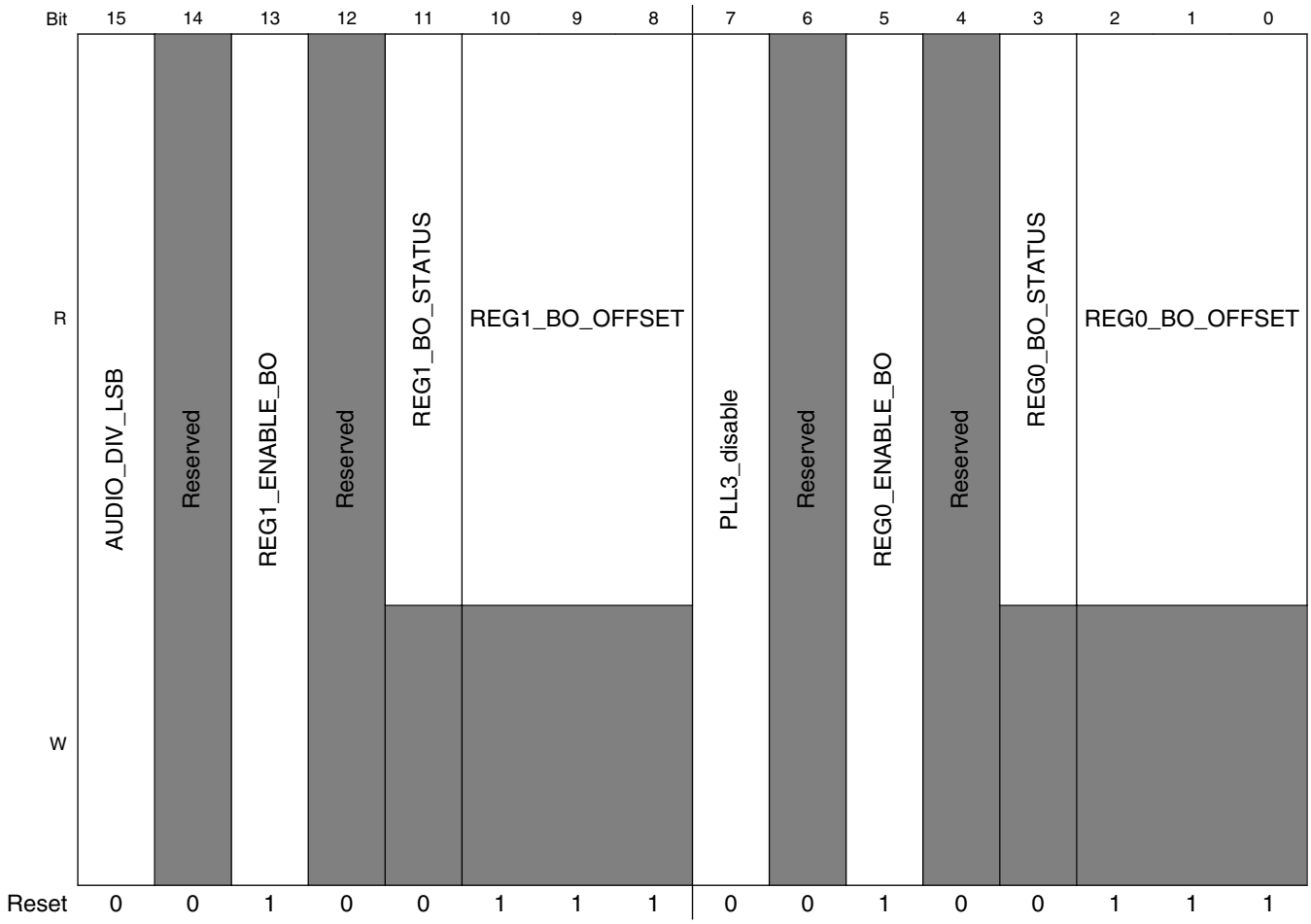
This register defines the control for miscellaneous analog blocks.

NOTE

This register is shared with PMU.

Address: 20C_8000h base + 170h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VIDEO_DIV		REG2_STEP_TIME		REG1_STEP_TIME		REG0_STEP_TIME		AUDIO_DIV_MSB	REG2_OK	REG2_ENABLE_BO	Reserved	REG2_BO_STATUS	REG2_BO_OFFSET		
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1



CCM_ANALOG_MISC2*n* field descriptions

Field	Description
31–30 VIDEO_DIV	Post-divider for video. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_VIDEOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
29–28 REG2_STEP_TIME	Number of clock periods (24MHz clock). **NOTE:** Not related to CCM. See Power Management Unit (PMU) 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
27–26 REG1_STEP_TIME	Number of clock periods (24MHz clock).

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
	NOTE: Not related to CCM. See Power Management Unit (PMU) 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
25–24 REG0_STEP_TIME	Number of clock periods (24MHz clock). **NOTE:** Not related to CCM. See Power Management Unit (PMU) 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
23 AUDIO_DIV_MSB	MSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** MSB bit value pertains to the first bit, please program the LSB bit (bit 15) as well to change divider value for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
22 REG2_OK	Signals that the voltage is above the brownout level for the SOC supply. 1 = regulator output > brownout_target **NOTE:** Not related to CCM. See Power Management Unit (PMU)
21 REG2_ENABLE_BO	Enables the brownout detection. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
20 -	This field is reserved.
19 REG2_BO_STATUS	Reg2 brownout status bit. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
18–16 REG2_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
15 AUDIO_DIV_LSB	LSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** LSB bit value pertains to the last bit, please program the MSB bit (bit 23) as well, to change divider value for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
14 - Reserved	This field is reserved.
13 REG1_ENABLE_BO	Enables the brownout detection. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
12 -	This field is reserved.
11 REG1_BO_STATUS	Reg1 brownout status bit. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 1 Brownout, supply is below target minus brownout offset.
10–8 REG1_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V
7 PLL3_disable	When USB is in low power suspend mode this Control bit is used to indicate if other system peripherals require the USB PLL3 clock when the SoC is not in low power mode. A user needs to set this bit if they want to optionally disable PLL3 while the SoC is not in any low power mode to save power. When the system does go into low power mode this bit setting would not have any affect. **NOTE:** When USB is in low power suspend mode users would need to ensure PLL3 is not being used before setting this bit in RUN mode. Please refer to the correct PLL disabling procedure in Disabling / Enabling PLLs 0 PLL3 is being used by peripherals and is enabled when SoC is not in any low power mode 1 PLL3 can be disabled when the SoC is not in any low power mode
6 -	This field is reserved.
5 REG0_ENABLE_BO	Enables the brownout detection. **NOTE:** Not related to CCM. See Power Management Unit (PMU)

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
4 -	This field is reserved.
3 REG0_BO_ STATUS	Reg0 brownout status bit. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 1 Brownout, supply is below target minus brownout offset.
REG0_BO_ OFFSET	This field defines the brown out voltage offset for the CORE power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. Some steps may be irrelevant because of input supply limitations or load operation. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V

Chapter 19
CMOS Sensor Interface (CSI)

19.1 Overview

This chapter presents the CMOS Sensor Interface (CSI) architecture, operation principles, and programming model.

The CSI enables the chip to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only Bayer and statistics data, while smart sensors support CCIR656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

The capabilities of the CSI include:

- Configurable interface logic to support most commonly available CMOS sensors.
- Support for CCIR656 video interface as well as traditional sensor interface.
- 8-bit / 16-bit / 24-bit data port for YCbCr, YUV, or RGB data input.
- 8-bit / 10-bit / 16-bit data port for Bayer data input.
- Full control of 8-bit/pixel, 10-bit/pixel or 16-bit / pixel data format to 64-bitreceive FIFO packing.
- 256 x 64 FIFO to store received image pixel data.
- Reveive FIFO overrun protection mechanism.
- Embedded DMA controllers to transfer data from receive FIFO or statistic FIFO through AHB bus.
- Support 2D DMA transfer from the receive FIFO to the frame buffers in the external memory.
- Support double bufferring two frames in the external memory.
- Single interrupt source to interrupt controller from maskable interrupt sources: Start of Frame, End of Frame, Change of Field, FIFO full, FIFO overrun, DMA transfer done, CCIR error and AHB bus response error.
- Configurable master clock frequency output to sensor.

- Statistic data generation for Auto Exposure (AE) and Auto White Balance (AWB) control of the camera (only for Bayer data and 8-bit/pixel format).
- Supports simple deinterlacing of interlaced input.
- Supports TV Decoder (VADC) input.

19.2 External Signals

The table below describes the external signals for the CSI. The external signals are tied between the CSI module and an external CMOS sensor.

Table 19-1. CSI External Signals

Signal	Description	Pad	Mode	Direction
DATA0	Data Sensor Signal 0, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA17	ALT3	I
		UART3_RX_DATA	ALT3	I
DATA1	Data Sensor Signal 1, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA16	ALT3	I
		UART3_TX_DATA	ALT3	I
DATA2	Data Sensor Signal 2, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA00	ALT0	I
		UART1_TX_DATA	ALT3	I
DATA3	Data Sensor Signal 3 part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA01	ALT0	I
		UART1_RX_DATA	ALT3	I
DATA4	Data Sensor Signal 4, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA02	ALT0	I
		UART1_CTS_B	ALT3	I
DATA5	Data Sensor Signal 5, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA03	ALT0	I
		UART1_RTS_B	ALT3	I
DATA6	Data Sensor Signal 6, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA04	ALT0	I
		UART2_TX_DATA	ALT3	I
DATA7	Data Sensor Signal 7 part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA05	ALT0	I
		UART2_RX_DATA	ALT3	I
DATA8	Data Sensor Signal 8, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA06	ALT0	I
		UART2_CTS_B	ALT3	I
DATA9	Data Sensor Signal 9 part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	CSI_DATA07	ALT0	I
		UART2_RTS_B	ALT3	I
DATA10	Data Sensor Signal 10, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA18	ALT3	I
		UART3_CTS_B	ALT3	I

Table continues on the next page...

Table 19-1. CSI External Signals (continued)

Signal	Description	Pad	Mode	Direction
DATA11	Data Sensor Signal 11, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA19	ALT3	I
		UART3_RTS_B	ALT3	I
DATA12	Data Sensor Signal 12, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA20	ALT3	I
		UART4_TX_DATA	ALT3	I
DATA13	Data Sensor Signal 13, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA21	ALT3	I
		UART4_RX_DATA	ALT3	I
DATA14	Data Sensor Signal 14, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA22	ALT3	I
		UART5_TX_DATA	ALT3	I
DATA15	Data Sensor Signal 15, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD_DATA23	ALT3	I
		UART5_RX_DATA	ALT3	I
DATA16	Data Sensor Signal 16, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_RX_DATA0	ALT3	I
		LCD_DATA08	ALT3	I
DATA17	Data Sensor Signal 17 part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_RX_DATA1	ALT3	I
		LCD_DATA09	ALT3	I
DATA18	Data Sensor Signal 18, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_RX_EN	ALT3	I
		LCD_DATA10	ALT3	I
DATA19	Data Sensor Signal 19 part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_TX_DATA0	ALT3	I
		LCD_DATA11	ALT3	I
DATA20	Data Sensor Signal 20, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_TX_DATA1	ALT3	I
		LCD_DATA12	ALT3	I
DATA21	Data Sensor Signal 21, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_TX_EN	ALT3	I
		LCD_DATA13	ALT3	I
DATA22	Data Sensor Signal 22, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_TX_CLK	ALT3	I
		LCD_DATA14	ALT3	I
DATA23	Data Sensor Signal 23, part of 24-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ENET1_RX_ER	ALT3	I
		LCD_DATA15	ALT3	I
FIELD	CSI Field Signal	GPIO1_IO05	ALT3	I
		NAND_DQS	ALT1	I
HSYNC	Horizontal Sync (Blank Signal)	CSI_HSYNC	ALT0	I
		GPIO1_IO09	ALT3	I
MCLK	"CMOS Sensor Master Clock *Note: MCLK is provided by the CCM module directly, not from the CSI module itself"	CSI_MCLK	ALT0	O
		GPIO1_IO06	ALT3	O
PIXCLK	Pixel Clock	CSI_PIXCLK	ALT0	I

Table continues on the next page...

Table 19-1. CSI External Signals (continued)

Signal	Description	Pad	Mode	Direction
		GPIO1_IO07	ALT3	I
VSYNC	Vertical Sync (Start Of Frame)	CSI_VSYNC	ALT0	I
		GPIO1_IO08	ALT3	I

19.3 Clocks

The following table describes the clock sources for CSI. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 19-2. CSI Clocks

Clock name	Clock Root	Description
csi_hclk	ahb_clk_root	Module clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
ipg_clk_s_raw	ipg_clk_root	Peripheral raw data clock

19.4 Principles of Operation

The information found here describes the modes of operation of the sensor interface.

The CSI is designed to support generic sensor interface timing as well as CCIR656 video interface timing. Traditional CMOS sensors typically use VSYNC (SOF), HSYNC (BLANK), and PIXCLK signals to output Bayer or YUV data. Smart CMOS sensors, that come with on-chip imaging processing, usually support video mode transfer. They use an embedded timing codec to replace the VSYNC and HSYNC signal. The timing codec is defined by the CCIR656 standard.

The CSI can support connection with the sensor as follows.

- To connect with one 8-bit sensor, the sensor data interface should connect to CSI_DATA[9:2].
- To connect with one 10-bit sensor, the sensor data interface should connect to CSI_DATA[9:0].
- To connect with one 16-bit sensor, the sensor data interface should connect to CSI_DATA[15:0].

- To connect with one 24-bit data, either video pass-through or TV Decoder input, the sensor data interface should connect to CSI_DATA[23:0].
- To connect with two 8-bit sensors, the sensor data interfaces should connect to CSI_DATA[7:0] and CSI_DATA[15:8].

Table 19-3. CSI input data format

Signal Name	TVdecoder YCbCr 1 Cycle	RGB888 1 Cycle	RGB888/ YUV4444 3 Cycle	RGB666 1 Cycle	RGB565 1 Cycle	YCbCr422 1 Cycle	YCbCr422 2 Cycle	Generic 10 bit	CCIR656
ipp_csi_d[23]	Y[7]	R[7]		R[5]					
ipp_csi_d[22]	Y[6]	R[6]		R[4]					
ipp_csi_d[21]	Y[5]	R[5]		R[3]					
ipp_csi_d[20]	Y[4]	R[4]		R[2]					
ipp_csi_d[19]	Y[3]	R[3]		R[1]					
ipp_csi_d[18]	Y[2]	R[2]		R[0]					
ipp_csi_d[17]	Y[1]	R[1]		Y[5]					
ipp_csi_d[16]	Y[0]	R[0]		R[4]					
ipp_csi_d[15]	Cb[7]	G[7]		G[5]	R[4]	Y[7]			
ipp_csi_d[14]	Cb[6]	G[6]		G[4]	R[3]	Y[6]			
ipp_csi_d[13]	Cb[5]	G[5]		G[3]	R[2]	Y[5]			
ipp_csi_d[12]	Cb[4]	G[4]		G[2]	R[1]	Y[4]			
ipp_csi_d[11]	Cb[3]	G[3]		G[1]	R[0]	Y[3]			
ipp_csi_d[10]	Cb[2]	G[2]		G[0]	G[5]	Y[2]			
ipp_csi_d[9]	Cb[1]	G[1]	R/G/B[7]	G[5]	G[4]	Y[1]	Y/C[7]	Ge[9]	C/Y[7]
ipp_csi_d[8]	Cb[0]	G[0]	R/G/B[6]	G[4]	G[3]	Y[0]	Y/C[6]	Ge[8]	C/Y[6]
ipp_csi_d[7]	Cr[7]	B[7]	R/G/B[5]	B[5]	G[2]	C[7]	Y/C[5]	Ge[7]	C/Y[5]
ipp_csi_d[6]	Cr[6]	B[6]	R/G/B[4]	B[4]	G[1]	C[6]	Y/C[4]	Ge[6]	C/Y[4]
ipp_csi_d[5]	Cr[5]	B[5]	R/G/B[3]	B[3]	G[0]	C[5]	Y/C[3]	Ge[5]	C/Y[3]

Table continues on the next page...

Table 19-3. CSI input data format (continued)

Signal Name	TVdecoder YCbCr 1 Cycle	RGB888 1 Cycle	RGB888/ YUV4444 3 Cycle	RGB666 1 Cycle	RGB565 1 Cycle	YCbCr422 1 Cycle	YCbCr422 2 Cycle	Generic 10 bit	CCIR656
ipp_csi_d[4]	Cr[4]	B[4]	R/G/B[2]	B[2]	B[4]	C[4]	Y/C[2]	Ge[4]	C/Y[2]
ipp_csi_d[3]	Cr[3]	B[3]	R/G/B[1]	B[1]	B[3]	C[3]	Y/C[1]	Ge[3]	C/Y[1]
ipp_csi_d[2]	Cr[2]	B[2]	R/G/B[0]	B[0]	B[2]	C[2]	Y/C[0]	Ge[2]	C/Y[0]
ipp_csi_d[1]	Cr[1]	B[1]		B[5]	B[1]	C[1]		Ge[1]	
ipp_csi_d[0]	Cr[0]	B[0]		B[4]	B[0]	C[0]		Ge[0]	

19.4.1 Data Transfer with the Embedded DMA Controllers

The CSI has two embedded DMA controllers, one for the receive FIFO and the other for the statistic FIFO. It supports 2D DMA transfer from the receive FIFO to the frame buffers in the external memory and linear DMA transfer from the statistic FIFO.

To transfer data from the RxFIFO to the external memory, the user should set the start address in the frame buffer where the transferred data is stored, the parameters of the frame buffers, and the parameters of the image coming from the sensor. The user can have two frame buffers in the external memory. Each one will store a frame of image coming from the sensor. The embedded DMA controller will first write the frame buffer1 and then frame buffer2. These two frame buffers will be written by turns. The start address should be aligned in double words and set in the CSIDMASA-FB1 and CSIDMASA-FB2 registers. In the CSIFBUF_PARA register, the user should set the stride of the frame buffer to show how many double words to skip before starting to write the next row of the image. In the CSIIMAG_PARA register, the user should set the width and height of the image coming from the sensor. The RxFF_LEVEL and DMA_REQ_EN_RFF bits in CSICR3 registers also need to be set before the data transfer starts. When the number of the data in the RxFIFO reaches the trigger level, a DMA request will be sent to the embedded DMA controller and the data will be read out from the RxFIFO and written through AHB bus into the external frame buffers. The burst type of transfer can be INCR4, INCR8 and INCR16 by setting DMA_BURST_TYPE_RFF bits in CSICR2 register. After all data in an image frame are transferred, the DMA_TSF_DONE_FB1 or DMA_TSF_DONE_FB2 bit will be set in CSISR register and the interrupt can be triggered if the corresponding enable bit is set in CSICR1 register. The DMA_REFLASH_RFF bit in CSICR3 can be used to activate or restart the embedded DMA controller.

The RxFIFO has the overrun protection mechanism in case the RxFIFO is overrun during data transfer. If the RxFIFO is full and more data needs to be received during the data transfer, the RxFIFO will be overwritten continuously and all 128 words of data in the RxFIFO before overrun occurred will be discarded; the corresponding 128 words memory space in the frame buffer will keep the previous values.

To transfer data from the statistic FIFO to the external memory, the user should set the start address of the externl memory where the transferred data is stored and the total transfer sizes. The start address and the transfer sizes are all aligned in double words and should be set in the CSIDMASA-STATFIFO and CSIDMATS-STATFIFO registers. The STATFF_LEVEL and DMA_REQ_EN_SFF bits in CSICR3 registers should also be set before the data transfer starts. When the number of the data in the STATFIFO reaches the trigger level, a dma request will be sent to the embedded DMA controller and the data will be read out from the STATFIFO and written through AHB bus into the external memory. The burst type of transfer can be INCR4, INCR8 and INCR16 by setting DMA_BURST_TYPE_SFF bits in CSICR2 register. After all expected data (defined by the total transfer sizes) are transferred, the DMA_TSF_DONE_SFF bit will be set in CSISR register and an interrupt can be triggered if the SFF_DMA_DONE_INTEN is enabled in CSICR1 register. The DMA_REFLASH_SFF bit in CSICR3 can be used to activate or re-start the embedded DMA controller.

19.4.2 Gated Clock Mode

VSYNC, HSYNC, and PIXCLK signals are used in gated clock mode.

A frame starts with an active edge on VSYNC, then HSYNC asserts and holds for the entire line. The Pixel clock is valid as long as HSYNC is asserted. Data is latched at the active edge of the valid pixel clocks. HSYNC deasserts at the end of line. Pixel clocks then become invalid and CSI stops receiving data from the stream. For the next line the HSYNC timing repeats. For the next frame the VSYNC timing repeats.

19.4.3 Non-Gated Clock Mode

In non-gated clock mode, only the VSYNC and PIXCLK signals are used; the HSYNC signal is ignored.

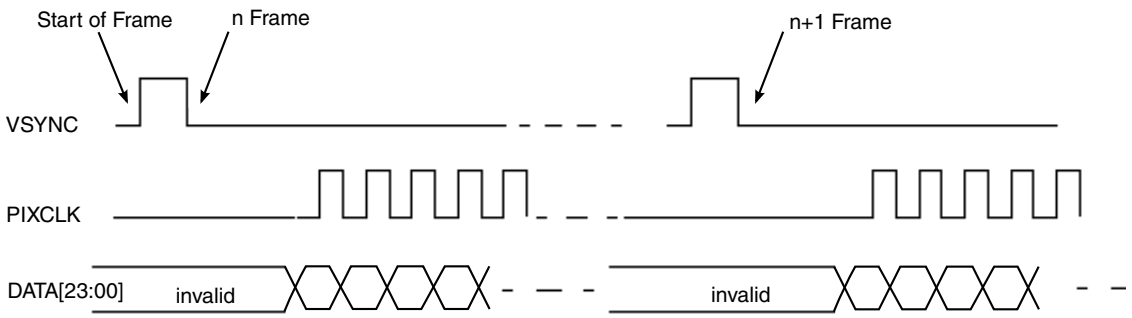


Figure 19-1. Non-Gated Clock Mode Timing Diagram

The overall timing of non-gated mode is the same as the gated-clock mode, except for the HSYNC signal. HSYNC signal is ignored by the CSI. All incoming pixel clocks are valid and cause data to be latched into RxFIFO. The PIXCLK signal is inactive (states low) until valid data is ready to be transmitted over the bus.

Figure 19-1 shows the timing of a typical sensor. Other sensors may have the slightly different timing from that shown. The CSI can be programed to support rising/falling-edge triggered VSYNC, active-high/low HSYNC, and rising/falling-edge triggered PIXCLK.

Below is the TV Decoder input mode timing diagram.

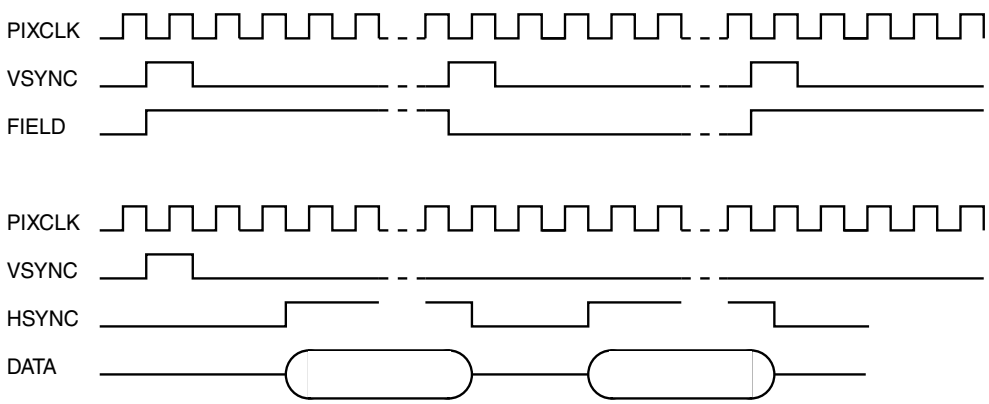


Figure 19-2. TV Decoder Input Mode Timing Diagram

19.4.4 CCIR656 Interlace Mode

In CCIR656 mode, only the PIXCLK and CSI_DATA[9:2] signals are used. The start of frame and blank signals are replaced by a timing codec which is embedded in the data stream. Each active line starts with an Start of Active Video (SAV) code and ends with

an End of Active Video (EAV) code. In some cases, digital blanking is inserted in between EAV and SAV code. The CSI decodes and filters out the timing-coding from the data stream, recovering VSYNC and HSYNC signals for internal use, such as statistical block control. Data is forwarded to the data receive and packing block in a sequential manner without reordering-that is, field 1 followed by field 2. The fields must be reordered in software to get back the original image.

Change of Field (COF) interrupt is triggered upon every field change. The interrupt service routine reads the status register to check for the current field.

According to the CCIR656 specification, the image must be in 625/50 PAL or 525/60 NTSC format. In addition, the image is interlaced into odd and even fields with vertical and horizontal blank data being filled into certain lines. Data must be in YCbCr422 format, each pixel contains 2 bytes, either Y + Cr or Y + Cb. These requirements are set for TV systems. The CSI module supports PAL and NTSC format only.

Figure below describes the frame structure in PAL system, showing vertical and horizontal blanking.

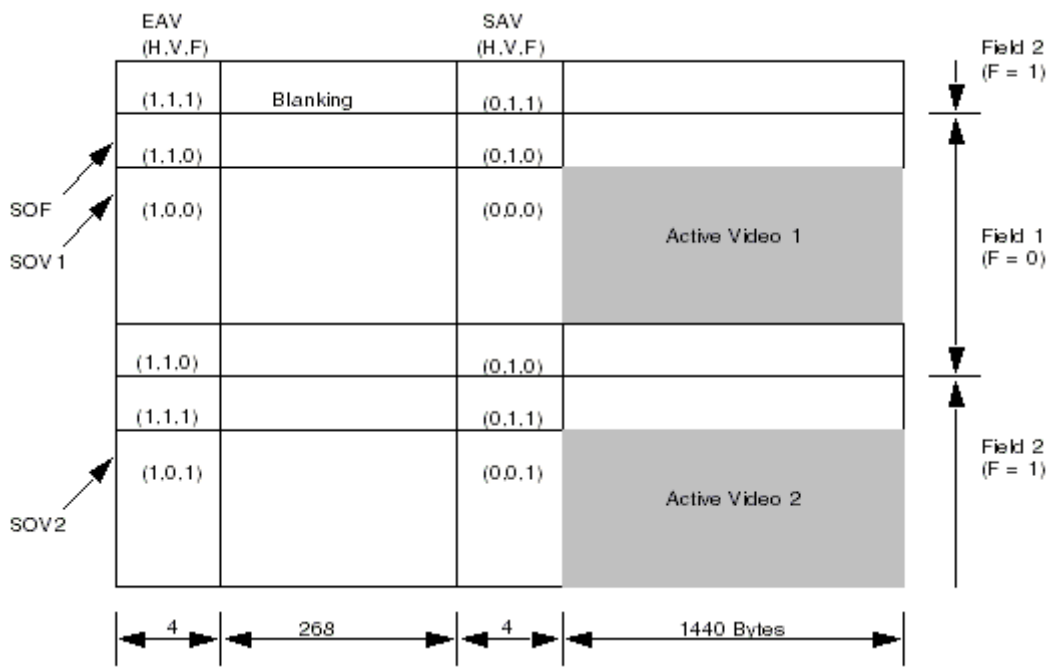


Figure 19-3. CCIR656 Interlace Mode (PAL)

Figure below describes the general timing for a single line, showing SAV and EAV.

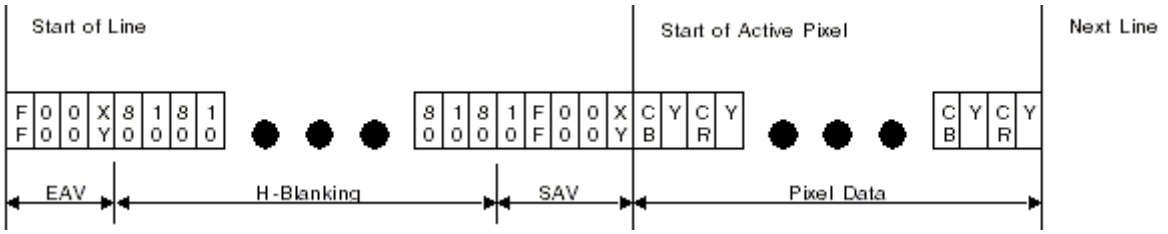


Figure 19-4. CCIR656 General Line Timing

The coding tables recommended by the CCIR656 specification are shown below. It is used in the CCIR656 mode to decode the video stream. An interrupt is generated for SOF, which is decoded from the embedded timing codec.

Table 19-4. Coding for SAV and EAV

Data Bit Number	1st Byte 0xFF	2nd Byte 0x00	3rd Byte 0x00	4th Byte 0xXY
7 (MSB)	1	0	0	1
6	1	0	0	F
5	1	0	0	V
4	1	0	0	H
3	1	0	0	P3
2	1	0	0	P2
1	1	0	0	P1
0	1	0	0	P0

Table 19-5. Codes with Protection bits for Error Detection/Correction

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

Table 19-6. Representations by F-Bit

F-Bit	Representations
0	ODD FIELD (FIELD 1)
1	EVEN FIELD (FIELD 2)

19.4.5 CCIR656 Progressive Mode

For a CMOS camera system of VGA or CIF resolution, strict adherence to the interlace requirements stated in the CIR standard is not required.

The image is considered to have only 1 active field which is scanned in a progressive manner. This active field is regarded as field 1 and the F-bit in the timing codec is ignored by the decoder. Most sensors support CCIR timing in this mode (progressive) by default.

Figure below shows the typical flow of progressive mode.

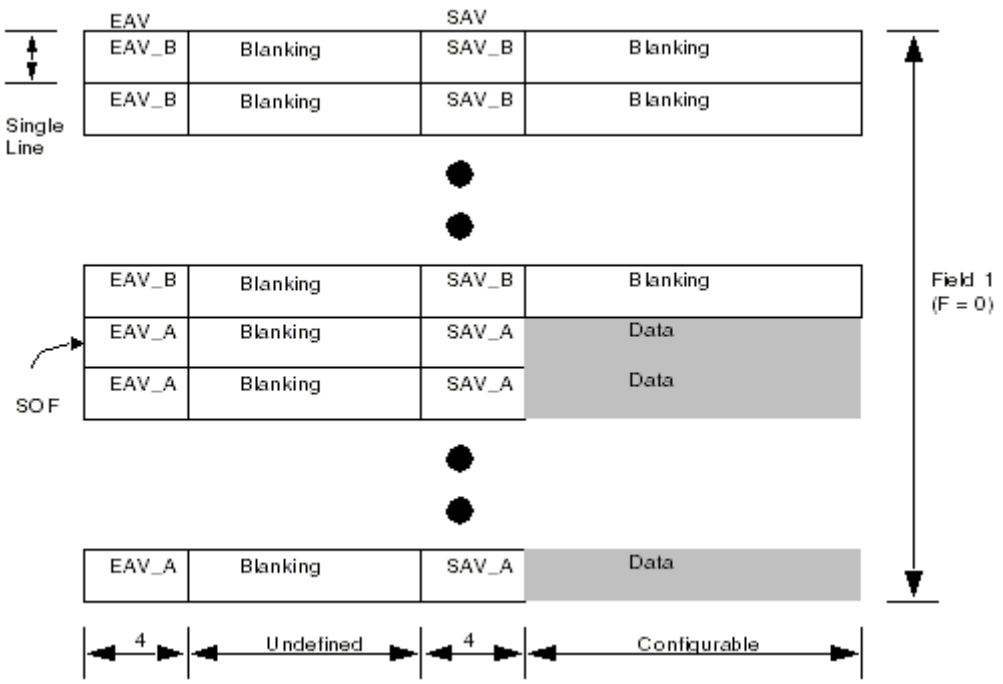


Figure 19-5. CCIR656 Progressive Mode (General Case)

An interrupt is generated for SOF but not for COF. In the general case, when SOF information is retrieved from the embedded coding, it is known as internal VSYNC mode. In other cases, when the VSYNC signal is provided by the sensor, it is known as external VSYNC mode. The CSI can be operated in internal or external VSYNC mode.

19.4.6 Error Correction for CCIR656 Coding

According to the algorithm for CCIR coding, protection bits in the SAV and EAV are encoded in the way that allows a 1-bit error to be corrected, or a 2-bit error to be detected by the decoder. This feature is supported by the interlace mode CCIR decoder in CSI.

For the 1-bit error case, users can select the error to be corrected automatically, or simply shown as a status flag instead. For the 2-bit error case, because the decoder is unable to make a correction, the error would be shown as a status flag only.

An interrupt can be generated upon the detection of an error. This signal can be enabled or disabled without affecting the operation of the status bit.

19.5 Interrupt Generation

The information found here describes CSI events that generate interrupts.

19.5.1 Start Of Frame Interrupt (SOF_INT)

The source of an SOF interrupt is dependent on the mode of operation.

In traditional mode, VSYNC signal is taken from sensor and SOF_INT is generated at the rising or falling edge (programmable) of VSYNC.

In CCIR interlace mode, the SOF interrupt information is retrieved from the embedded coding and SOF_INT is generated.

In CCIR progressive mode, there are two sources of an SOF interrupt:

- In *internal* VSYNC mode, SOF is retrieved from the embedded coding.
- In *external* VSYNC mode, VSYNC is taken from the sensor and SOF is generated at the rising edge of VSYNC.

19.5.2 End Of Frame Interrupt (EOF_INT)

An EOF interrupt is generated when the frame ends and the complete frame data in RXFIFO is read.

The EOF event triggering works with the RX count register (CSIRXCNT). Software sets the RX count register to the frame size (in words). The CSI RX logic then counts the number of pixel data being received and compares it with the RX count. If the preset value is reached, an EOF interrupt is generated and the data in the RXFIFO are read. If a SOF event is detected before this happens, the EOF interrupt is not generated.

19.5.3 Change Of Field Interrupt (COF_INT)

The Change of Field interrupt is only valid in CCIR Interlace mode. The COF interrupt is generated when the field toggles, either from field 1 to field 2, or field 2 to field 1.

Software should first check COF_INT bit in the CSI Status Register (CSISTAT) before checking that F1_INT or F2_INT is turned on.

In PAL systems, the field changes at the beginning of the frame and coincides with SOF. For the first field, a COF interrupt is not generated, only an SOF. The COF interrupt is generated for the second field.

19.5.4 CCIR Error Interrupt (ECC_INT)

The CCIR Error Interrupt is only valid for CCIR Interlace mode. An ECC interrupt is generated when an error is found on the SAV or EAV codes in the incoming stream. When this happens, the ECC_INT status bit is set.

19.5.5 RxFIFO Full Interrupt (RxFF_INT)

A RxFIFO full interrupt is generated when the number of data in RXFIFO reaches the water mark defined by RxFF_LEVEL in CSICR3.

19.5.6 Statistic FIFO Full Interrupt (STATFF_INT)

A StatFIFO full interrupt is generated when the number of data in STATFIFO reaches the water mark defined by STATFF_LEVEL in CSICR3.

19.5.7 RxFIFO Overrun Interrupt (RFF_OR_INT)

A RxFIFO Overrun interrupt is generated when the RxFIFO has 128 words data and more data is being written in.

19.5.8 Statistic FIFO Overrun Interrupt (SFF_OR_INT)

A StatFIFO Overrun interrupt is generated when the STATFIFO has 64 words data and more data is being written in.

19.5.9 Frame Buffer1 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB1)

A DMA transfer done interrupt of frame buffer1 is generated when one frame of data are transferred from RxFIFO to the frame buffer1 in the external memory.

19.5.10 Frame Buffer2 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB2)

A DMA transfer done interrupt of frame buffer2 is generated when one frame of data are transferred from RxFIFO to the frame buffer2 in the external memory.

19.5.11 Statistic FIFO DMA Transfer Done Interrupt (DMA_TSF_DONE_SFF)

A StatFIFO DMA transfer done interrupt is generated when all the data are transferred from StatFIFO to the external memory. The transfer size is defined in the STATFIFO DMA Transfer Size Register.

19.5.12 AHB Bus Response Error Interrupt (HRESP_ERR_INT)

An AHB Bus response error interrupt is generated when a bus error is detected.

19.5.13 DMA Field 0 Transfer Done Interrupt (DMA_FIELD0_DONE)

A DMA transfer done interrupt of field 0 is generated when one field of data are transferred from RxFIFO to the frame buffer in external memory. This signal should work on interlaced mode.

19.5.14 DMA Field 1 Transfer Done Interrupt (DMA_FIELD1_DONE)

A DMA transfer done interrupt of field 1 is generated when one field of data are transferred from RxFIFO to the frame buffer in external memory. This signal should work on interlaced mode.

19.5.15 Base Address Change Error Interrupt (BASEADDR_CHANGE_ERROR)

A Base Address change error is generated when the base address changed while the last frame data transfer are not finished.

19.6 Data Packing Style

Careful attention to endianess is needed given the different port sizes at different stages of the image capture path.

To enable flexible packing of image data before storage in the FIFOs, the CSI module can swap data fields by use of the PACK_DIR and the SWAP16_EN bit in CSI Control Register 1 (CSICR1).

The CSI module accepts 8-bit, 10-bit or 16-bit data from the sensor by configuring PIXEL_BIT bit in CSI Control Register 1 (CSICR1) and TWO_8BIT_SENSOR bit in CSI Control Register3 (CSICR3). The input data is packed according to the setting of PACK_DIR bit. The packed data is stored in the RX FIFO according to the setting of the SWAP16_EN bit.

For 10-bit per pixel data format, each pixel is expanded to 16 bits by appending 6 zeros bits to the most significant bit. For 16-bit data format, the data path can be a combination by two 8-bit sensors. One sensor is connected to the CSI_DATA[7:0]. The other sensor is connected to CSI_DATA[15:8].

19.6.1 RX FIFO Path

19.6.1.1 Bayer Data

Bayer data is a type of raw data from the image sensor. This byte-wide data must be converted to the RGB space or YUV space by software. The data path for Bayer data is from the CSI to memory. If the system is in little endian, then the PACK_DIR bit should be set to 0. 8-bit data format from a sensor is packed to 64 bits as P7.P6.P5.P4.P3.P2.P1.P0, where P0 is the pixel coming in time slot 0 (first data) and P3 is the pixel coming in time slot 3 (the last data in the 64-bit word). When the data is addressed as bytes by software, P0 is transferred first, P1 is transferred next, and so on. 10-bit data format is packed to 64 bits as 000000.P3.000000.P2.000000.P1.000000.P0, where P0 is the 10-bit data coming in time slot 0 (first pixel) and P3 is the 10-bit data coming in time slot 3 (fourth pixel). 16-bit data, from two sensors, is packed to 64 bits as P7.P6.P5.P4.P3.P2.P1.P0, where P0 and P1 are the two 8-bit data coming in time slot 0 (P0 is the first pixel of the sensor connected with CSI_DATA[7:0] and P1 is the first pixel of the sensor connected with CSI_DATA[15:8]). P2 and P3 are the two 8-bit data coming in time slot 1 (second pixels of the two sensors).

19.6.1.2 RGB565 Data

RGB565 data is processed data from the image sensor, which can be put directly into the display buffer. The data is 16 bits wide. The data path is from CSI to memory to the display controller. On the sensor side, data must be transmitted as P0 first, followed by P1, and so on. For each pixel, whether the MSB or LSB is sent first depends on the endianness of the sensor. Data is 16 bits wide with the MSB labeled RG, and the LSB labeled GB. P0 is represented as RG0 and GB0.

CSI receives data in one of the following sequence:

- RG0, GB0, RG1, GB1, while RG0 comes out at time slot 0 (first data), and GB1 comes out at time slot 3 (last data)
- GB0, RG0, GB1, RG1

Using the first sequence as an example, and assuming the system is running in little endian, the data is presented as:

- 8-bit data from sensor: RG0, GB0, RG1, GB1, …
- 64-bit data before storage in the CSI RX FIFO (PACK_DIR bit = 1): RG0GB0RG1GB1RG2GB2RG3GB3

- 64-bitdata in CSI RX FIFO (SWAP16_EN bit enabled): RG3GB3RG2GB2RG1GB1RG0GB0
- 64-bit transfer to system memory: RG3GB3RG2GB2RG1GB1RG0GB0
- 16-bit read by display controller: RG0GB0, RG1GB1

19.6.1.3 RGB888 Data

This is another kind of processed data from image sensor, which can be used for further image processing directly. Each of the data consist of 8-bit Red, 8-bit Green, and 8-bit Blue data. An example of timing scheme is shown in the figure below.

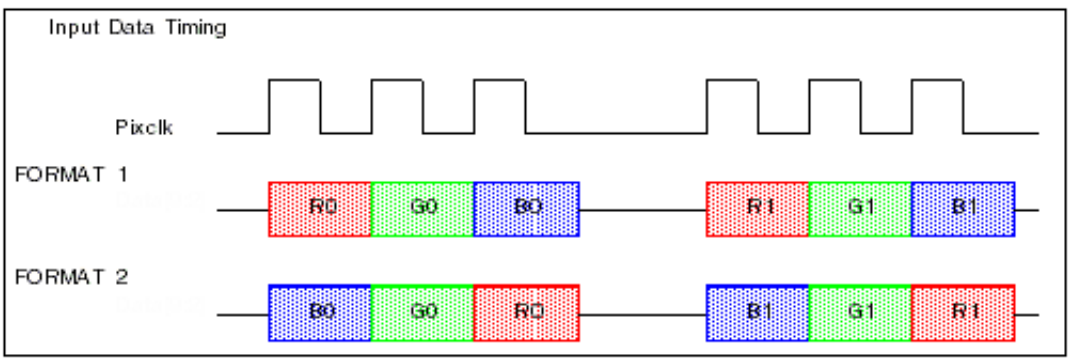


Figure 19-6. Sample Timing Diagram for RGB888 8 bits/cycle Data

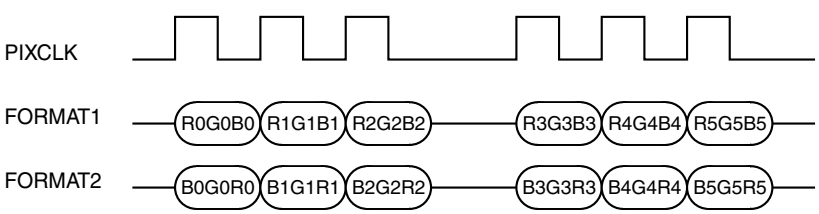
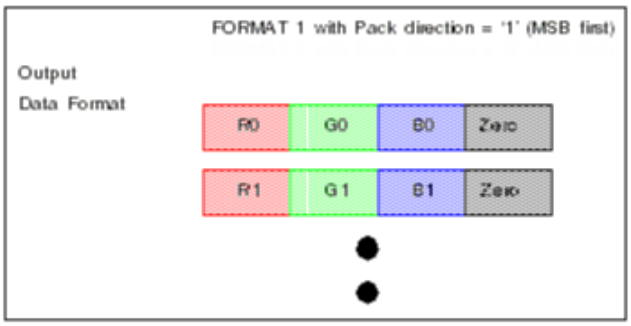


Figure 19-7. Sample Timing Diagram for RGB888 24 bits/cycle Data

An optional scheme to pack a dummy byte is provided. For every group of 3 bytes data, a dummy zero is packed to form a 32-bit word as shown in the figure below. The dummy zero can be packed at the LSB position or MSB position. Using RGB888A_FORMAT_SEL in CSI_CSICR18[18] to determine to put the dummy bytes packed at LSB or MSB position.


FORMAT 1 with Pack direction = '1' (MSB first)
Output
Data Format
R0
G0
B0
Zero
R1
G1
B1
Zero

19.6.2 STAT FIFO Path

Statistics only works for Bayer data in 8-bit per pixel format. It generates 16-bit statistical output from the 8-bit Bayer input (CSI_DATA[13:6]). The outputs are Sum of Green (G), Sum of Red (R), Sum of Blue (B), and Auto Focus (F). Each output is 16-bits wide.

The settings of PACK_DIR and SWAP16_EN bits in the CSICR1 register have no effect on the input path. The PACK_DIR only controls how the 16-bit stat output is packed into the 32-bit STAT FIFO.

When the PACK_DIR bit = 1, the stat data is packed as:

First 32-bit: RG

Second 32-bit: BF

…

When the PACK_DIR bit = 0, the stat data is packed as:

First 32-bit GR

Second 32-bit: FB

...

19.7 CSI Memory Map/Register Definition

All the 32-bit registers of the CSI module are summarized in the Memory Map below:

CSI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_4000	CSI Control Register 1 (CSI_CSICR1)	32	R/W	4000_0800h	19.7.1/751
21C_4004	CSI Control Register 2 (CSI_CSICR2)	32	R/W	0000_0000h	19.7.2/755
21C_4008	CSI Control Register 3 (CSI_CSICR3)	32	R/W	0000_0000h	19.7.3/757
21C_400C	CSI Statistic FIFO Register (CSI_CSISTATFIFO)	32	R	0000_0000h	19.7.4/759
21C_4010	CSI RX FIFO Register (CSI_CSIRFIFO)	32	R	0000_0000h	19.7.5/760
21C_4014	CSI RX Count Register (CSI_CSIRXCNT)	32	R/W	0000_9600h	19.7.6/760
21C_4018	CSI Status Register (CSI_CSISR)	32	R/W	0000_4000h	19.7.7/761

Table continues on the next page...

CSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_4020	CSI DMA Start Address Register - for STATFIFO (CSI_CSIDMASA_STATFIFO)	32	R/W	0000_0000h	19.7.8/764
21C_4024	CSI DMA Transfer Size Register - for STATFIFO (CSI_CSIDMATS_STATFIFO)	32	R/W	0000_0000h	19.7.9/764
21C_4028	CSI DMA Start Address Register - for Frame Buffer1 (CSI_CSIDMASA_FB1)	32	R/W	0000_0000h	19.7.10/ 765
21C_402C	CSI DMA Transfer Size Register - for Frame Buffer2 (CSI_CSIDMASA_FB2)	32	R/W	0000_0000h	19.7.11/ 766
21C_4030	CSI Frame Buffer Parameter Register (CSI_CSIFBUF_PARA)	32	R/W	0000_0000h	19.7.12/ 766
21C_4034	CSI Image Parameter Register (CSI_CSIIMAG_PARA)	32	R/W	0000_0000h	19.7.13/ 767
21C_4048	CSI Control Register 18 (CSI_CSICR18)	32	R/W	0002_D000h	19.7.14/ 768
21C_404C	CSI Control Register 19 (CSI_CSICR19)	32	R/W	0000_0000h	19.7.15/ 770

19.7.1 CSI Control Register 1 (CSI_CSICR1)

This register controls the sensor interface timing and interrupt generation. The interrupt enable bits in this register control the interrupt signals and the status bits. That means status bits will only function when the corresponding interrupt bits are enabled.

Address: 21C_4000h base + 0h offset = 21C_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SWAP16_EN	EXT_VSYNC	EOF_INT_EN	PrP_IF_EN	VIDEO_MODE	COF_INT_EN	SF_OR_INTEN	RF_OR_INTEN	Reserved	SFF_DMA_DONE_INTEN	STATFF_INTEN	FB2_DMA_DONE_INTEN	FB1_DMA_DONE_INTEN	RXFF_INTEN	SOF_POL	SOF_INTEN
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				HSYNC_POL	CCIR_EN	Reserved	FCC	PACK_DIR	CLR_STATFIFO	CLR_RXFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	PIXEL_BIT
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

CSI_CSICR1 field descriptions

Field	Description
31 SWAP16_EN	SWAP 16-Bit Enable. This bit enables the swapping of 16-bit data. Data is packed from 8-bit or 10-bit to 32-bit first (according to the setting of PACK_DIR) and then swapped as 16-bit words before being put into the RX FIFO. The action of the bit only affects the RX FIFO and has no affect on the STAT FIFO. **NOTE:** Example of swapping enabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x 33441122 **NOTE:** Example of swapping disabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x11223344

Table continues on the next page...

CSI_CSICR1 field descriptions (continued)

Field	Description
	0 Disable swapping 1 Enable swapping
30 EXT_VSYNC	External VSYNC Enable. This bit controls the operational VSYNC mode. **NOTE:** This only works when the CSI is in CCIR progressive mode. 0 Internal VSYNC mode 1 External VSYNC mode
29 EOF_INT_EN	End-of-Frame Interrupt Enable. This bit enables and disables the EOF interrupt. 0 EOF interrupt is disabled. 1 EOF interrupt is generated when RX count value is reached.
28 PrP_IF_EN	CSI-PrP Interface Enable. This bit controls the CSI to PrP bus. When enabled the RxFIFO is detached from the AHB bus and connected to PrP. All CPU reads or DMA accesses to the RxFIFO register are ignored. All CSI interrupts are also masked. 0 CSI to PrP bus is disabled 1 CSI to PrP bus is enabled
27 VIDEO_MODE	Video mode select. This bit controls the video mode in CCIR mode and TV decoder input. 0 Progressive mode is selected 1 Interlace mode is selected
26 COF_INT_EN	Change Of Image Field (COF) Interrupt Enable. This bit enables the COF interrupt. This bit works only in CCIR interlace mode which is when CCIR_EN = 1 and CCIR_MODE = 1. 0 COF interrupt is disabled 1 COF interrupt is enabled
25 SF_OR_INTEN	STAT FIFO Overrun Interrupt Enable. This bit enables the STATFIFO overrun interrupt. 0 STATFIFO overrun interrupt is disabled 1 STATFIFO overrun interrupt is enabled
24 RF_OR_INTEN	RxFIFO Overrun Interrupt Enable. This bit enables the RX FIFO overrun interrupt. 0 RxFIFO overrun interrupt is disabled 1 RxFIFO overrun interrupt is enabled
23 -	This field is reserved. Reserved. This bit is reserved and should read 0.
22 SFF_DMA_ DONE_INTEN	STATFIFO DMA Transfer Done Interrupt Enable. This bit enables the interrupt of STATFIFO DMA transfer done. 0 STATFIFO DMA Transfer Done interrupt disable 1 STATFIFO DMA Transfer Done interrupt enable
21 STATFF_INTEN	STATFIFO Full Interrupt Enable. This bit enables the STAT FIFO interrupt. 0 STATFIFO full interrupt disable 1 STATFIFO full interrupt enable
20 FB2_DMA_ DONE_INTEN	Frame Buffer2 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer2 DMA transfer done.

Table continues on the next page...

CSI_CSICR1 field descriptions (continued)

Field	Description
	0 Frame Buffer2 DMA Transfer Done interrupt disable 1 Frame Buffer2 DMA Transfer Done interrupt enable
19 FB1_DMA_DONE_INTEN	Frame Buffer1 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer1 DMA transfer done. 0 Frame Buffer1 DMA Transfer Done interrupt disable 1 Frame Buffer1 DMA Transfer Done interrupt enable
18 RXFF_INTEN	RxFIFO Full Interrupt Enable. This bit enables the RxFIFO full interrupt. 0 RxFIFO full interrupt disable 1 RxFIFO full interrupt enable
17 SOF_POL	SOF Interrupt Polarity. This bit controls the condition that generates an SOF interrupt. 0 SOF interrupt is generated on SOF falling edge 1 SOF interrupt is generated on SOF rising edge
16 SOF_INTEN	Start Of Frame (SOF) Interrupt Enable. This bit enables the SOF interrupt. 0 SOF interrupt disable 1 SOF interrupt enable
15–12 Reserved	This field is reserved. Reserved. This field is reserved.
11 HSYNC_POL	HSYNC Polarity Select. This bit controls the polarity of HSYNC. This bit only works in gated-clock-that is, GCLK_MODE = 1 and CCIR_EN = 0. 0 HSYNC is active low 1 HSYNC is active high
10 CCIR_EN	CCIR656 Interface Enable. This bit selects the type of interface used. When the CCIR656 timing decoder is enabled, it replaces the function of timing interface logic. 0 Traditional interface is selected. Timing interface logic is used to latch data. 1 CCIR656 interface is selected.
9 Reserved	This field is reserved. This field is reserved.
8 FCC	FIFO Clear Control. This bit determines how the RXFIFO and STATFIFO are cleared. When Synchronous FIFO clear is selected the RXFIFO and STATFIFO are cleared, and STAT block is reset, on every SOF. FIFOs and STAT block restarts immediately after reset. For information on the operation when Asynchronous FIFO clear is selected, refer to the descriptions for the CLR_RXFIFO and CLR_STATFIFO bits. 0 Asynchronous FIFO clear is selected. 1 Synchronous FIFO clear is selected.
7 PACK_DIR	Data Packing Direction. This bit Controls how 8-bit/10-bit image data is packed into 32-bit RX FIFO, and how 16-bit statistical data is packed into 32-bit STAT FIFO. 0 Pack from LSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x44332211 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xBBBBAAAA in STAT FIFO. 1 Pack from MSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x11223344 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xAAAABBBB in STAT FIFO.

Table continues on the next page...

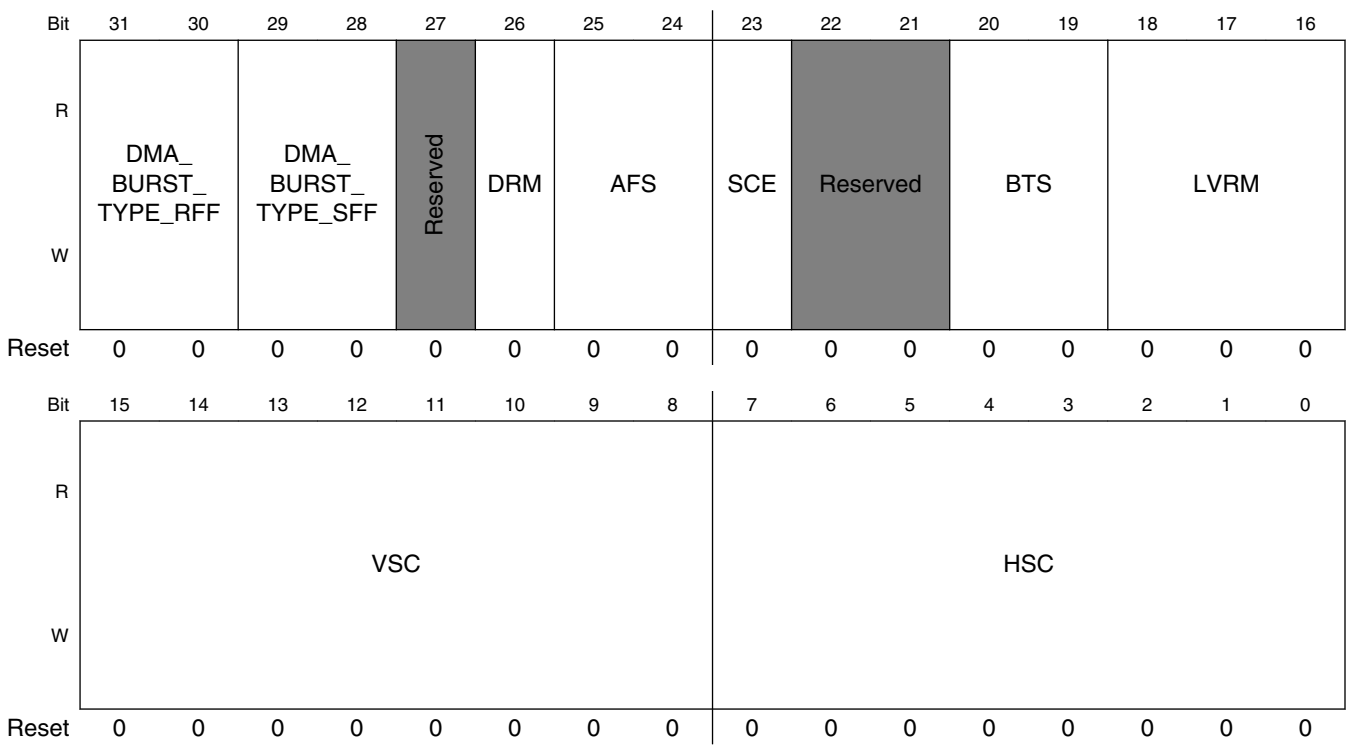
CSI_CSICR1 field descriptions (continued)

Field	Description
6 CLR_STATFIFO	Asynchronous STATFIFO Clear. This bit clears the STATFIFO and Reset STAT block. This bit works only in async FIFO clear mode-that is, FCC = 0. Otherwise this bit is ignored. Writing 1 will clear STATFIFO and reset STAT block immediately, STATFIFO and STAT block then wait and restart after the arrival of next SOF. The bit is restored to 0 automatically after finish. Normally reads 0.
5 CLR_RXFIFO	Asynchronous RXFIFO Clear. This bit clears the RXFIFO. This bit works only in async FIFO clear mode-that is, FCC = 0. Otherwise this bit is ignored. Writing 1 clears the RXFIFO immediately, RXFIFO restarts immediately after that. The bit is restored to 0 automatically after finish. Normally reads 0.
4 GCLK_MODE	Gated Clock Mode Enable. Controls if CSI is working in gated or non-gated mode. This bit works only in traditional mode-that is, CCIR_EN = 0. Otherwise this bit is ignored. 0 Non-gated clock mode. All incoming pixel clocks are valid. HSYNC is ignored. 1 Gated clock mode. Pixel clock signal is valid only when HSYNC is active.
3 INV_DATA	Invert Data Input. This bit enables or disables internal inverters on the data lines. 0 CSI_D[7:0] data lines are directly applied to internal circuitry 1 CSI_D[7:0] data lines are inverted before applied to internal circuitry
2 INV_PCLK	Invert Pixel Clock Input. This bit determines if the Pixel Clock (CSI_PIXCLK) is inverted before it is applied to the CSI module. 0 CSI_PIXCLK is directly applied to internal circuitry 1 CSI_PIXCLK is inverted before applied to internal circuitry
1 REDGE	Valid Pixel Clock Edge Select. Selects which edge of the CSI_PIXCLK is used to latch the pixel data. 0 Pixel data is latched at the falling edge of CSI_PIXCLK 1 Pixel data is latched at the rising edge of CSI_PIXCLK
0 PIXEL_BIT	Pixel Bit. This bit indicates the bayer data width for each pixel. This bit should be configured before activating or re-starting the embedded DMA controller. 0 8-bit data for each pixel 1 10-bit data for each pixel

19.7.2 CSI Control Register 2 (CSI_CSICR2)

This register provides the statistic block with data about which live view resolution is being used, and the starting sensor pixel of the Bayer pattern. It also contains the horizontal and vertical count used to determine the number of pixels to skip between the 64 x 64 blocks of statistics when generating statistics on live view image that are greater than 512 x 384.

Address: 21C_4000h base + 4h offset = 21C_4004h



CSI_CSICR2 field descriptions

Field	Description
31–30 DMA_BURST_ TYPE_RFF	Burst Type of DMA Transfer from RxFIFO. Selects the burst type of DMA transfer from RxFIFO. X0 INCR8 01 INCR4 11 INCR16
29–28 DMA_BURST_ TYPE_SFF	Burst Type of DMA Transfer from STATFIFO. Selects the burst type of DMA transfer from STATFIFO. X0 INCR8 01 INCR4 11 INCR16

Table continues on the next page...

CSI_CSICR2 field descriptions (continued)

Field	Description
27 -	This field is reserved. Reserved. These bit is reserved and should read 0.
26 DRM	Double Resolution Mode. Controls size of statistics grid. 0 Stats grid of 8 x 6 1 Stats grid of 8 x 12
25–24 AFS	Auto Focus Spread. Selects which green pixels are used for auto-focus. 00 Abs Diff on consecutive green pixels 01 Abs Diff on every third green pixels 1x Abs Diff on every four green pixels
23 SCE	Skip Count Enable. Enables or disables the skip count feature. 0 Skip count disable 1 Skip count enable
22–21 -	This field is reserved. Reserved. These bits are reserved and should read 0.
20–19 BTS	Bayer Tile Start. Controls the Bayer pattern starting point. 00 GR 01 RG 10 BG 11 GB
18–16 LVRM	Live View Resolution Mode. Selects the grid size used for live view resolution. 0 512 x 384 1 448 x 336 2 384 x 288 3 384 x 256 4 320 x 240 5 288 x 216 6 400 x 300
15–8 VSC	Vertical Skip Count. Contains the number of rows to skip. SCE must be 1, otherwise VSC is ignored. 0-255 Number of rows to skip minus 1
HSC	Horizontal Skip Count. Contains the number of pixels to skip. SCE must be 1, otherwise HSC is ignored. 0-255 Number of pixels to skip minus 1

19.7.3 CSI Control Register 3 (CSI_CSICR3)

This read/write register acts as an extension of the functionality of the CSI Control register 1, adding additional control and features.

Address: 21C_4000h base + 8h offset = 21C_4008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FRMCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FRMCNT_RST	DMA_REFLASH_RFF	DMA_REFLASH_SFF	DMA_REQ_EN_RFF	DMA_REQ_EN_SFF	STATFF_LEVEL			HRESP_ERR_EN	RxFF_LEVEL			TWO_8BIT_SENSOR	ZERO_PACK_EN	ECC_INT_EN	ECC_AUTO_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSICR3 field descriptions

Field	Description
31–16 FRMCNT	Frame Counter. This is a 16-bit Frame Counter (Wraps around automatically after reaching the maximum)
15 FRMCNT_RST	Frame Count Reset. Resets the Frame Counter. (Cleared automatically after reset is done) 0 Do not reset 1 Reset frame counter immediately
14 DMA_REFLASH_RFF	Reflash DMA Controller for RxFIFO. This bit reflash the embedded DMA controller for RxFIFO. It should be reflashed before the embedded DMA controller starts to work. (Cleared automatically after reflashing is done) 0 No reflashing 1 Reflash the embedded DMA controller
13 DMA_REFLASH_SFF	Reflash DMA Controller for STATFIFO. This bit reflash the embedded DMA controller for STATFIFO. It should be reflashed before the embedded DMA controller starts to work. (Cleared automatically after reflashing is done) 0 No reflashing 1 Reflash the embedded DMA controller

Table continues on the next page...

CSI_CSICR3 field descriptions (continued)

Field	Description
12 DMA_REQ_EN_RFF	DMA Request Enable for RxFIFO. This bit enables the dma request from RxFIFO to the embedded DMA controller. 0 Disable the dma request 1 Enable the dma request
11 DMA_REQ_EN_SFF	DMA Request Enable for STATFIFO. This bit enables the dma request from STATFIFO to the embedded DMA controller. 0 Disable the dma request 1 Enable the dma request
10–8 STATFF_LEVEL	STATFIFO Full Level. When the number of data in STATFIFO reach this level, STATFIFO full interrupt is generated, or STATFIFO DMA request is sent. 000 4 Double words 001 8 Double words 010 12 Double words 011 16 Double words 100 24 Double words 101 32 Double words 110 48 Double words 111 64 Double words
7 HRESP_ERR_EN	Hresponse Error Enable. This bit enables the hresponse error interrupt. 0 Disable hresponse error interrupt 1 Enable hresponse error interrupt
6–4 RxFF_LEVEL	**RxFIFO Full Level**. When the number of data in RxFIFO reaches this level, a RxFIFO full interrupt is generated, or an RXFIFO DMA request is sent. 000 4 Double words 001 8 Double words 010 16 Double words 011 24 Double words 100 32 Double words 101 48 Double words 110 64 Double words 111 96 Double words
3 TWO_8BIT_SENSOR	Two 8-bit Sensor Mode. This bit indicates one 16-bit sensor or two 8-bit sensors are connected to the 16-bit data ports. This bit should be set if there is one 16-bit sensor or two 8-bit sensors are connected. This bit should be configured before activating or restarting the embedded DMA controller. 0 Only one sensor is connected. 1 Two 8-bit sensors are connected or one 16-bit sensor is connected.
2 ZERO_PACK_EN	Dummy Zero Packing Enable. This bit causes a dummy zero to be packed with every 3 incoming bytes, forming a 32-bit word. The dummy zero is always packed to the LSB position. This packing function is only available in 8-bit/pixel mode. 0 Zero packing disabled 1 Zero packing enabled

Table continues on the next page...

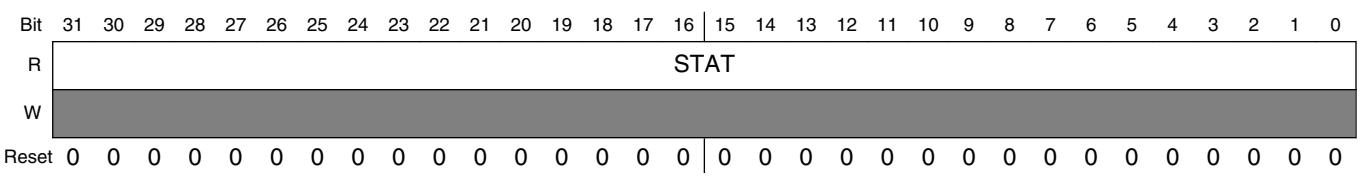
CSI_CSICR3 field descriptions (continued)

Field	Description
1 ECC_INT_EN	Error Detection Interrupt Enable. This bit enables and disables the error detection interrupt. This feature only works in CCIR interlace mode. 0 No interrupt is generated when error is detected. Only the status bit ECC_INT is set. 1 Interrupt is generated when error is detected.
0 ECC_AUTO_EN	Automatic Error Correction Enable. This bit enables and disables the automatic error correction. If an error occurs and error correction is disabled only the ECC_INT status bit is set. This feature only works in CCIR interlace mode. 0 Auto Error correction is disabled. 1 Auto Error correction is enabled.

19.7.4 CSI Statistic FIFO Register (CSI_CSISTATFIFO)

The StatFIFO is a read-only register containing statistic data from the sensor. Writing to this register has no effect.

Address: 21C_4000h base + Ch offset = 21C_400Ch



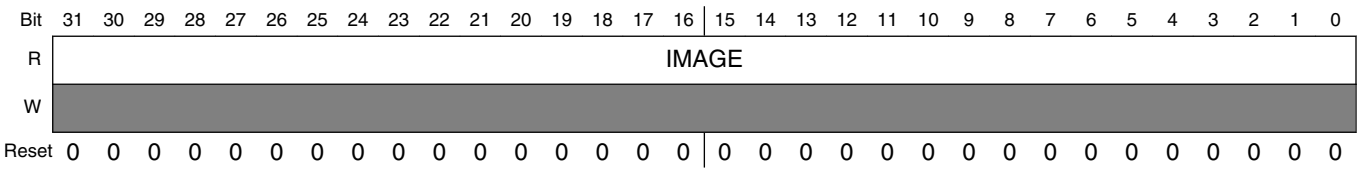
CSI_CSISTATFIFO field descriptions

Field	Description
STAT	Static data from sensor

19.7.5 CSI RX FIFO Register (CSI_CSIRFIFO)

This read-only register contains received image data. Writing to this register has no effect.

Address: 21C_4000h base + 10h offset = 21C_4010h



CSI_CSIRFIFO field descriptions

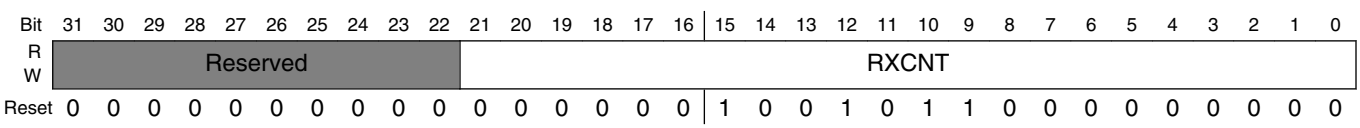
Field	Description
IMAGE	Received image data

19.7.6 CSI RX Count Register (CSI_CSIRXCNT)

This register works for EOF interrupt generation. It should be set to the number of words to receive that would generate an EOF interrupt.

There is an internal counter that counts the number of words read from the RX FIFO. Whenever the RX FIFO is being read, by either the CPU or the embedded DMA controller, the counter value is updated and compared with this register. If the values match, then an EOF interrupt is triggered.

Address: 21C_4000h base + 14h offset = 21C_4014h



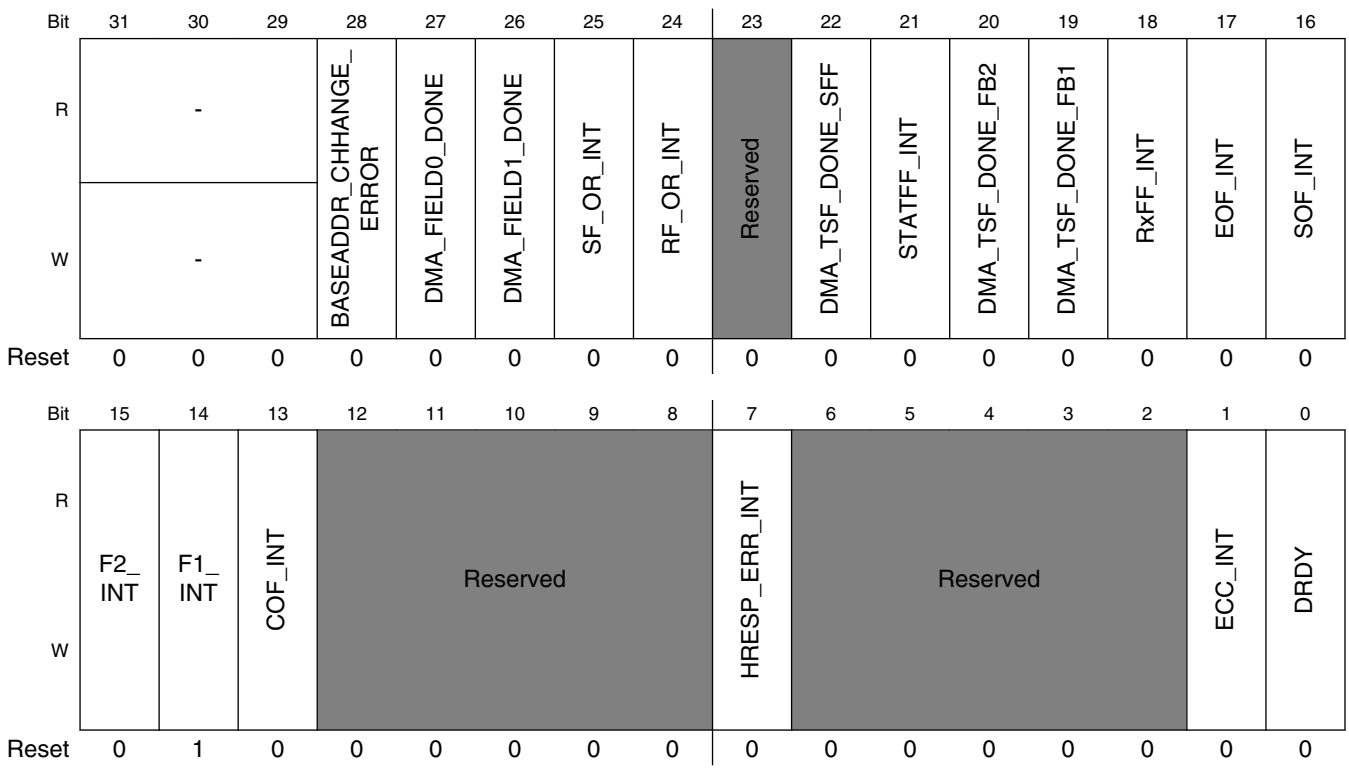
CSI_CSIRXCNT field descriptions

Field	Description
31–22 -	This field is reserved. Reserved. These bits are reserved and should read 0.
RXCNT	RxFIFO Count. This 22-bit counter for RXFIFO is updated each time the RXFIFO is read by CPU or DMA.This counter should be set to the expected number of words to receive that would generate an EOF interrupt.

19.7.7 CSI Status Register (CSI_CSISR)

This read/write register shows sensor interface status, and which kind of interrupt is being generated. The corresponding interrupt bits must be set for the status bit to function. Status bits should function normally even if the corresponding interrupt enable bits are not enabled.

Address: 21C_4000h base + 18h offset = 21C_4018h



CSI_CSISR field descriptions

Field	Description
31–29 -	Reserved. These bits are reserved and should read 0.
28 BASEADDR_ CHHANGE_ ERROR	When using base address switching enable, this bit will be 1 when switching occur before DMA complete. This bit will be clear by writing 1. When this interrupt happens, follow the steps listed below. 1. Unassert the CSI enable, CSIx_CSICR18 bit31, 2. Reflash the DMA, assert the CSIX_CSICR3 bit 14, 3. Assert the CSI enable, CSIx_CSICR18 bit31.
27 DMA_FIELD0_ DONE	When DMA field 0 is complete, this bit will be set to 1(clear by writing 1).

Table continues on the next page...

CSI_CSISR field descriptions (continued)

Field	Description
26 DMA_FIELD1_DONE	When DMA field 0 is complete, this bit will be set to 1(clear by writing 1).
25 SF_OR_INT	STATFIFO Overrun Interrupt Status. Indicates the overflow status of the STATFIFO register. (Cleared by writing 1) 0 STATFIFO has not overflowed. 1 STATFIFO has overflowed.
24 RF_OR_INT	RxFIFO Overrun Interrupt Status. Indicates the overflow status of the RxFIFO register. (Cleared by writing 1) 0 RXFIFO has not overflowed. 1 RXFIFO has overflowed.
23 -	This field is reserved. Reserved. This bit is reserved and should read 0.
22 DMA_TSF_DONE_SFF	DMA Transfer Done from StatFIFO. Indicates that the dma transfer from StatFIFO is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by writting 1 or reflashing the StatFIFO dma controller in CSICR3.(Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
21 STATFF_INT	STATFIFO Full Interrupt Status. Indicates the number of data in the STATFIFO reaches the trigger level. (this bit is cleared automatically by reading the STATFIFO) 0 STATFIFO is not full. 1 STATFIFO is full.
20 DMA_TSF_DONE_FB2	DMA Transfer Done in Frame Buffer2. Indicates that the DMA transfer from RxFIFO to Frame Buffer2 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by by writting 1 or reflashing the RxFIFO dma controller in CSICR3. (Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
19 DMA_TSF_DONE_FB1	DMA Transfer Done in Frame Buffer1. Indicates that the DMA transfer from RxFIFO to Frame Buffer1 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by by writting 1 or reflashing the RxFIFO dma controller in CSICR3. (Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
18 RxFF_INT	RXFIFO Full Interrupt Status. Indicates the number of data in the RxFIFO reaches the trigger level. (this bit is cleared automatically by reading the RxFIFO) 0 RxFIFO is not full. 1 RxFIFO is full.
17 EOF_INT	End of Frame (EOF) Interrupt Status. Indicates when EOF is detected. (Cleared by writing 1) 0 EOF is not detected. 1 EOF is detected.
16 SOF_INT	Start of Frame Interrupt Status. Indicates when SOF is detected. (Cleared by writing 1)

Table continues on the next page...

CSI_CSISR field descriptions (continued)

Field	Description
	0 SOF is not detected. 1 SOF is detected.
15 F2_INT	CCIR Field 2 Interrupt Status. Indicates the presence of field 2 of video in CCIR mode. (Cleared automatically when current field does not match) **NOTE:** Only works in CCIR Interlace mode. 0 Field 2 of video is not detected 1 Field 2 of video is about to start
14 F1_INT	CCIR Field 1 Interrupt Status. Indicates the presence of field 1 of video in CCIR mode. (Cleared automatically when current field does not match) **NOTE:** Only works in CCIR Interlace mode. 0 Field 1 of video is not detected. 1 Field 1 of video is about to start.
13 COF_INT	Change Of Field Interrupt Status. Indicates that a change of the video field has been detected. Only works in CCIR Interlace mode. Software should read this bit first and then dispatch the new field from F1_INT and F2_INT. (Cleared by writing 1) 0 Video field has no change. 1 Change of video field is detected.
12–8 -	This field is reserved. Reserved. These bits are reserved and should read 0.
7 HRESP_ERR_INT	Hresponse Error Interrupt Status. Indicates that a hresponse error has been detected. (Cleared by writing 1) 0 No hresponse error. 1 Hresponse error is detected.
6–2 -	This field is reserved. Reserved. These bits are reserved and should read 0.
1 ECC_INT	CCIR Error Interrupt. This bit indicates an error has occurred. This only works in CCIR Interlace mode. (Cleared by writing 1) 0 No error detected 1 Error is detected in CCIR coding
0 DRDY	RXFIFO Data Ready. Indicates the presence of data that is ready for transfer in the RxFIFO. (Cleared automatically by reading FIFO) 0 No data (word) is ready 1 At least 1 datum (word) is ready in RXFIFO.

19.7.8 CSI DMA Start Address Register - for STATFIFO (CSI_CSIDMASA_STATFIFO)

This register provides the start address for the embedded DMA controller of STATFIFO. The embedded DMA controller will read data from STATFIFO and write it to the external memory from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 21C_4000h base + 20h offset = 21C_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_SFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

CSI_CSIDMASA_STATFIFO field descriptions

Field	Description
31–2 DMA_START_ ADDR_SFF	DMA Start Address for STATFIFO. Indicates the start address to write data. The embedded DMA controller will read data from STATFIFO and write it from this address through AHB bus. The address should be double words aligned.
-	This field is reserved. Reserved. These bits are reserved and should read 0.

19.7.9 CSI DMA Transfer Size Register - for STATFIFO (CSI_CSIDMATS_STATFIFO)

This register provides the total transfer size for the embedded DMA controller of STATFIFO. This register should be configured before activating or restarting the embedded DMA controller.

Address: 21C_4000h base + 24h offset = 21C_4024h

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

CSI_CSIDMATS_STATFIFO field descriptions

Field	Description
DMA_TSF_ SIZE_SFF	DMA Transfer Size for STATFIFO. Indicates how many words to be transfered by the embedded DMA controller. The size should be double words aligned.

19.7.10 CSI DMA Start Address Register - for Frame Buffer1 (CSI_CSIDMASA_FB1)

This register provides the start address in the frame buffer1 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer1 from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 21C_4000h base + 28h offset = 21C_4028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_FB1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_FB1														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIDMASA_FB1 field descriptions

Field	Description
31–2 DMA_START_ ADDR_FB1	DMA Start Address in Frame Buffer1. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. The address should be double words aligned.
-	This field is reserved. Reserved. These bits are reserved and should read 0.

19.7.11 CSI DMA Transfer Size Register - for Frame Buffer2 (CSI_CSIDMASA_FB2)

This register provides the start address in the frame buffer2 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer2 from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 21C_4000h base + 2Ch offset = 21C_402Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_FB2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_FB2														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIDMASA_FB2 field descriptions

Field	Description
31–2 DMA_START_ADDR_FB2	DMA Start Address in Frame Buffer2. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. The address should be double words aligned.
-	This field is reserved. Reserved. These bits are reserved and should read 0.

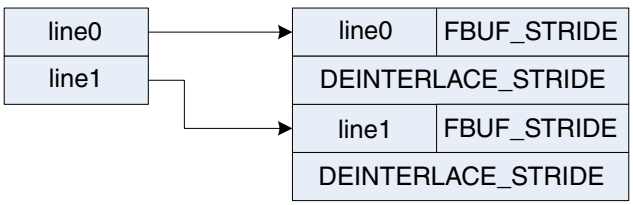
19.7.12 CSI Frame Buffer Parameter Register (CSI_CSIFBUF_PARA)

This register provides the stride of the frame buffer to show how many words to skip before starting to write the next row of the image. The width of the frame buffer minus the width of the image is the stride. This register should be configured before activating or restarting the embedded DMA controller.

Address: 21C_4000h base + 30h offset = 21C_4030h

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

CSI_CSIFBUF_PARA field descriptions

Field	Description
31–16 DEINTERLACE_ STRIDE	DEINTERLACE_STRIDE is only used in the deinterlace mode. If line stride feature is supported in deinterlace mode, FBUF_STRIDE and DEINTERLACE_STRIDE need to be configured at the same time. DEINTERLACE_STRIDE is configured the same as line width. In normal line stride feature, only FBUF_STRIDE needs to be configured. 
FBUF_STRIDE	Frame Buffer Parameter. Indicates the stride of the frame buffer. The width of the frame buffer(in double words) minus the width of the image(in double words) is the stride. The stride should be double words aligned. The embedded DMA controller will skip the stride before starting to write the next row of the image.

19.7.13 CSI Image Parameter Register (CSI_CSIIMAG_PARA)

This register provides the width and the height of the image from the sensor. The width and height should be aligned in pixel. The width of the image multiplied by the height is the total pixel size that will be transfered in a frame by the embedded DMA controller. This register should be configured before activating or restarting the embedded DMA controller.

Address: 21C_4000h base + 34h offset = 21C_4034h

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

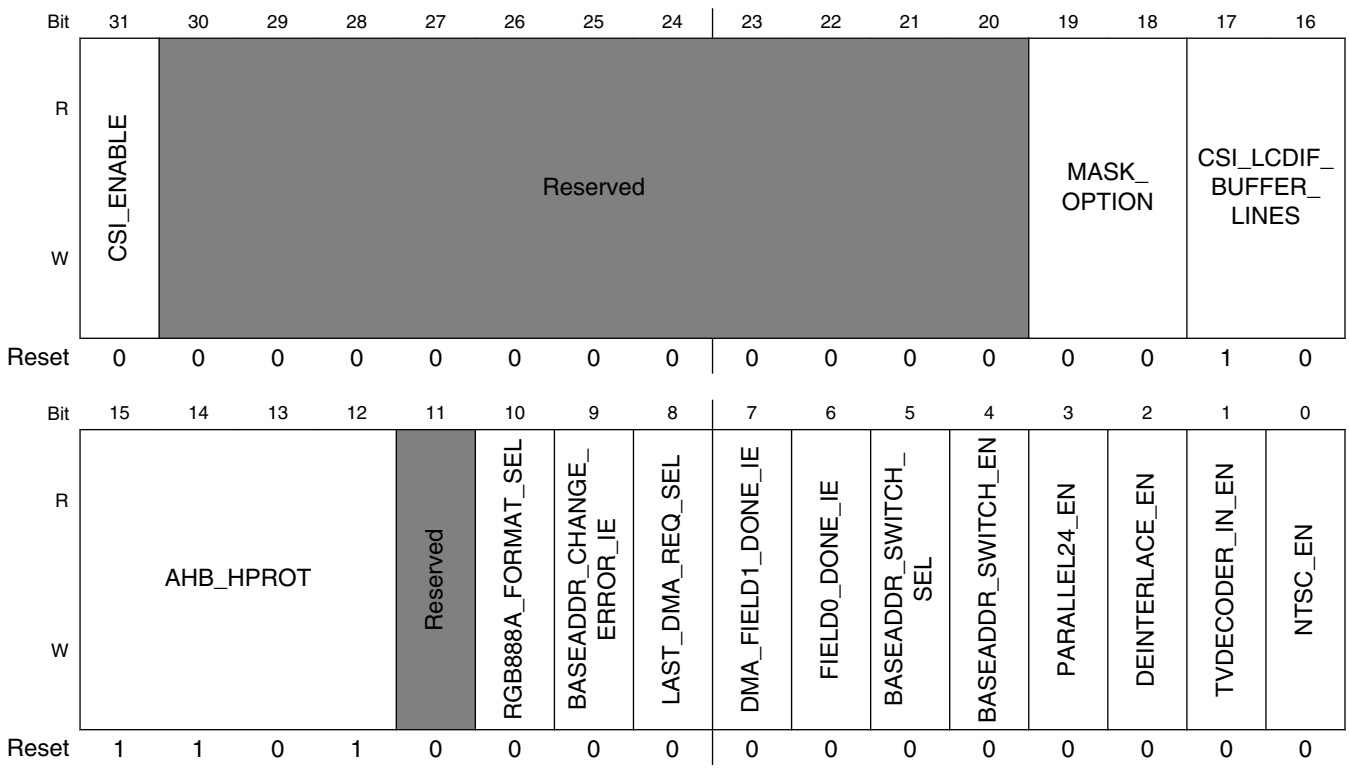
CSI_CSIIMAG_PARA field descriptions

Field	Description
31–16 IMAGE_WIDTH	Image Width. Indicates how many pixels in a line of the image from the sensor. If the input data from the sensor is 8-bit/pixel format, the IMAGE_WIDTH should be a multiple of 8 pixels. If the input data from the sensor is 10-bit/pixel or 16-bit/pixel format, the IMAGE_WIDTH should be a multiple of 2 pixels.
IMAGE_HEIGHT	Image Height. Indicates how many pixels in a column of the image from the sensor.

19.7.14 CSI Control Register 18 (CSI_CSICR18)

This read/write register acts as an extension of the functionality of the CSI Control register 1

Address: 21C_4000h base + 48h offset = 21C_4048h



CSI_CSICR18 field descriptions

Field	Description
31 CSI_ENABLE	CSI global enable signal. Only when this bit is 1, CSI can start to receive the data and store to memory.
30–20 -	This field is reserved.
19–18 MASK_OPTION	These bits used to choose the method to mask the CSI input. 00 Writing to memory from first completely frame, when using this option, the CSI_ENABLE should be 1. 01 Writing to memory when CSI_ENABLE is 1. 02 Writing to memory from second completely frame, when using this option, the CSI_ENABLE should be 1. 03 Writing to memory when data comes in, not matter the CSI_ENABLE is 1 or 0.

Table continues on the next page...

CSI_CSICR18 field descriptions (continued)

Field	Description
17–16 CSI_LCDIF_ BUFFER_LINES	The number of lines are used in handshake mode with LCDIF. 00 4 lines 01 8 lines 02 16 lines 03 16 lines
15–12 AHB_HPROT	Hprot value in AHB bus protocol.
11 -	This field is reserved.
10 RGB888A_ FORMAT_SEL	Output is 32-bit format. 0 {8'h0, data[23:0]} 1 {data[23:0], 8'h0}
9 BASEADDR_ CHANGE_ ERROR_IE	Base address change error interrupt enable signal.
8 LAST_DMA_ REQ_SEL	Choosing the last DMA request condition. 0 fifo_full_level 1 hburst_length
7 DMA_FIELD1_ DONE_IE	When in interlace mode, field 1 done interrupt enable. 0 Interrupt disabled 1 Interrupt enabled
6 FIELD0_DONE_ IE	In interlace mode, fileld 0 means interrupt enabled. 0 Interrupt disabled 1 Interrupt enabled
5 BASEADDR_ SWITCH_SEL	CSI 2 base addresses switching method. When using this bit, BASEADDR_SWITCH_EN is 1. 0 Switching base address at the edge of the vsync 1 Switching base address at the edge of the first data of each frame
4 BASEADDR_ SWITCH_EN	When this bit is enabled, CSI DMA will switch the base address according to BASEADDR_SWITCH_SEL rather than atomically by DMA completed.
3 PARALLEL24_ EN	When input is parallel rgb888/yuv444 24bit, this bit can be enabled.
2 DEINTERLACE_ EN	This bit is used to select the output method When input is TVDECODER or standard CCIR656 video. 0 Deinterlace disabled 1 Deinterlace enabled
1 TVDECODER_ IN_EN	When input is from TV decoder, this bit is enabled.
0 NTSC_EN	This bit is used to select NTSC/PAL mode When input is TVDECODER or standard CCIR656 video.

Table continues on the next page...

CSI_CSICR18 field descriptions (continued)

Field	Description
	0 PAL 1 NTSC

19.7.15 CSI Control Register 19 (CSI_CSICR19)

This read/write register acts as an extension of the functionality of the CSI Control register 1

Address: 21C_4000h base + 4Ch offset = 21C_404Ch

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

CSI_CSICR19 field descriptions

Field	Description
31–8 -	This field is reserved.
DMA_RFIFO_ HIGHEST_FIFO_ LEVEL	This byte stores the highest FIFO level achieved by CSI FIFO timely and will be clear by writing 8'ff to it.

Chapter 20
Enhanced Configurable SPI (ECSPI)

20.1 Overview

The Enhanced Configurable Serial Peripheral Interface (ECSPI) is a full-duplex, synchronous, four-wire serial communication block.

The ECSPI contains a 64 x 32 receive buffer (RXFIFO) and a 64 x 32 transmit buffer (TXFIFO). With data FIFOs, the ECSPI allows rapid data communication with fewer software interrupts. The figure below shows a block diagram of the ECSPI.

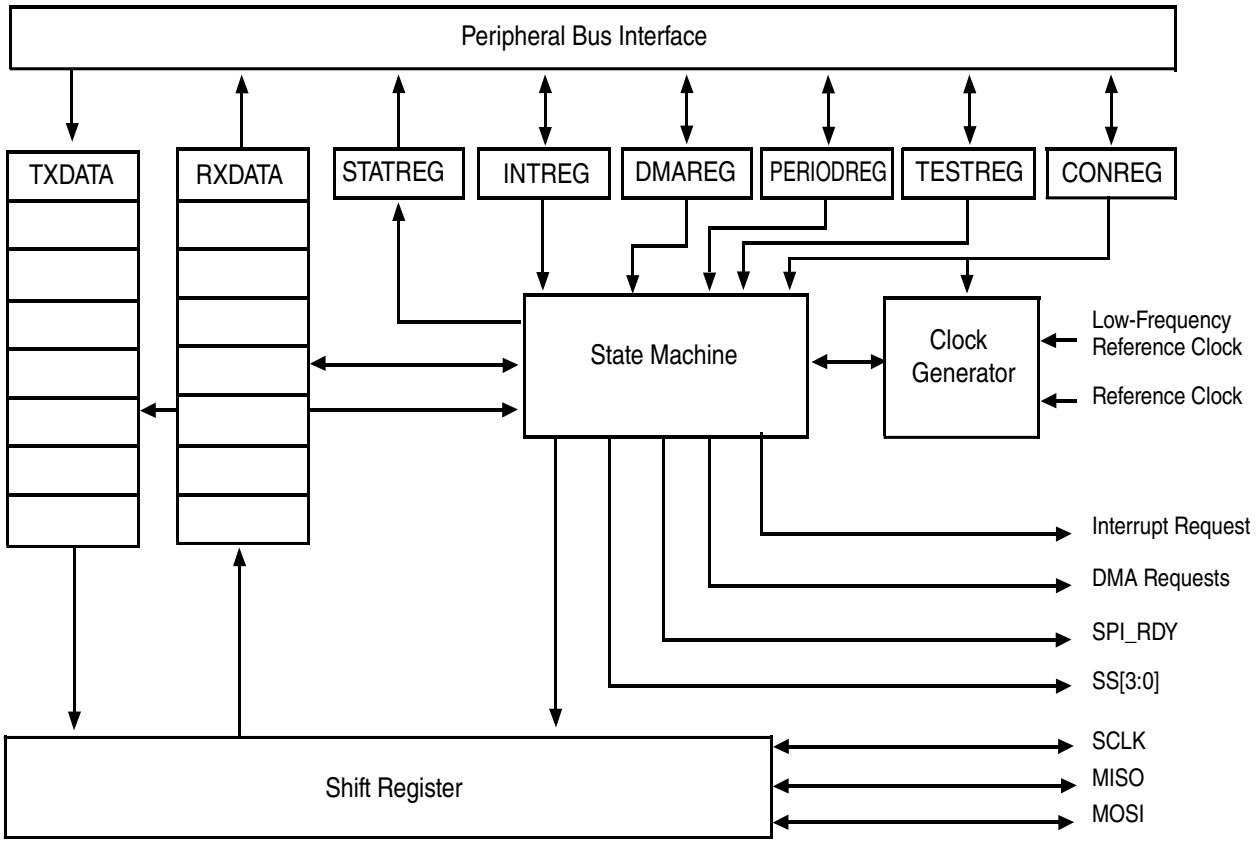


Figure 20-1. ECSPI Block Diagram

20.1.1 Features

Key features of the ECSPI include:

- Full-duplex synchronous serial interface
- Master/Slave configurable
- Four Chip Select (SS) signals to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 64-entry FIFO for both transmit and receive data
- Polarity and phase of the Chip Select (SS) and SPI Clock (SCLK) are configurable
- Direct Memory Access (DMA) support
- Max operation frequency up to the reference clock frequency.

20.1.2 Modes and Operations

The ECSPI supports the modes described in the indicated sections:

- Master Mode
- Slave Mode
- Low Power Modes

As described in Operations, the ECSPI supports the operations described in the indicated sections:

- Typical Master Mode
  - Master Mode with SPI_RDY
  - Master Mode with Wait States
  - Master Mode with SS_CTL[3:0] Control
  - Master Mode with Phase Control
- Typical Slave Mode

20.2 External Signals

Figure 20-2 shows the ECSPI in master mode connected to four external devices in a one-way communication link.

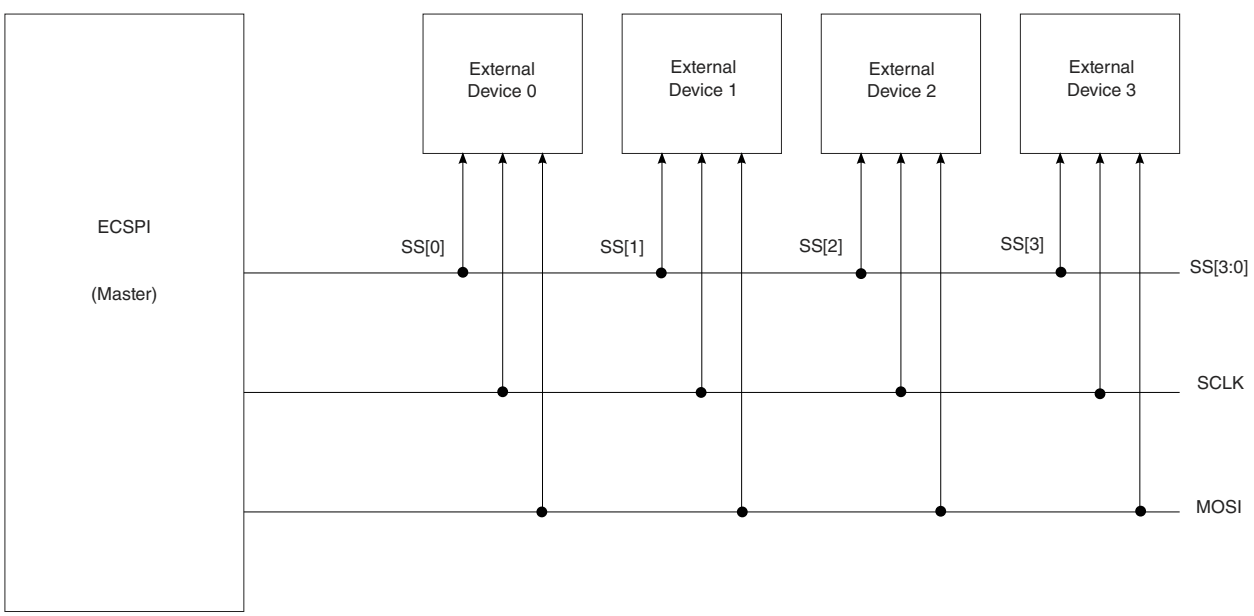


Figure 20-2. Example Connection Diagram

20.3 Clocks

The following table describes the clock sources for eCSPI. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 20-1. eCSPI Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32kHz)
ipg_clk_per	ecspi_clk_root	eCSPI module clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

20.4 Functional Description

This section provides a complete functional description of the ECSPI. The figure found here shows the relationship of SCLK and data lines while ECSPI has been configured with different POL and PHA settings.

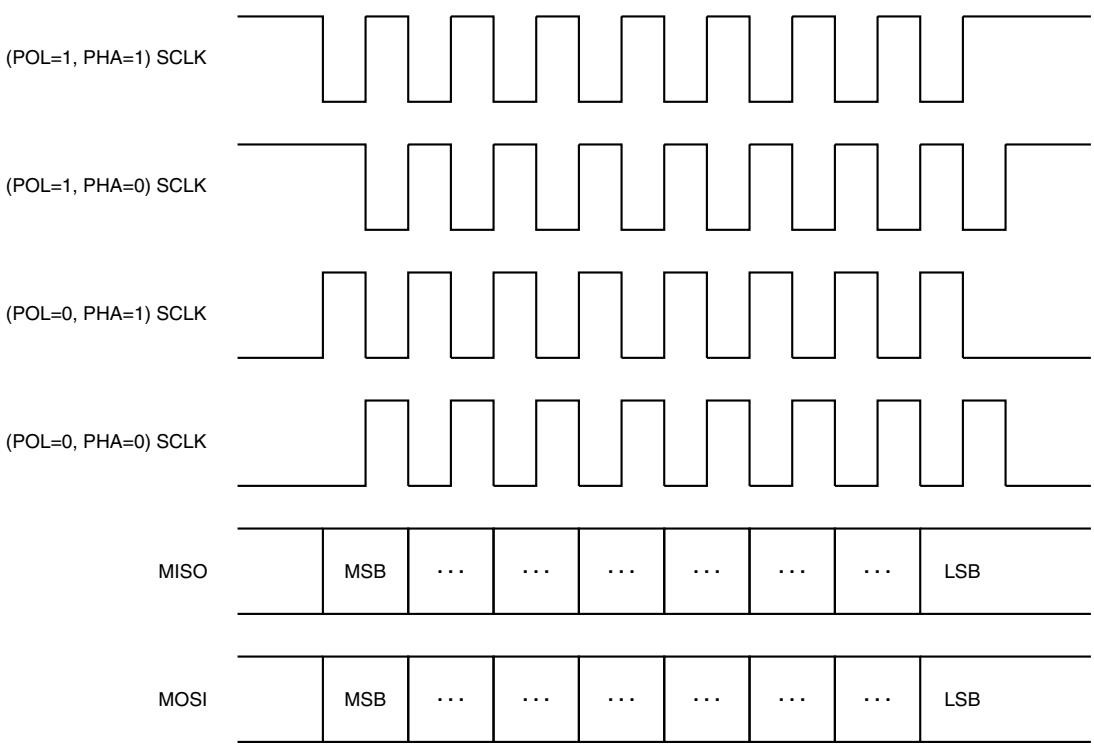


Figure 20-3. ECSPI SCLK, MISO, and MOSI Relationship

20.4.1 Master Mode

When the ECSPI is configured as a master, it uses a serial link to transfer data between the ECSPI and an external slave device.

One of the Chip Select (SS) signals and the clock signal (SCLK) are used to transfer data between two devices. If the external device is a transmit-only device, the ECSPI master's output port can be ignored and used for other purposes. In order to use the internal TXFIFO and RXFIFO, two auxiliary output signals, Chip Select (SS) and SPI_RDY, are used for data transfer rate control. Software can also configure the sample period control register to a fixed data transfer rate.

20.4.2 Slave Mode

When the ECSPI is configured as a slave, software can configure the ECSPI Control register to match the external SPI master's timing.

In this configuration, Chip Select ($\overline{SS}$) becomes an input signal, and is used to control data transfers through the Shift register, as well as to load/store the data FIFO.

Slave mode only supports the case when ECSPIx_CONFIGREG[SS_CTL] is cleared. The accurate burst length should always be specified using the BURST_LENGTH parameter. ECSPIx_CONFIGREG[SS_CTL] set to 1 is not supported in slave mode.

20.4.3 Low Power Modes

The ECSPI does not operate under low power mode.

It holds its operation when its clock is gated off in master mode. In slave mode, the ECSPI does not respond when its clock is gated off.

20.4.4 Operations

The information found here describes the ECSPI's operations.

20.4.4.1 Typical Master Mode

The ECSPI master uses the Chip Select (SS) signal to enable an external SPI device, and uses the SCLK signal to transfer data in and out of the Shift register.

The SPI_RDY enables fast data communication with fewer software interrupts. By programming the ECSPI_PERIODREG register accordingly, the ECSPI can be used for a fixed data transfer rate.

When the ECSPI is in Master mode the SS, SCLK, and MOSI are output signals, and the MISO signal is an input.

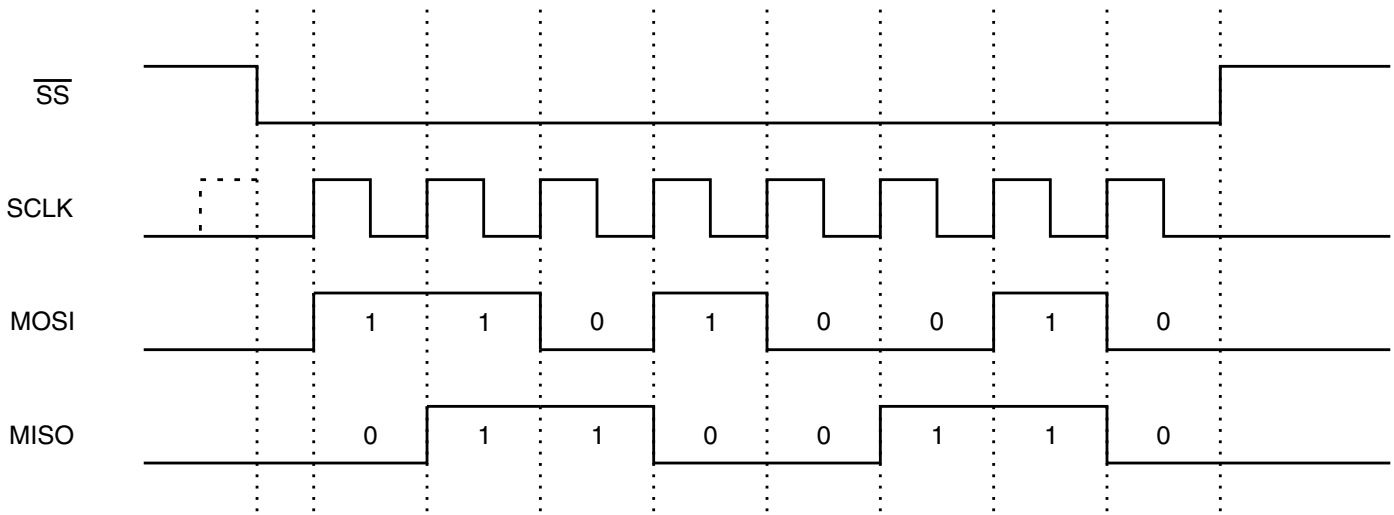


Figure 20-4. Typical SPI Burst (8-bit Transfer)

In the above figure, the Chip Select (SS) signal enables the selected external SPI device, and the SCLK synchronizes the data transfer. The MOSI and MISO signals change on rising edge of SCLK and the MISO signal is latched on the falling edge of the SCLK. The figure above shows a data of 0xD2 is shifted out, and a data of 0x66 is shifted in.

20.4.4.1.1 Master Mode with SPI_RDY

By default, the ECSPI does not use the SPI_RDY signal in master mode (MODE =1).

A SPI burst begins when the following events happen:

- The ECSPI is enabled, TXFIFO has data in it, and ECSPI_CONREG[XCH] bit or the ECSPI_CONREG[SMC] bit is set.
- When the SPI Data Ready Control (ECSPI_CONREG[DRCTL]) bits contains either 01 or 10, the SPI_RDY signal controls when a SPI burst starts.

A SPI burst is defined as a bus transaction that starts when the slave select is asserted and ends when the slave select is negated. The Chip Select (SS) signal will remain asserted until all the bits in a SPI burst are shifted out.

If ECSPI_CONREG[DRCTL] is set to 01, the SPI burst can be triggered only if a falling edge of the SPI_RDY signal has been detected.

The following figure shows the relationship between a SPI burst and the falling edge of SPI_RDY signal.

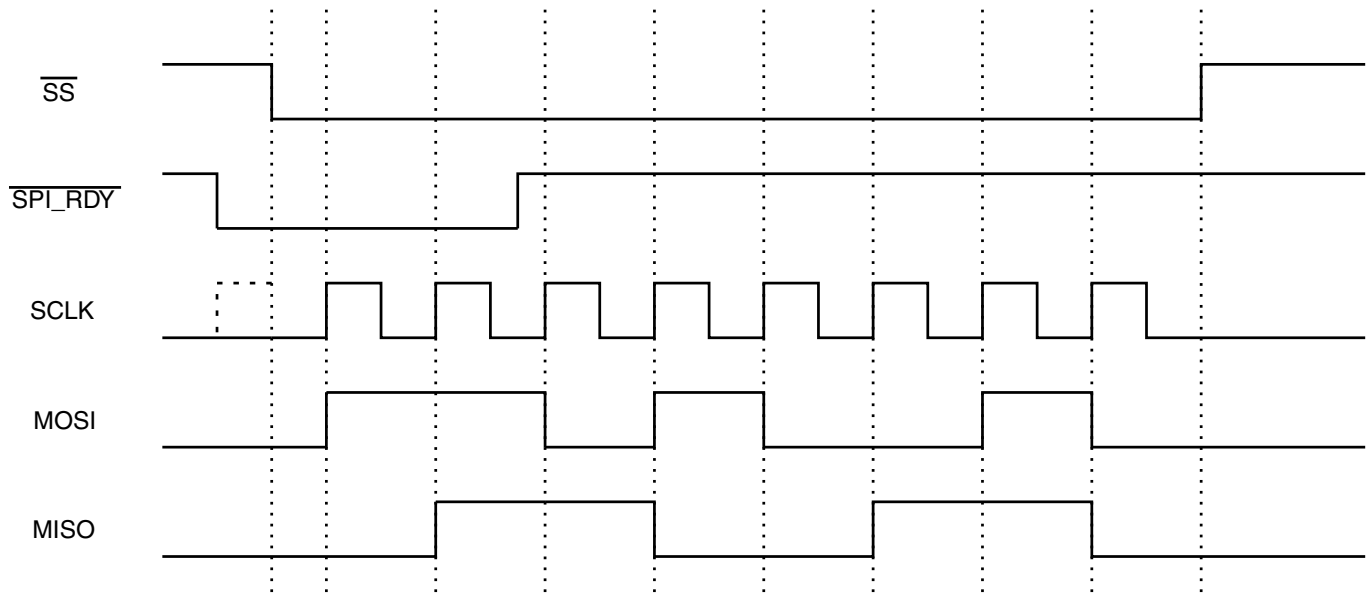


Figure 20-5. Relationship Between a SPI Burst and SPI_RDY: Falling-Edge Triggered

A SPI burst does not start until the falling edge of the SPI_RDY signal is detected. The next SPI burst starts when the next SPI_RDY falling edge is detected, after the last burst has finished.

If SPI Data Ready Control (ECSPI_CONREG[DRCTL]) is set to 10, the SPI burst can be triggered only if the SPI_RDY signal is low.

The following figure shows the relationship between a SPI burst and the SPI_RDY signal. The SPI burst does not begin until the SPI_RDY signal goes low. The ECSPI will keep transmitting SPI burst if the SPI_RDY signal remains low.

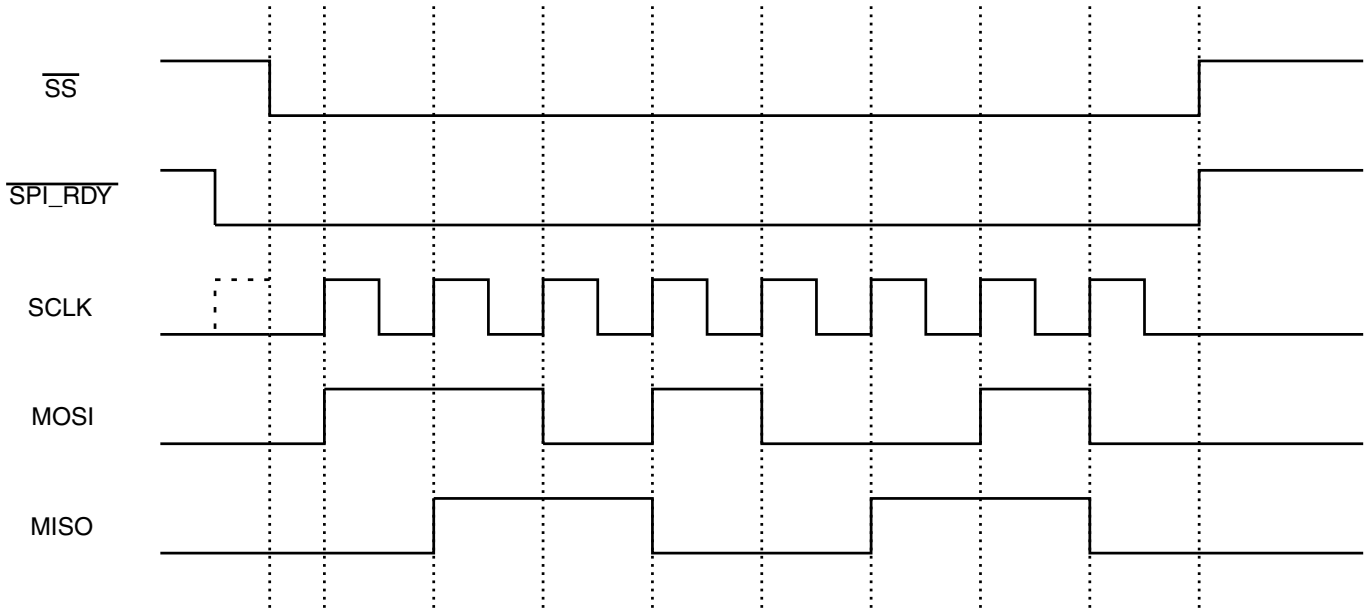


Figure 20-6. Relationship Between a SPI Burst and SPI_RDY: Low-Level Triggered

20.4.4.1.2 Master Mode with Wait States

Wait states can be inserted between SPI bursts. This provides a way for software to slow down the SPI burst to meet the timing requirements of a slower SPI device.

The following figure shows wait states inserted between SPI bursts.

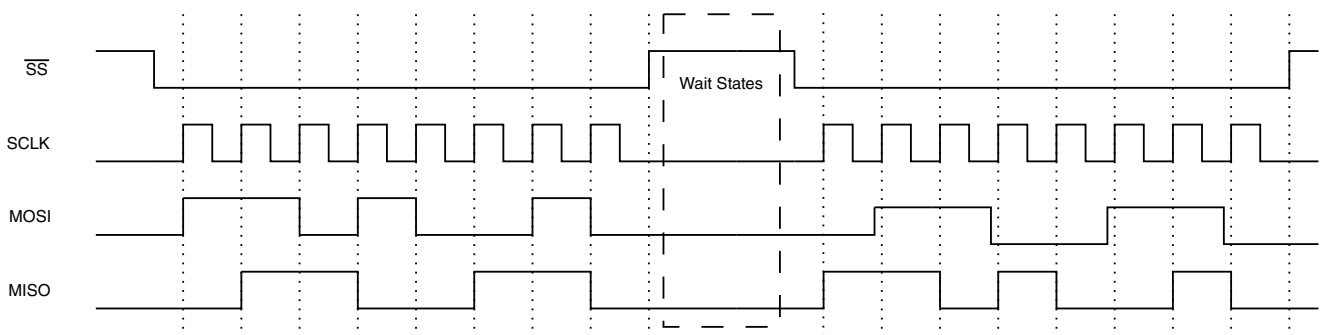


Figure 20-7. SPI Bursts with Wait States

In this case, the number of wait states is controlled by ECSPI_PERIODREG[SAMPLE PERIOD] and the wait states' clock source is selected by ECSPI_PERIODREG[CSRC].

20.4.4.1.3 Master Mode with SS_CTL[3:0] Control

The SPI SS Control (SS_CTL[3:0]) controls whether the current operation is single burst or multiple bursts.

When the SPI SS Wave Form Select (SS_CTL[3:0]) is set, the current operation is multiple bursts transfer. When the SPI SS Wave Form Select (SS_CTL[3:0]) bit is cleared, the current operation is single burst transfer. A SPI burst can contains multiple words as defined in the BURST LENGTH field of the ECSPI_CONREG register.

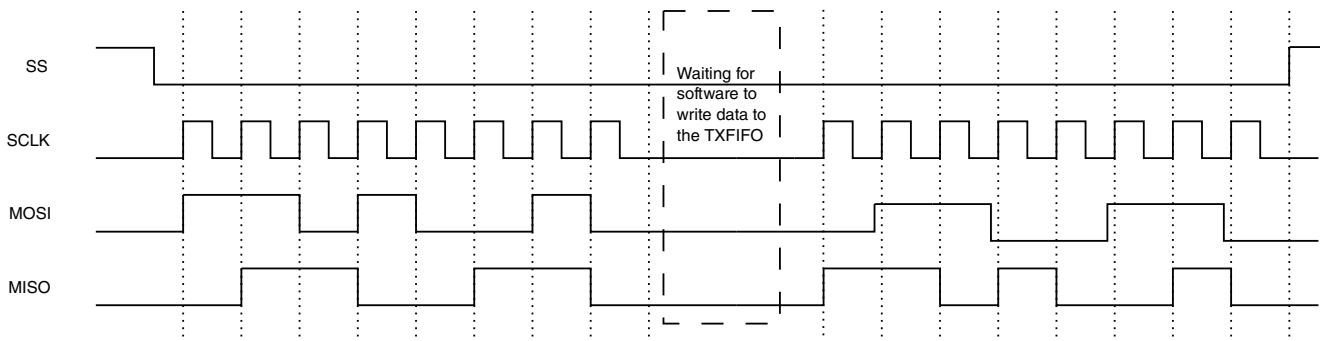


Figure 20-8. SPI Burst While SS_CTL[3:0] is Clear

In Figure 20-8, two 8-bit bursts in the TXFIFO have been combined and transmitted in one SPI burst. The maximum length of a single SPI burst is defined in the BURST LENGTH field of the ECSPI_CONREG control register. (Figure 20-8 corresponds to a BURST LENGTH of 8.) This provides a way for transferring a longer SPI burst by writing data into TXFIFO while the ECSPI is transmitting.

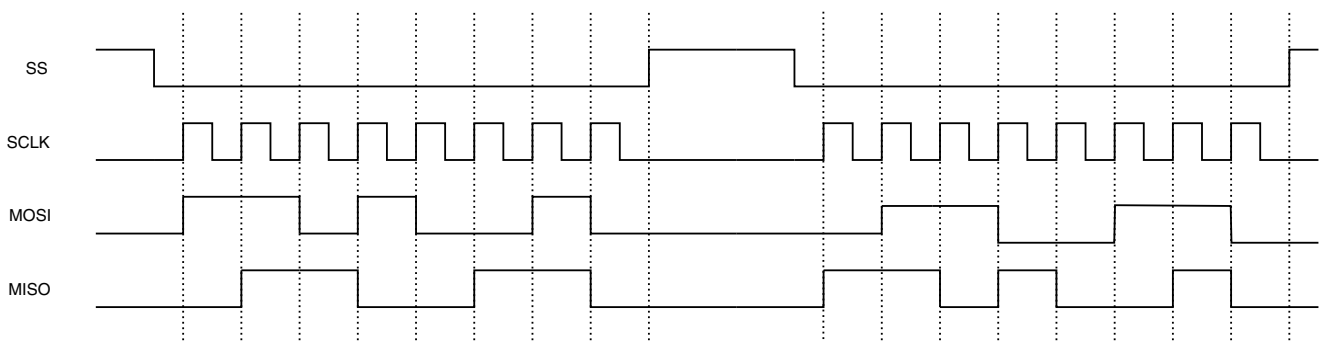


Figure 20-9. SPI Bursts While SS_CTL[3:0] is Set

In Figure 20-9, two FIFO entries are transmitted, one entry with each SPI burst. The ECSPI will continue to transmit SPI bursts until the TXFIFO is empty. When wait states can be inserted between SPI bursts, the SS will negate between SPI bursts until the wait states finish.

20.4.4.1.4 Master Mode with Phase Control

The Phase Control (ECSPI_CONREG[PHA]) bit controls how the transmit data shifts out and the receive data shifts in.

When the Phase control (ECSPI_CONREG[PHA]) bit is set, the transmit data will shift out on the rising edge of SCLK, and the receive data is latched on the falling edge of SCLK. The most-significant bit is output on the first rising SCLK edge.

When ECSPI_CONREG[PHA] is cleared, the transmit data is shifted out on the falling edge of SCLK and the receive data is latched on the rising edge of SCLK. The MSB is output when the host processor loads the transmitted data.

Inverting the SCLK polarity does not impact the edge-triggered operations because they are internal to the serial peripheral interface master. Figure 20-10 shows how SPI burst works with different POL and PHA configuration.

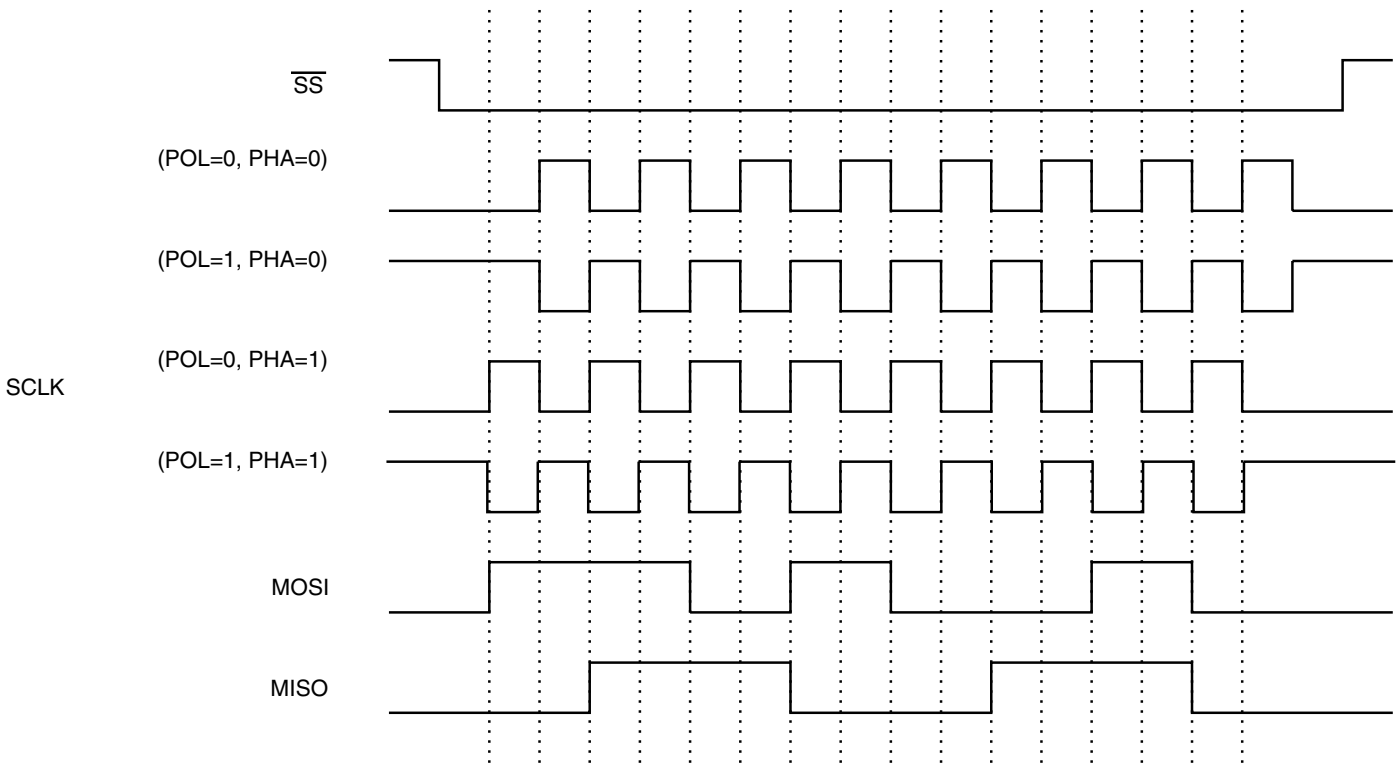


Figure 20-10. SPI Burst with Different POL and PHA Configurations

20.4.4.2 Typical Slave Mode

When the ECSPI is configured as a slave (Mode = 0), software can configure the ECSPI Control register to match the external SPI master's timing. In this configuration, SS becomes an input signal, and is used to latch data in and out of the internal data Shift registers, as well as to advance the data FIFO.

The SS, SCLK, and MOSI are inputs and MISO is output. Most of the timing diagrams are similar to the diagrams shown previously for the SPI in Master mode (Mode = 1), because the inputs come from a SPI master device.

However, the timing is different when SS is used to advance the data FIFO. When the SS_POL=0 while the ECSPI is configured in Slave mode, the data FIFO will advance on the rising edge of the SS signal. When the polarity is reversed (SS_POL = 1), the data FIFO will advance on the falling edge of the SS signal.

The figure below shows a SPI burst in which the data FIFO is advanced by the rising edge of the SS signal.

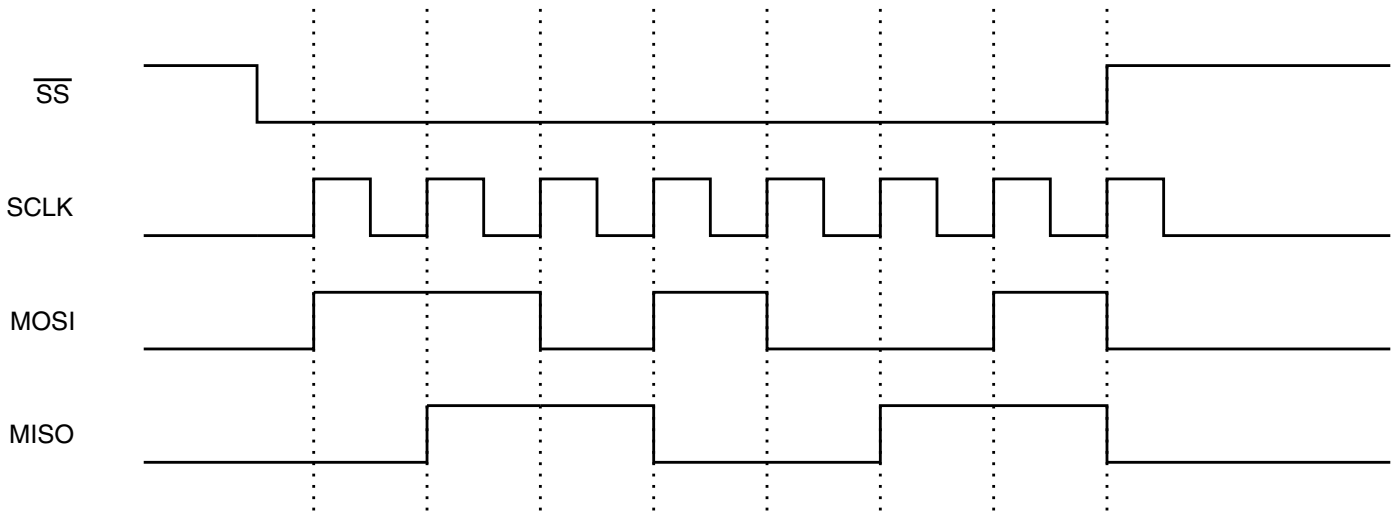


Figure 20-11. Advancing the Data FIFO on the Rising Edge of SS

In the above case, only the most significant 7 bits are loaded to the RXFIFO.

20.4.5 Reset

Whenever a device reset occurs, a reset is performed on the ECSPI, resetting all registers to their default values.

Software can reset the block using the CONREG[EN] bit; see ECSPI.

20.4.6 Interrupts

Interrupt control provides a way to manage the ECSPI FIFOs:

- For transmitting data, software can enable the TXFIFO empty, TXFIFO data request, and TXFIFO full interrupts to maintain the TXFIFO using an interrupt service routine.
- For receiving data, software can enable the RXFIFO ready, RXFIFO data request, and RXFIFO full interrupts to retrieve data from the RXFIFO using an interrupt service routine.

Other interrupt sources can be used to control or debug the SPI bursts:

- The transfer-completed interrupt means that there is no data left in the TXFIFO and that the data in the Shift register has been shifted out.
- The RXFIFO overflow interrupt means that the RXFIFO received more than 64 words and will not accept any other words.

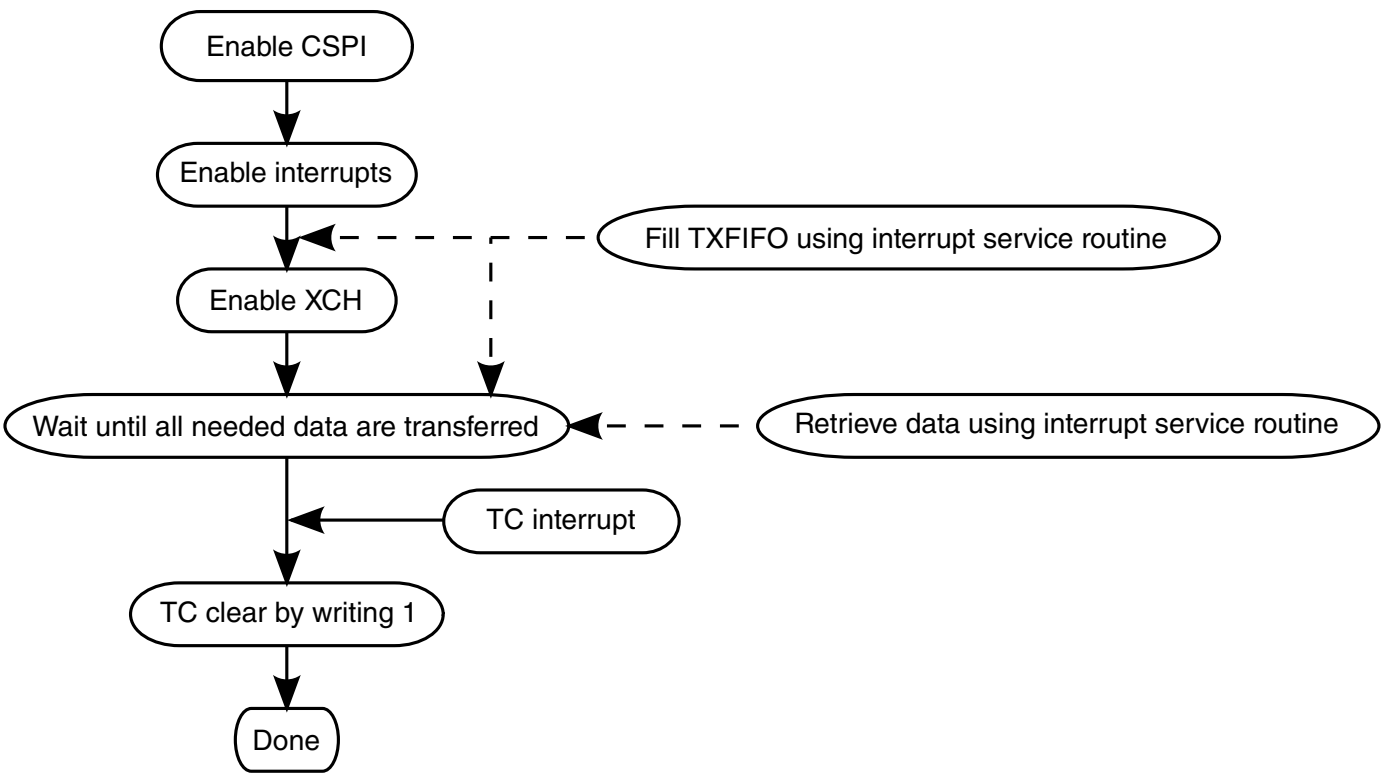


Figure 20-12. Program Sequence of SPI Burst Using Interrupt Control

20.4.7 DMA

DMA control provides another method to utilize the FIFOs in the ECSPI. By using DMA request and acknowledge signals, larger amounts of data can be transferred, and will reduce interrupts and host processor loading. When the appropriate conditions are matched, the block will send out a DMA request.

The DMA can deal with the following conditions:

- TXFIFO empty
- TXFIFO data request
- RXFIFO data request
- RXFIFO full

The figure below shows a program sequence of SPI bursts using DMA control.

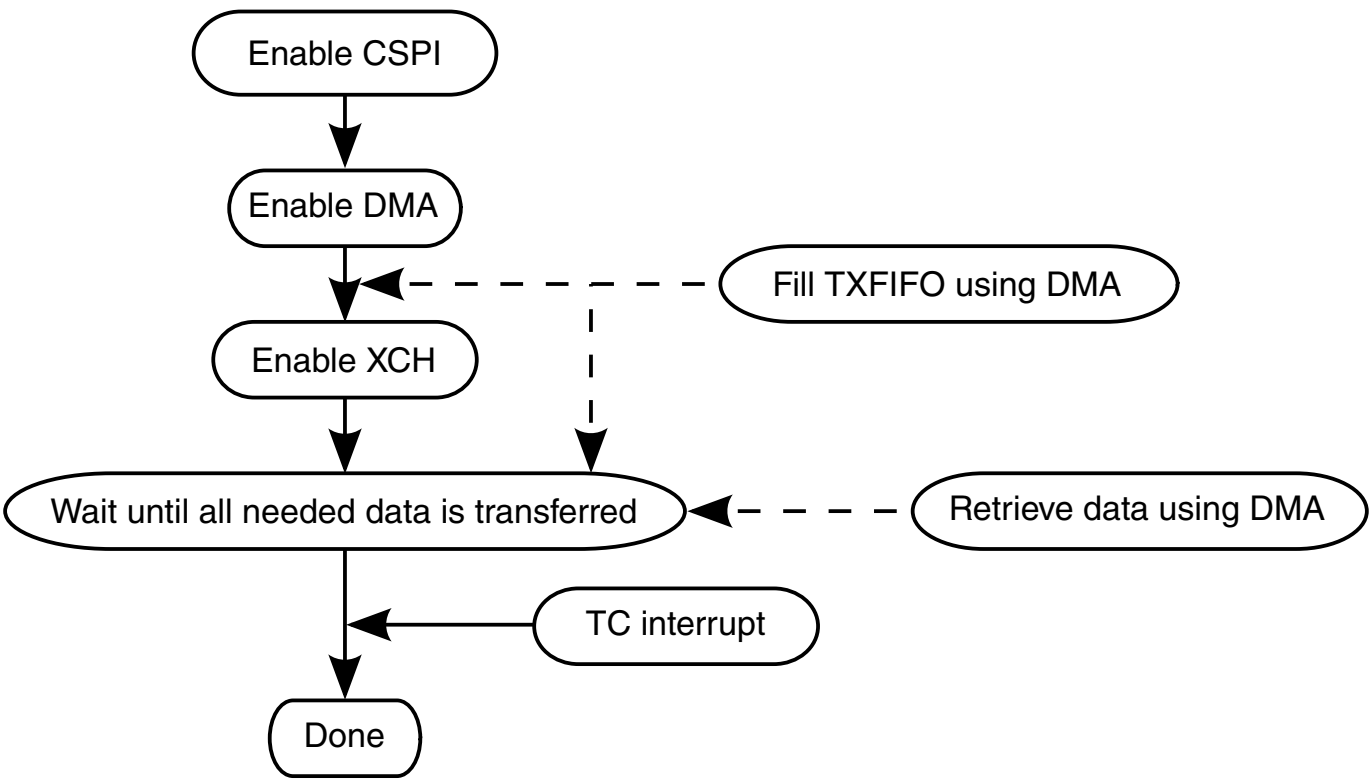


Figure 20-13. Program Sequence of SPI Burst Using DMA

20.4.8 Byte Order

The ECSPI does not support byte re-ordering in hardware.

20.5 Initialization

This section provides initialization information for ECSPI.

To initialize the block:

1. Clear the EN bit in ECSPI_CONREG to reset the block.
2. Enable the clocks for ECSPI.
3. Set the EN bit in ECSPI_CONREG to put ECSPI out of reset.
4. Configure corresponding IOMUX for ECSPI external signals.
5. Configure registers of ECSPI properly according to the specifications of the external SPI device.

20.6 Applications

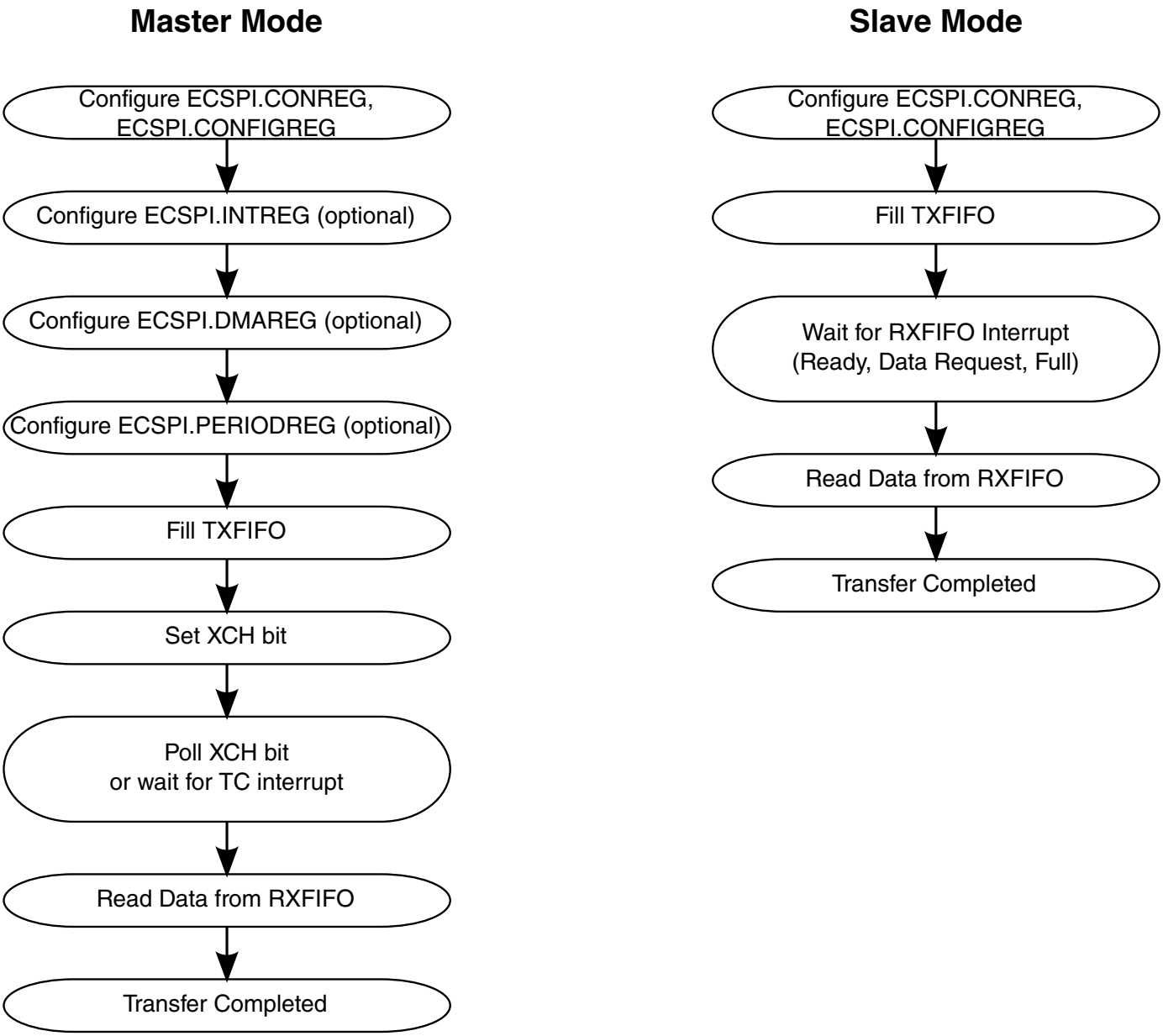


Figure 20-14. Flowchart of the ECSPI Operation

20.7 ECSPI Memory Map/Register Definition

This section includes the block memory map and detailed descriptions of all registers. For the base address of a particular block instantiation, see the system memory map.

ECSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200_8000	Receive Data Register (ECSPI1_RXDATA)	32	R	0000_0000h	20.7.1/787
200_8004	Transmit Data Register (ECSPI1_TXDATA)	32	W	0000_0000h	20.7.2/788
200_8008	Control Register (ECSPI1_CONREG)	32	R/W	0000_0000h	20.7.3/788
200_800C	Config Register (ECSPI1_CONFIGREG)	32	R/W	0000_0000h	20.7.4/791
200_8010	Interrupt Control Register (ECSPI1_INTREG)	32	R/W	0000_0000h	20.7.5/793
200_8014	DMA Control Register (ECSPI1_DMAREG)	32	R/W	0000_0000h	20.7.6/794
200_8018	Status Register (ECSPI1_STATREG)	32	R/W	0000_0003h	20.7.7/796
200_801C	Sample Period Control Register (ECSPI1_PERIODREG)	32	R/W	0000_0000h	20.7.8/797
200_8020	Test Control Register (ECSPI1_TESTREG)	32	R/W	0000_0000h	20.7.9/799
200_8040	Message Data Register (ECSPI1_MSGDATA)	32	W	0000_0000h	20.7.10/ 800
200_C000	Receive Data Register (ECSPI2_RXDATA)	32	R	0000_0000h	20.7.1/787
200_C004	Transmit Data Register (ECSPI2_TXDATA)	32	W	0000_0000h	20.7.2/788
200_C008	Control Register (ECSPI2_CONREG)	32	R/W	0000_0000h	20.7.3/788
200_C00C	Config Register (ECSPI2_CONFIGREG)	32	R/W	0000_0000h	20.7.4/791
200_C010	Interrupt Control Register (ECSPI2_INTREG)	32	R/W	0000_0000h	20.7.5/793
200_C014	DMA Control Register (ECSPI2_DMAREG)	32	R/W	0000_0000h	20.7.6/794
200_C018	Status Register (ECSPI2_STATREG)	32	R/W	0000_0003h	20.7.7/796
200_C01C	Sample Period Control Register (ECSPI2_PERIODREG)	32	R/W	0000_0000h	20.7.8/797
200_C020	Test Control Register (ECSPI2_TESTREG)	32	R/W	0000_0000h	20.7.9/799
200_C040	Message Data Register (ECSPI2_MSGDATA)	32	W	0000_0000h	20.7.10/ 800
201_0000	Receive Data Register (ECSPI3_RXDATA)	32	R	0000_0000h	20.7.1/787
201_0004	Transmit Data Register (ECSPI3_TXDATA)	32	W	0000_0000h	20.7.2/788
201_0008	Control Register (ECSPI3_CONREG)	32	R/W	0000_0000h	20.7.3/788
201_000C	Config Register (ECSPI3_CONFIGREG)	32	R/W	0000_0000h	20.7.4/791
201_0010	Interrupt Control Register (ECSPI3_INTREG)	32	R/W	0000_0000h	20.7.5/793
201_0014	DMA Control Register (ECSPI3_DMAREG)	32	R/W	0000_0000h	20.7.6/794
201_0018	Status Register (ECSPI3_STATREG)	32	R/W	0000_0003h	20.7.7/796
201_001C	Sample Period Control Register (ECSPI3_PERIODREG)	32	R/W	0000_0000h	20.7.8/797
201_0020	Test Control Register (ECSPI3_TESTREG)	32	R/W	0000_0000h	20.7.9/799
201_0040	Message Data Register (ECSPI3_MSGDATA)	32	W	0000_0000h	20.7.10/ 800
201_4000	Receive Data Register (ECSPI4_RXDATA)	32	R	0000_0000h	20.7.1/787
201_4004	Transmit Data Register (ECSPI4_TXDATA)	32	W	0000_0000h	20.7.2/788
201_4008	Control Register (ECSPI4_CONREG)	32	R/W	0000_0000h	20.7.3/788
201_400C	Config Register (ECSPI4_CONFIGREG)	32	R/W	0000_0000h	20.7.4/791
201_4010	Interrupt Control Register (ECSPI4_INTREG)	32	R/W	0000_0000h	20.7.5/793
201_4014	DMA Control Register (ECSPI4_DMAREG)	32	R/W	0000_0000h	20.7.6/794

Table continues on the next page...

ECSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
201_4018	Status Register (ECSPI4_STATREG)	32	R/W	0000_0003h	20.7.7/796
201_401C	Sample Period Control Register (ECSPI4_PERIODREG)	32	R/W	0000_0000h	20.7.8/797
201_4020	Test Control Register (ECSPI4_TESTREG)	32	R/W	0000_0000h	20.7.9/799
201_4040	Message Data Register (ECSPI4_MSGDATA)	32	W	0000_0000h	20.7.10/ 800

20.7.1 Receive Data Register (ECSPIx_RXDATA)

The Receive Data register (ECSPI_RXDATA) is a read-only register that forms the top word of the 64 x 32 receive FIFO. This register holds the data received from an external SPI device during a data transaction. Only word-sized read operations are allowed.

Address: Base address + 0h offset

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

ECSPIx_RXDATA field descriptions

Field	Description
ECSPI_RXDATA	Receive Data. This register holds the top word of the receive data FIFO. The FIFO is advanced for each read of this register. The data read is undefined when the Receive Data Ready (RR) bit in the Interrupt Control/Status register is cleared. Zeros are read when ECSPI is disabled.

20.7.2 Transmit Data Register (ECSPIx_TXDATA)

The Transmit Data (ECSPI_TXDATA) register is a write-only data register that forms the bottom word of the 64 x 32 TXFIFO. The TXFIFO can be written to as long as it is not full, even when the SPI Exchange bit (XCH) in ECSPI_CONREG is set. This allows software to write to the TXFIFO during a SPI data exchange process. Writes to this register are ignored when the ECSPI is disabled (ECSPI_CONREG[EN] bit is cleared).

Address: Base address + 4h offset

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

ECSPIx_TXDATA field descriptions

Field	Description
ECSPI_TXDATA	Transmit Data. This register holds the top word of data loaded into the FIFO. Data written to this register must be a word operation. The number of bits actually transmitted is determined by the BURST_LENGTH field of the corresponding SPI Control register. If this field contains more bits than the number specified by BURST_LENGTH, the extra bits are ignored. For example, to transfer 10 bits of data, a 32-bit word must be written to this register. Bits 9-0 are shifted out and bits 31-10 are ignored. When the ECSPI is operating in Slave mode, zeros are shifted out when the FIFO is empty. Zeros are read when ECSPI is disabled.

20.7.3 Control Register (ECSPIx_CONREG)

The Control Register (ECSPI_CONREG) allows software to enable the ECSPI , configure its operating modes, specify the divider value, and SPI_RDY control signal, and define the transfer length.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BURST_LENGTH												CHANNEL_ SELECT		DRCTL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRE_DIVIDER				POST_DIVIDER				CHANNEL_MODE				SMC	XCH	HT	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ECSPIx_CONREG field descriptions

Field	Description
31–20 BURST_ LENGTH	Burst Length. This field defines the length of a SPI burst to be transferred. The Chip Select (SS) will remain asserted until all bits in a SPI burst are shifted out. A maximum of 2^12 bits can be transferred in a single SPI burst. In master mode, it controls the number of bits per SPI burst. Since the shift register always loads 32-bit data from transmit FIFO, only the n least-significant (n = BURST LENGTH + 1) will be shifted out. The remaining bits will be ignored. Number of Valid Bits in a SPI burst. 0x000 A SPI burst contains the 1 LSB in a word. 0x001 A SPI burst contains the 2 LSB in a word. 0x002 A SPI burst contains the 3 LSB in a word. ... 0x01F A SPI burst contains all 32 bits in a word. 0x020 A SPI burst contains the 1 LSB in first word and all 32 bits in second word. 0x021 A SPI burst contains the 2 LSB in first word and all 32 bits in second word. ... 0xFFE A SPI burst contains the 31 LSB in first word and 2^7 -1 words. 0xFFF A SPI burst contains 2^7 words.
19–18 CHANNEL_ SELECT	SPI CHANNEL SELECT bits. Select one of four external SPI Master/Slave Devices. In master mode, these two bits select the external slave devices by asserting the Chip Select (SSn) outputs. Only the selected Chip Select (SSn) signal can be active at a given time; the remaining three signals will be negated. 00 Channel 0 is selected. Chip Select 0 (SS0) will be asserted. 01 Channel 1 is selected. Chip Select 1 (SS1) will be asserted. 10 Channel 2 is selected. Chip Select 2 (SS2) will be asserted. 11 Channel 3 is selected. Chip Select 3 (SS3) will be asserted.
17–16 DRCTL	SPI Data Ready Control. This field selects the utilization of the $\overline{\text{SPI_RDY}}$ signal in master mode. ECSPI checks this field before it starts an SPI burst. 00 The $\overline{\text{SPI_RDY}}$ signal is a don't care. 01 Burst will be triggered by the falling edge of the SPI_RDY signal (edge-triggered). 10 Burst will be triggered by a low level of the SPI_RDY signal (level-triggered). 11 Reserved.
15–12 PRE_DIVIDER	SPI Pre Divider. ECSPI uses a two-stage divider to generate the SPI clock. This field defines the pre-divider of the reference clock. 0000 Divide by 1. 0001 Divide by 2. 0010 Divide by 3. ... 1101 Divide by 14. 1110 Divide by 15. 1111 Divide by 16.
11–8 POST_DIVIDER	SPI Post Divider. ECSPI uses a two-stage divider to generate the SPI clock. This field defines the post-divider of the reference clock using the equation: 2^{n} . 0000 Divide by 1. 0001 Divide by 2.

Table continues on the next page...

ECSPIx_CONREG field descriptions (continued)

Field	Description
	0010 Divide by 4. ... 1110 Divide by 2^{14}. 1111 Divide by 2^{15}.
7–4 CHANNEL_ MODE	SPI CHANNEL MODE selects the mode for each SPI channel. CHANNEL MODE[3] is for SPI channel 3. CHANNEL MODE[2] is for SPI channel 2. CHANNEL MODE[1] is for SPI channel 1. CHANNEL MODE[0] is for SPI channel 0. 0 Slave mode. 1 Master mode.
3 SMC	Start Mode Control. This bit applies only to channels configured in Master mode (CHANNEL MODE = 1). It controls how the ECSPI starts a SPI burst, either through the SPI exchange bit, or immediately when the TXFIFO is written to. 0 SPI Exchange Bit (XCH) controls when a SPI burst can start. Setting the XCH bit will start a SPI burst or multiple bursts. This is controlled by the SPI SS Wave Form Select (SS_CTL). Refer to XCH and SS_CTL descriptions. 1 Immediately starts a SPI burst when data is written in TXFIFO.
2 XCH	SPI Exchange Bit. This bit applies only to channels configured in Master mode (CHANNEL MODE = 1). If the Start Mode Control (SMC) bit is cleared, writing a 1 to this bit starts one SPI burst or multiple SPI bursts according to the SPI SS Wave Form Select (SS_CTL). The XCH bit remains set while either the data exchange is in progress, or when the ECSPI is waiting for an active input if SPIRDY is enabled through DRCTL. This bit is cleared automatically when all data in the TXFIFO and the shift register has been shifted out. 0 Idle. 1 Initiates exchange (write) or busy (read).
1 HT	Hardware Trigger Enable. This bit is used in master mode only. It enables hardware trigger (HT) mode. Note, HT mode is not supported by this product. 0 Disable HT mode. 1 Enable HT mode.
0 EN	SPI Block Enable Control. This bit enables the ECSPI. This bit must be set before writing to other registers or initiating an exchange. Writing zero to this bit disables the block and resets the internal logic with the exception of the ECSPI_CONREG. The block's internal clocks are gated off whenever the block is disabled. 0 Disable the block. 1 Enable the block.

20.7.4 Config Register (ECSPIx_CONFIGREG)

The Config Register (ECSPI_CONFIGREG) allows software to configure each SPI channel, configure its operating modes, specify the phase and polarity of the clock, configure the Chip Select (SS), and define the HT transfer length. Note, HT mode is not supported by this product.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			HT_LENGTH					SCLK_CTL				DATA_CTL				SS_POL				SS_CTL				SCLK_POL				SCLK_PHA			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ECSPIx_CONFIGREG field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28–24 HT_LENGTH	HT LENGTH. This field defines the message length in HT Mode. Note, HT mode is not supported by this product. The length in bits of one message is (HT LENGTH + 1).
23–20 SCLK_CTL	SCLK CTL. This field controls the inactive state of SCLK for each SPI channel. SCLK CTL[3] is for SPI channel 3. SCLK CTL[2] is for SPI channel 2. SCLK CTL[1] is for SPI channel 1. SCLK CTL[0] is for SPI channel 0. 0 Stay low. 1 Stay high.
19–16 DATA_CTL	DATA CTL. This field controls inactive state of the data line for each SPI channel. DATA CTL[3] is for SPI channel 3. DATA CTL[2] is for SPI channel 2. DATA CTL[1] is for SPI channel 1. DATA CTL[0] is for SPI channel 0. 0 Stay high. 1 Stay low.
15–12 SS_POL	SPI SS Polarity Select. In both Master and Slave modes, this field selects the polarity of the Chip Select (SS) signal. SS POL[3] is for SPI channel 3. SS POL[2] is for SPI channel 2. SS POL[1] is for SPI channel 1. SS POL[0] is for SPI channel 0.

Table continues on the next page...

ECSPIx_CONFIGREG field descriptions (continued)

Field	Description
	0 Active low. 1 Active high.
11–8 SS_CTL	SPI SS Wave Form Select. In master mode, this field controls the output wave form of the Chip Select (SS) signal when the SMC (Start Mode Control) bit is cleared. The SS_CTL are ignored if the SMC bit is set. SS CTL[3] is for SPI channel 3. SS CTL[2] is for SPI channel 2. SS CTL[1] is for SPI channel 1. SS CTL[0] is for SPI channel 0. In slave mode, this bit controls when the SPI burst is completed. An SPI burst is completed by the Chip Select (SS) signal edges. (SSPOL = 0: rising edge; SSPOL = 1: falling edge) The RXFIFO is advanced whenever a Chip Select (SS) signal edge is detected or the shift register contains 32-bits of valid data. 0 In master mode - only one SPI burst will be transmitted. 1 In master mode - Negate Chip Select (SS) signal between SPI bursts. Multiple SPI bursts will be transmitted. The SPI transfer will automatically stop when the TXFIFO is empty. 1 Reserved
7–4 SCLK_POL	SPI Clock Polarity Control. This field controls the polarity of the SCLK signal. See Figure 20-10 for more information. SCLK_POL[3] is for SPI channel 3. SCLK_POL[2] is for SPI channel 2. SCLK_POL[1] is for SPI channel 1. SCLK_POL[0] is for SPI channel 0. 0 Active high polarity (0 = Idle). 1 Active low polarity (1 = Idle).
SCLK_PHA	SPI Clock/Data Phase Control. This field controls the clock/data phase relationship. See Figure 20-10 for more information. SCLK PHA[3] is for SPI channel 3. SCLK PHA[2] is for SPI channel 2. SCLK PHA[1] is for SPI channel 1. SCLK PHA[0] is for SPI channel 0. 0 Phase 0 operation. 1 Phase 1 operation.

20.7.5 Interrupt Control Register (ECSPIx_INTREG)

The Interrupt Control Register (ECSPI_INTREG) enables the generation of interrupts to the host processor. If the ECSPI is disabled, this register reads zero.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								TCEN	ROEN	RFEN	RDREN	RREN	TFEN	TDREN	TEEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ECSPIx_INTREG field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 TCEN	Transfer Completed Interrupt enable. This bit enables the Transfer Completed Interrupt. 0 Disable 1 Enable
6 ROEN	RXFIFO Overflow Interrupt enable. This bit enables the RXFIFO Overflow Interrupt. 0 Disable 1 Enable
5 RFEN	RXFIFO Full Interrupt enable. This bit enables the RXFIFO Full Interrupt. 0 Disable 1 Enable
4 RDREN	RXFIFO Data Request Interrupt enable. This bit enables the RXFIFO Data Request Interrupt when the number of data entries in the RXFIFO is greater than RX_THRESHOLD. 0 Disable 1 Enable
3 RREN	RXFIFO Ready Interrupt enable. This bit enables the RXFIFO Ready Interrupt. 0 Disable 1 Enable

Table continues on the next page...

ECSPIx_INTREG field descriptions (continued)

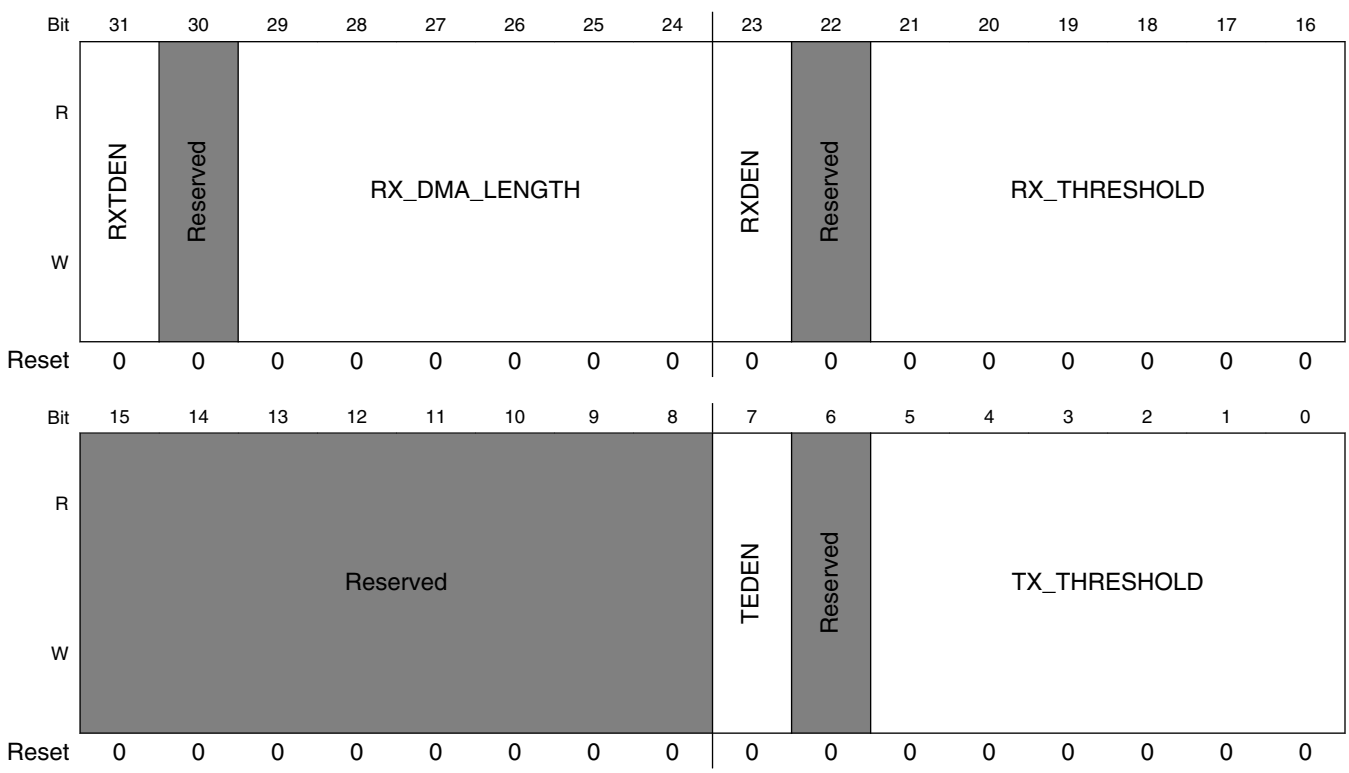
Field	Description
2 TFEN	TXFIFO Full Interrupt enable. This bit enables the TXFIFO Full Interrupt. 0 Disable 1 Enable
1 TDREN	TXFIFO Data Request Interrupt enable. This bit enables the TXFIFO Data Request Interrupt when the number of data entries in the TXFIFO is less than or equal to TX_THRESHOLD. 0 Disable 1 Enable
0 TEEN	TXFIFO Empty Interrupt enable. This bit enables the TXFIFO Empty Interrupt. 0 Disable 1 Enable

20.7.6 DMA Control Register (ECSPIx_DMAREG)

The Direct Memory Access Control Register (ECSPI_DMAREG) provides software a way to use an on-chip DMA controller for ECSPI data. Internal DMA request signals enable direct data transfers between the ECSPI FIFOs and system memory. The ECSPI sends out DMA requests when the appropriate FIFO conditions are matched.

If the ECSPI is disabled, this register is read as 0.

Address: Base address + 14h offset



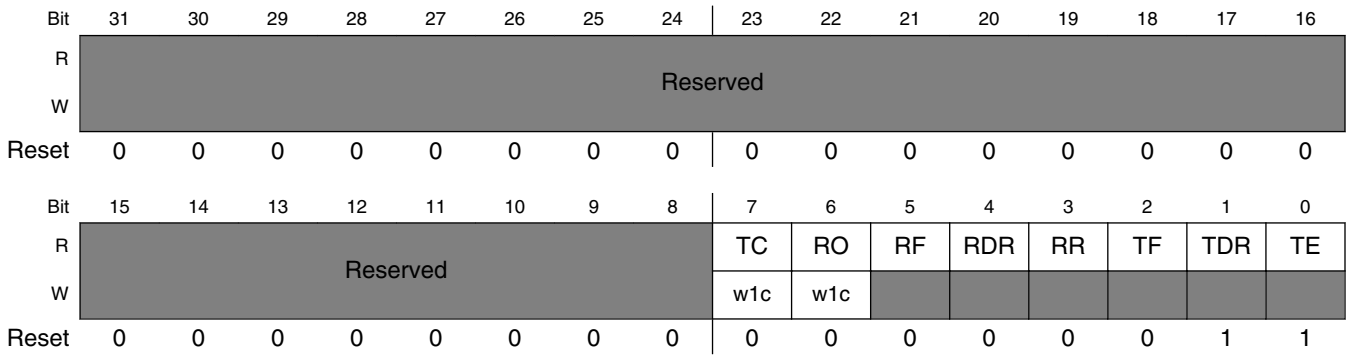
ECSPIx_DMAREG field descriptions

Field	Description
31 RXTDEN	RXFIFO TAIL DMA Request Enable. This bit enables an internal counter that is increased at each read of the RXFIFO. This counter is cleared automatically when it reaches RX DMA LENGTH. If the number of words remaining in the RXFIFO is greater than or equal to RX DMA LENGTH, a DMA request is generated even if it is less than or equal to RX_THRESHOLD. 0 Disable 1 Enable
30 -	This field is reserved. Reserved
29–24 RX_DMA_ LENGTH	RX DMA LENGTH. This field defines the burst length of a DMA operation. Applies only when RXTDEN is set.
23 RXDEN	RXFIFO DMA Request Enable. This bit enables/disables the RXFIFO DMA Request. 0 Disable 1 Enable
22 -	This field is reserved. Reserved
21–16 RX_ THRESHOLD	RX THRESHOLD. This field defines the FIFO threshold that triggers a RX DMA/INT request. A RX DMA/INT request is issued when the number of data entries in the RXFIFO is greater than RX_THRESHOLD.
15–8 -	This field is reserved. Reserved
7 TEDEN	TXFIFO Empty DMA Request Enable. This bit enables/disables the TXFIFO Empty DMA Request. 0 Disable 1 Enable
6 -	This field is reserved. Reserved
TX_ THRESHOLD	TX THRESHOLD. This field defines the FIFO threshold that triggers a TX DMA/INT request. A TX DMA/INT request is issued when the number of data entries in the TXFIFO is not greater than TX_THRESHOLD.

20.7.7 Status Register (ECSPIx_STATREG)

The ECSPI Status Register (ECSPI_STATREG) reflects the status of the ECSPI's operating condition. If the ECSPI is disabled, this register reads 0x0000_0003.

Address: Base address + 18h offset



ECSPIx_STATREG field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 TC	Transfer Completed Status bit. Writing 1 to this bit clears it. 0 Transfer in progress. 1 Transfer completed.
6 RO	RXFIFO Overflow. When set, this bit indicates that RXFIFO has overflowed. Writing 1 to this bit clears it. 0 RXFIFO has no overflow. 1 RXFIFO has overflowed.
5 RF	RXFIFO Full. This bit is set when the RXFIFO is full. 0 Not Full. 1 Full.
4 RDR	RXFIFO Data Request. 0 When RXTDE is set - Number of data entries in the RXFIFO is not greater than RX_THRESHOLD. 1 When RXTDE is set - Number of data entries in the RXFIFO is greater than RX_THRESHOLD or a DMA TAIL DMA condition exists. 0 When RXTDE is clear - Number of data entries in the RXFIFO is not greater than RX_THRESHOLD. 1 When RXTDE is clear - Number of data entries in the RXFIFO is greater than RX_THRESHOLD.
3 RR	RXFIFO Ready. This bit is set when one or more words are stored in the RXFIFO. 0 No valid data in RXFIFO. 1 More than 1 word in RXFIFO.
2 TF	TXFIFO Full. This bit is set when if the TXFIFO is full.

Table continues on the next page...

ECSPIx_STATREG field descriptions (continued)

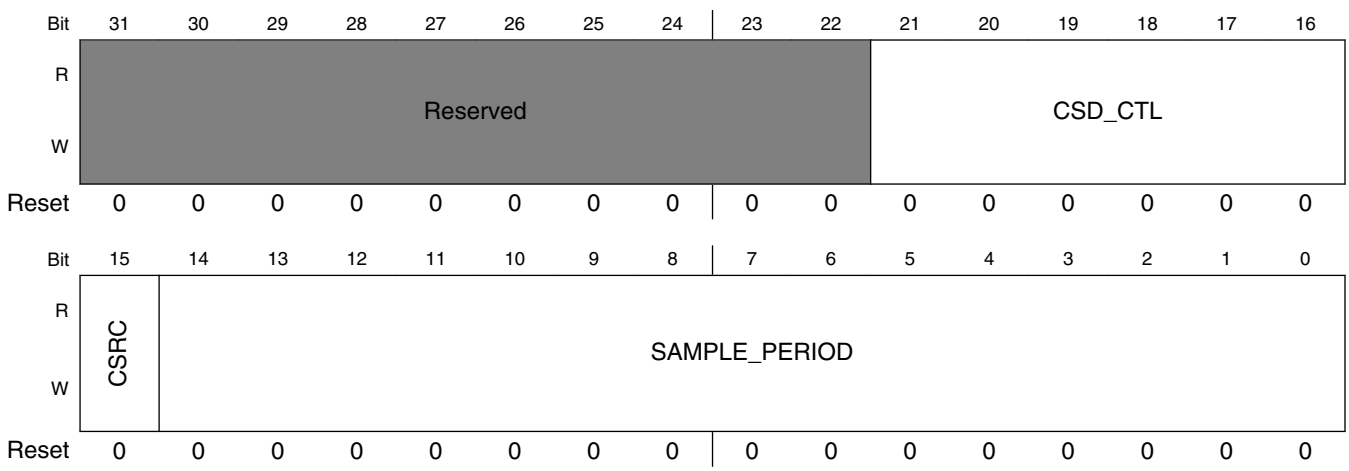
Field	Description
	0 TXFIFO is not Full. 1 TXFIFO is Full.
1 TDR	TXFIFO Data Request. 0 Number of valid data slots in TXFIFO is greater than TX_THRESHOLD. 1 Number of valid data slots in TXFIFO is not greater than TX_THRESHOLD.
0 TE	TXFIFO Empty. This bit is set if the TXFIFO is empty. 0 TXFIFO contains one or more words. 1 TXFIFO is empty.

20.7.8 Sample Period Control Register (ECSPIx_PERIODREG)

The Sample Period Control Register (ECSPI_PERIODREG) provides software a way to insert delays (wait states) between consecutive SPI transfers. Control bits in this register select the clock source for the sample period counter and the delay count indicating the number of wait states to be inserted between data transfers.

The delay counts apply only when the current channel is operating in Master mode (ECSPI_CONREG[CHANNEL MODE] = 1).ECSPI_PERIODREG also contains the CSD CTRL field used to insert a delay between the Chip Select's active edge and the first SPI Clock edge.

Address: Base address + 1Ch offset



ECSPIx_PERIODREG field descriptions

Field	Description
31–22 -	This field is reserved. Reserved

Table continues on the next page...

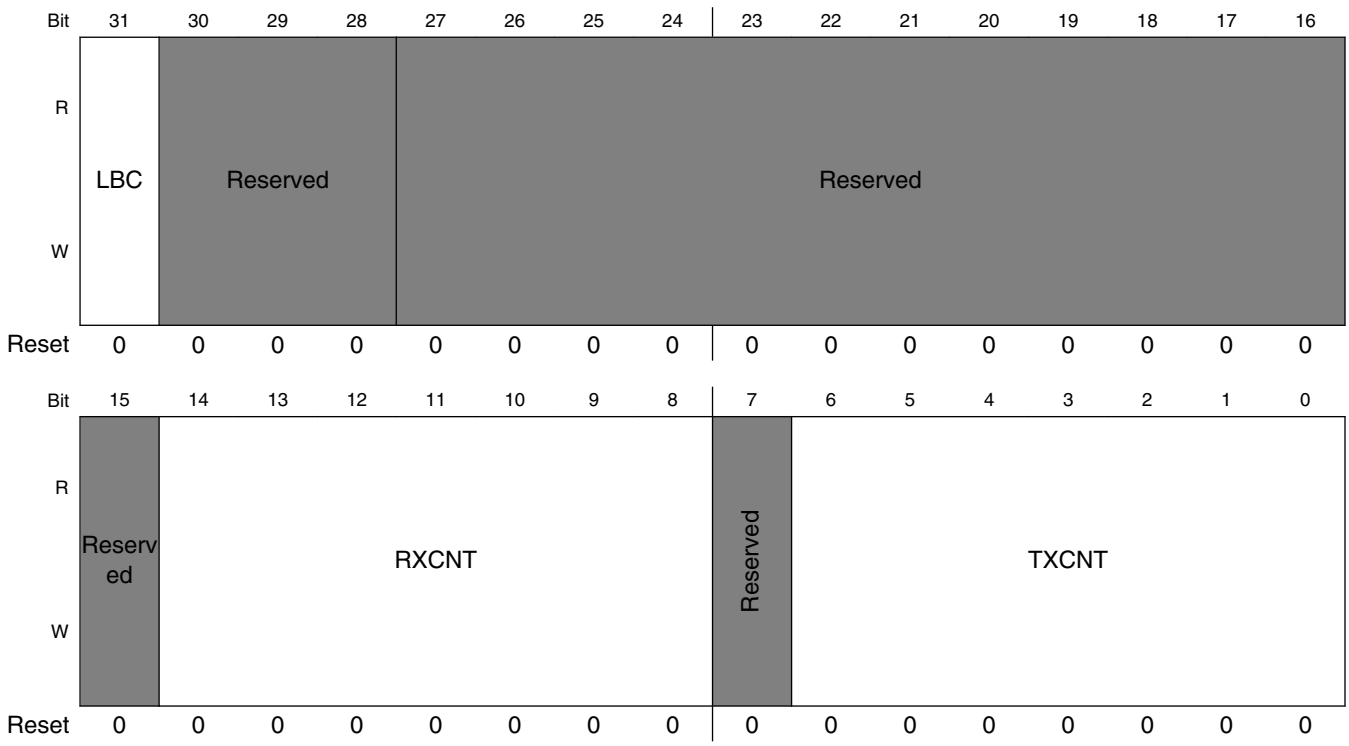
ECSPIx_PERIODREG field descriptions (continued)

Field	Description
21–16 CSD_CTL	Chip Select Delay Control bits. This field defines how many SPI clocks will be inserted between the chip select's active edge and the first SPI clock edge. The range is from 0 to 63.
15 CSRC	Clock Source Control. This bit selects the clock source for the sample period counter. 0 SPI Clock (SCLK) 1 Low-Frequency Reference Clock (32.768 KHz)
SAMPLE_ PERIOD	Sample Period Control. These bits control the number of wait states to be inserted in data transfers. During the idle clocks, the state of the SS output will operate according to the SS_CTL control field in the ECSPI_CONREG register. 0x0000 0 wait states inserted 0x0001 1 wait state inserted ... ... 0x7FFE 32766 wait states inserted 0x7FFF 32767 wait states inserted

20.7.9 Test Control Register (ECSPIx_TESTREG)

The Test Control Register (ECSPI_TESTREG) provides software a mechanism to internally connect the receive and transmit devices of the ECSPI, and monitor the contents of the receive and transmit FIFOs.

Address: Base address + 20h offset



ECSPIx_TESTREG field descriptions

Field	Description
31 LBC	Loop Back Control. This bit is used in Master mode only. When this bit is set, the ECSPI connects the transmitter and receiver sections internally, and the data shifted out from the most-significant bit of the shift register is looped back into the least-significant bit of the Shift register. In this way, a self-test of the complete transmit/receive path can be made. The output pins are not affected, and the input pins are ignored. 0 Not connected. 1 Transmitter and receiver sections internally connected for Loopback.
30–28 -	This field is reserved. Reserved, all bits should be ignored.
27–15 -	This field is reserved. Reserved
14–8 RXCNT	RXFIFO Counter. This field indicates the number of words in the RXFIFO.

Table continues on the next page...

ECSPIx_TESTREG field descriptions (continued)

Field	Description
7 -	This field is reserved. Reserved
TXCNT	TXFIFO Counter. This field indicates the number of words in the TXFIFO.

20.7.10 Message Data Register (ECSPIx_MSGDATA)

The Message Data Register (ECSPI_MSGDATA) forms the top word of the 16 x 32 MSG Data FIFO. Only word-size accesses are allowed for this register. Reads to this register return zero, and writes to this register store data in the MSG Data FIFO.

Address: Base address + 40h offset

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

ECSPIx_MSGDATA field descriptions

Field	Description
ECSPI_ MSGDATA	ECSPI_MSGDATA holds the top word of MSG Data FIFO. The MSG Data FIFO is advanced for each write of this register. The data read is zero. The data written to this register is stored in the MSG Data FIFO.

Chapter 21
External Interface Module (EIM)

21.1 Overview

The EIM handles the interface to devices external to the chip, including generation of chip selects, clock and control for external peripherals and memory. It provides asynchronous access to devices with SRAM-like interface and synchronous access to devices with NOR-Flash-like or PSRAM-like interface.

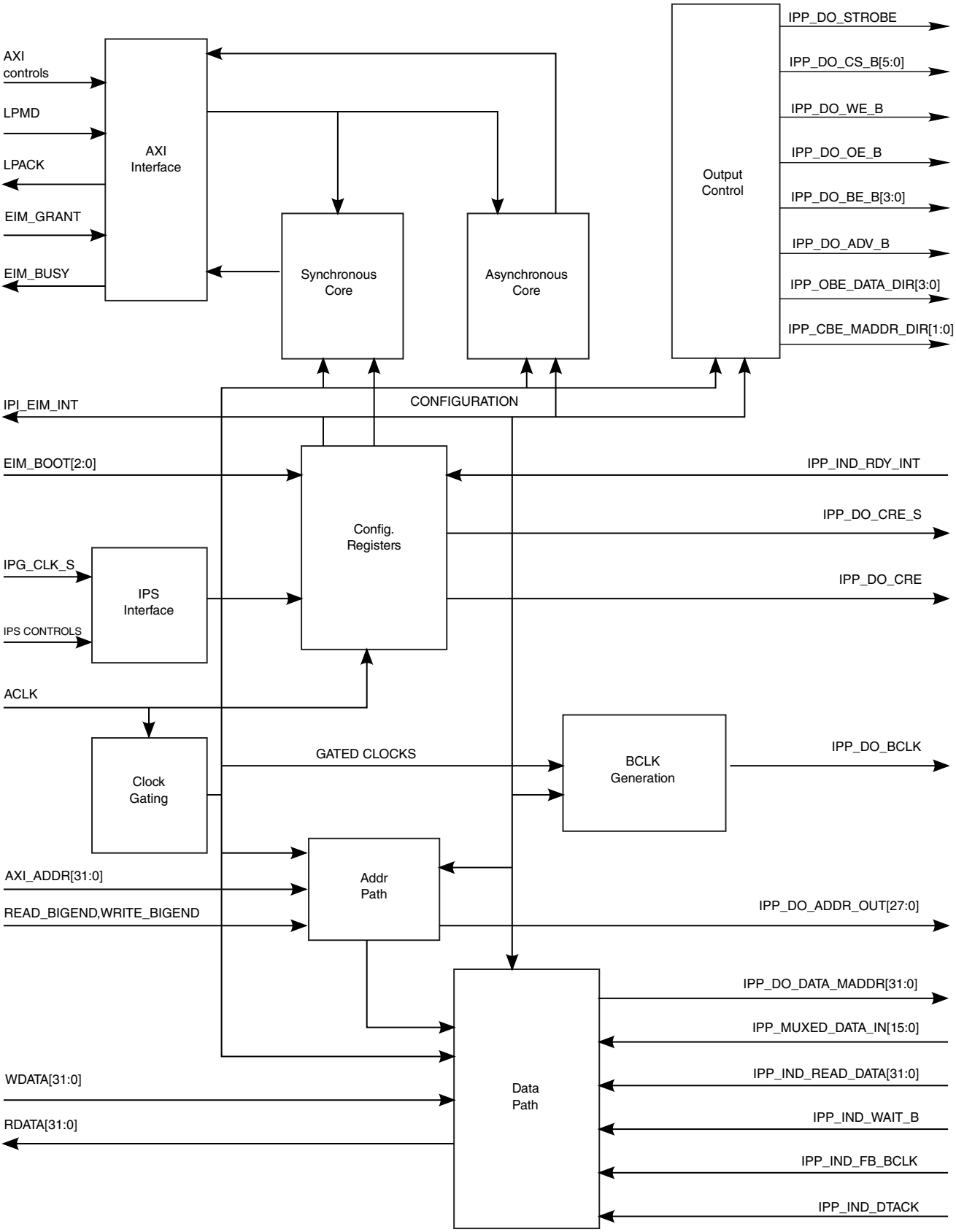


Figure 21-1. EIM Diagram

21.1.1 Features

- Four chip selects for external devices
  - Flexible address decoding. Each chip select memory space determined separately, according to VIA port configuration (see Chip Select Memory Map). Configurable Chip Select 0 base address (by VIA)
  - Individual select signal for each one of the memory space defined. Up to 6 memory spaces may be defined and programmed individually.
  - 128 MByte maximum supported density by default (AUS bit is cleared). When the AUS bit is set, maximum supported density is 32 MBytes.
- Selectable Write Protection for each Chip Select
- Support for multiplexed address / data bus operation x16 port size
- Programmable Data Port Size for each Chip Select (x8, x16)
- Programmable Wait-State generator for each Chip Select, for write and read accesses separately
- Asynchronous accesses with programmable setup and hold times for control signals
- Support for Asynchronous page mode accesses (x16 port size)
- Independent synchronous Memory Burst Read Mode support for NOR-Flash and PSRAM memories (x16 port size)
- Independent synchronous Memory Burst Write Mode support for PSRAM and NOR-Flash like memories (CellularRAM™ from Micron, Infineon, and Cypress, OneNAND™ and utRAM™ from Samsung, and COSMORAM™ from Toshiba)
- Support of NAND-Flash devices with NOR-Flash like interface - MDOC™ (M-Systems), OneNAND™ (Samsung)
- Independent programmable variable/fix Latency support for read and write synchronous (burst) mode
- Support for Big Endian and Little Endian operation modes per access
- ARM AXI slave interface. One ID at a time support.

21.1.2 Modes of Operation

The EIM has the following modes of operation:

- Asynchronous Mode
- Asynchronous Page Mode
- Multiplexed Address/Data mode
- Burst Clock Mode
- Low Power Modes
- Boot Mode

See details in the EIM Operational Modes.

21.1.2.1 Asynchronous Mode

This is a non-burst mode that is used for SRAM access. In this mode, a single data is read/written with each access (asserted address).

All controls' timings are controlled by preset values in Chip Select Configuration Registers.

21.1.2.2 Asynchronous Page Read Mode

Setting the APR bit causes the EIM to perform memory burst accesses by emulating page mode operation.

The external address asserts for each piece of data. The initial access timing is according to RWSC field, and the next address assertions timing is according to PAT field. When APR bit is set, RCSN OEN, RADVN and RBEN fields are ignored for burst access to the external device.

The page size can be set via the BL field to 2, 4, 8, 16, or 32 words (the word size is determined by the DSZ field).

21.1.2.3 Multiplexed Address/Data Mode

In this mode, multiplexing addresses and data bits on the same pins is supported for synchronous/asynchronous accesses to x8/x16 data width memory devices.

For more information about the pins that drive data/address in 8/16 non-muxed mode and 16 muxed mode, see the following table.

Table 21-1. EIM Multiplexing

Setup	Non Multiplexed Addresss/Data Mode						Multiplexed Address/ Data Mode
	8-Bit				16-Bit		16 Bit
	NUM = 0, DSZ = 100	NUM = 0, DSZ = 101	NUM = 0, DSZ = 110	NUM = 0, DSZ = 111	NUM = 0, DSZ = 001	NUM = 0, DSZ = 010	NUM = 1, DSZ = 001
EIM_ADDR[15:00]	EIM_AD[15:00]	EIM_AD[15:00]	EIM_AD[15:00]	EIM_AD [15:00]	EIM_AD[15:00]	EIM_AD[15:00]	EIM_AD[15:00]
EIM_ADDR[26:16]	EIM_ADDR[26:16]	EIM_ADDR[26:16]	EIM_ADDR[26:16]	EIM_ADDR[26:16]	EIM_ADDR[26:16]	EIM_ADDR[26:16]	EIM_ADDR[26:16]

Table continues on the next page...

Table 21-1. EIM Multiplexing (continued)

Setup	Non Multiplexed Addresss/Data Mode						Multiplexed Address/ Data Mode
	8-Bit				16-Bit		16 Bit
	NUM = 0, DSZ = 100	NUM = 0, DSZ = 101	NUM = 0, DSZ = 110	NUM = 0, DSZ = 111	NUM = 0, DSZ = 001	NUM = 0, DSZ = 010	NUM = 1, DSZ = 001
EIM_DATA[07:00], EIM_EB0_B	EIM_DATA[07:00]	-	Reserved	Reserved	EIM_DATA[07:00]	Reserved	EIM_AD[07:00]
EIM_DATA[15:8], EIM_EB1_B	-	EIM_DATA[15:08]	Reserved	Reserved	EIM_DATA[15:08]	Reserved	EIM_A[15:08]

21.1.2.4 Burst Clock Mode

The controller has the ability to support burst synchronous operations in various frequencies, depending on the frequency of the input clock supplied by the system (EIM clock).

The EIM clock can be divided by one, two, three or four, and its frequency can be changed according to the requirements. Variable and fix latency are supported for this mode, according to the external device requirements.

- Synchronous read mode. This is a burst mode, which is used for reading from Flash/ PSRAM memory devices. In this mode, after address assertion a burst of sequential data can be read. Data exchange is carried out according to BCLK being generated by EIM. An access is delayed according to external WAIT_B signal assertion (signal from the memory device).
- Synchronous write mode. A burst mode used for accessing external devices, which support synchronous write type of access (PSRAM protocol). In this mode, after address assertion a burst of sequential data can be written to the external device. Access may be delayed according to WAIT_B signal assertion (signal from the memory device) before first piece of data arrived to the external device.

NOTE

Maximum frequency of the EIM main clock is 133 Mhz. It may be reduced by the system for special cases of external devices, which demand a different frequency then integer division of the 133 MHz clock.

21.1.2.5 Low Power Modes

The input clock is gated by ACT_CS bits. When all the ACT_CS are negated (all CS disable) the internal clock is turned off; awready/wready & arready signal are de-asserted and the master can't access the EIM.

21.1.2.6 Boot Mode

It is possible to perform a boot operation from external device located on CS0. The configuration of the relevant bits are done with boot mode signals according to the external device parameters (for example, port size and protocol assertion).

See for more details.

21.2 External Signals

The following table describes the external signals of EIM:

21.2.1 Other Important Block I/O Signals Internal to the SoC

The following table provides a description of other signals which are internal to the i.MX 6UltraLite that are important to understand the function of EIM.

Name	I/O	Description
EIM_FB_BCLK	Input	Burst Clock Feedback. This block input is used to sample read data during high transfer speeds. The signal provides feedback from the I/O pad of the BCLK output pin and tends to align more closely with data from the external memory device.
EIM_BOOT	Input	EIM Boot Configuration. These block inputs determine the reset state of DSZ[1:0] and MUM.
ACLK	Input	AXI clock, maximum frequency 133 Mhz
IPG_CLK_S	Input	EIM module IPG clock
RST_B	Input	Active low HW reset
EIM_WARM_RESET	Input	Warm Reset. If this signal is asserted the rst_b will reset only the internal FF and state machine while S/W registers will keep their current state. This signal is active high signal.

21.3 Clocks

The following table describes the clock sources for EIM. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 21-2. EIM Clocks

Clock name	Clock Root	Description
aclk	aclk_eim_slow_clk_root	EIM clock (main)
aclk_slow	aclk_eim_slow_clk_root	EIM clock (slow)
ipg_clk_s	ipg_clk_root	Peripheral access clock
aclk_exsc	aclk_eim_slow_clk_root	EIM clock (external device)

- ACLK: EIM clock (main clock, AXI clock) with a Max frequency of 133 MHz. Can be gated externally when there is no active AXI access.
- ACLK_SLOW: EIM all time running ACLK. Used for flip-flops that must be active even when EIM is in low power down mode to provide clock for lpack/lpmd registers, IP registers and IP to AXI sync registers.
- IPG_CLK_S: IPG clock for IP accesses. IP registers are activated by ACLK_SLOW clock.
- ACLK_EXSC: Clock created from EIM clock for External device usage. Integer division by 1, 2, 3 and 4 of the clock can be use with BCD bit field configuration, according to external devices demands. EIM clock frequency may be reduced for lower frequency support which cannot be achieved via BCD bit field.

21.4 Chip Select Memory Map

The EIM memory space is mapped into 128 MB total memory space in the processor memory. For addresses, see the System memory map table and the CM4 memory map table. The total 128 MB of memory can be divided among the EIM four chip selects. The memory configuration across the chip selects is controlled by the IOMUXC_GPR1 register. The ADDRSn[10] fields control how much memory is allotted to each chip select.

The following four configurations are supported:

- CS0 (128 MB), CS1 (0 MB), CS2 (0 MB), CS3 (0 MB) [default configuration]
- CS0 (64 MB), CS1 (64 MB), CS2 (0 MB), CS3 (0 MB)
- CS0 (64 MB), CS1 (32 MB), CS2 (32 MB), CS3 (0 MB)
- CS0 (32 MB), CS1 (32 MB), CS2 (32 MB), CS3 (32 MB)

21.5 Functional Description

This section provides the functional description for the EIM.

21.5.1 Bus Sizing Configuration

The EIM supports byte, half word and word operands allowing access to x8, x16, x32 ports. It can be address/data multiplexed in x16, x32 ports. The port size is programmable via the DSZ bit field in the corresponding Chip Select Configuration Register. An 8-bit port can reside in each one of the bytes of the data bus. A 16-bit port can reside on the lower 16 bits of the data bus, DATA_IN/OUT[15:0] or on the higher 16 bits of the data bus, DATA_IN/OUT[31:16].

In the case of a multi-cycle transfer, the lower two address bits (ADDR[1:0]) are incremented appropriately. The EIM address bus is configured according to DSZ bit field and AUS bits. There is either one bit (for x16 port size) or two bits (for x32 port size) right shift of the address bits (only when AUS=0) and no bit shift when AUS = 1 or DSZ[2] = 1.

The EIM has a data multiplexer which takes the four bytes of the AXI data bus and routes them to their required positions to properly interface to memory.

NOTE

A word access to or from a x16 port requires two external bus cycles to complete the transfer.

A word access to or from a x8 port requires four external bus cycles to complete the transfer.

21.5.1.1 8 BIT PORT SUPPORT

EIM has limited support for mot68000 & intel 386 protocols.

21.5.1.1.1 MOTOROLA 68000

EIM has limited support for mot68000 protocol. Only basic read or write asynchronous operations are supported.

The following operations are not supported:

- Read modify write
- Sync access
- All special accesses (ARM platform space, bus arbitration, bus control, bus error & reset operations)
- FC outputs

21.5.1.1.2 INTEL 386

EIM has limited support for intel 386 protocol. Only basic read or write async non-pipelined operations are supported.

The following operations are not supported:

- Other bus cycles (interrupt, halt & refresh)
- Bus lock
- M/IO, DC, LBA, NA, REFRESH & BS8 signals

21.5.2 EIM Operational Modes

Listed here are the main operational modes for EIM selected by control bit fields settings.

For details, see the bit field descriptions of SWR / SRD / MUM. All modes are supported in with 8-, 16- or 32-bit port configuration, according to DSZ bit field.

Table 21-3. EIM Operation Modes Field Settings

Control bit fields			Brief mode description
MUM	SRD	SWR	
0	0	0	Asynchronous write / Asynchronous read for APR=0 / Asynchronous page read for APR=1, none multiplexed
		1	Synchronous write/ Asynchronous read or APR=0 / Asynchronous page read for APR=1,none multiplexed
	1	0	Asynchronous write/Synchronous read none multiplexed
		1	Synchronous write/read none multiplexed
1	0	0	Asynchronous write/read multiplexed
		1	Synchronous write/ Asynchronous read multiplexed
	1	0	Asynchronous write/Synchronous read multiplexed
		1	Synchronous write/read multiplexed

21.5.3 Burst Mode (Synchronous) Memory Operation

This mode is enabled for read or write access. Bit SWR sets the burst mode for write operations at the corresponding chip select and bit SRD sets it for read operation.

When this mode is set, the controller attempts to translate the Master burst accesses to memory burst accesses, being limited by the memory burst length, predefined by BL value, or memory and Master WRAP/INCR boundary crossing non-matching. Only the first address accessed is put by the controller on the external address bus in a memory burst sequence.

EIM may translate from some Master sequential accesses to one or several memory bursts, but not from two Master individual accesses to one memory burst.

For the first access in a memory burst sequence, the EIM asserts $\overline{\text{ADV}}$, causing the external burst device to latch the starting burst address; then toggle the burst clock (BCLK) for a predefined number of cycles in order to latch the first unit of data. Subsequent accessed data units can then be burst in fewer clock cycles, realizing an overall increase in bus bandwidth.

NOTE

The BCLK signal toggles only when burst access is executed toward the external device (BCM=1'b0 for normal mode use). It runs with a 50% duty cycle until the end of access is reached. When access is terminated, BCLK stops toggling.

Memory burst accesses are terminated by the EIM whenever it detects the following:

- The specific burst length has executed completely (end of access)
- Write access - missing data in write buffer (Master is delaying the data transfer toward the EIM)
- Next sequential access crosses boundary with unequal condition (wrap/increment, burst length) on the Master and memory
- Current memory burst length reached

21.5.4 Burst Clock Divisor (BCD)

In some cases, it may be necessary to slow the external bus in relation to the internal bus to allow accesses to burst devices that have a maximum operating frequency less than the operating frequency of the internal bus.

The internal bus frequency can be divided by one, two, three or four for presentation on the external bus in burst mode operation.

BCLK can only be set to integer divisions of the incoming clock frequency. To get a specific frequency on BCLK, configure the divider to change the incoming EIM clock accordingly.

By programming the BCD bit field to various values, two signals on the external bus are affected; $\overline{ADV}$ and BCLK. The $\overline{ADV}$ signal is asserted according to RADVA or WADVA bit fields programming, and is negated according to the formula mentioned in RADVN and WADVN bit fields description. The BCLK signal runs with a 50% duty cycle until the end of access is reached.

If BCM = 1, the BCLK runs at frequency according to GBCD bit field settings on every async memory access, regardless of the SWR and SRD bits configuration. Caution should be exercised when using BCM bit; GBCD bit field should be updated once and should not change when BCLK is toggling. The BCM bit is used mainly for system debug mode. It has no functional use of the EIM in normal mode.

21.5.5 Burst Clock Start (BCS)

In an effort to allow greater flexibility in achieving the minimum number of wait states on burst accesses, you can determine when you want the BCLK to start toggling after the start of access. This allows the BCLK to be skewed from point of data capture on the EIM clock by any number of EIM clock cycles.

Care must be exercised when setting BCS bit field in conjunction with the BCD and RWSC/WWSC bit fields. See the external timing diagrams in Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=1 and Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=0 for examples of how to use the BCS, BCD and RWSC/WWSC bit fields together.

21.5.6 Multiplexed Address/Data Mode Support

The control bit MUM allows support memory with multiplexed address/data bus both in asynchronous and in synchronous modes.

Caution should be exercised for using OEA/WEA & ADH bit fields. They should be configured according to the external device requirements, as it determines the time point of end of address phase and start of data phase.

21.5.7 Mixed Master/Memory Burst Modes Support

To provide mixed sequential/wrap accesses with different length, EIM interprets burst signal and generate additional $\overline{ADV}$ signals whenever there appear unequal address or burst boundary crossing condition.

BL bit field is used to notify EIM about current memory burst and wrap condition for properly external address generation. In case of non-matching boundaries in both the memory and Master access, EIM starts a new memory burst access by updating address from Master on address bus and generating $\overline{\text{ADV}}$ signal.

21.5.8 AXI (Master) Bus Cycles Support

The EIM uses an ARM AXI slave interface. It has a 32-bit bus and supports one access (one ID) at a time. No out of order or parallel accesses are supported.

The following AXI protocol signals are not supported:

- AWLOCK
- AWCACHE
- ARLOCK
- ARCACHE

ARID bus is sampled when:

- new read access is valid on the read address channel and is reflected on the RID bus output toward the master.

AWID bus is sampled when:

- new write access is valid on the write address channel and is reflected on the WID/BID bus output toward the master.

ARPROT and AWPROT signal are partially used. ARPROT[0] and AWPROT[0] bits are used for normal/privileged access detection. ARPROT[2:1] and AWPROT[2:1] are not used.

When sampling a valid access on both of the address channels, the read access will be performed first while write access is pending. After last data transfer completed, the pending write will be executed.

A new access may be executed one cycle after sampling a valid access on the read or write address channels, assuming there is no current access (back to back) which can cause a recovery or end of access penalty cycles, for write access, also assuming data is in write buffer for fast execution.

NOTE

- Only 32-bit word size accesses are supported for burst mode accesses.
- Only 8-bit (1 byte), 16-bit (2 byte) and 32-bit (4 byte) word size supported for single access.

- Maximum number of burst length is 16.
- According to AXI protocol, burst access should not cross 4 KB blocks. In case EIM gets an access that crosses the 4 KB, memory address calculation is invalid.

AXI transfers shown in the table below are also supported. These AXI cycles will be translated into the necessary cycles on the memory side. For example, for optimal operation in case ARM cache is configured to 8 beat burst with wrap, a synchronous flash and cellular RAM memory should be configured in 16 word wrap burst mode when using a 16-bit data port, and in 8 word wrap burst mode when using a 32-bit data port. EIM uses BL bit field to support different memory configurations. The controller splits the transaction when needed in some cases. See Table 21-5.

Table 21-4. AXI Burst Cycles Supported

Burst Length - Number of data transfers	Burst size - Bytes in transfer	Burst type	Description
1	1	INCR	Single transfer
1	2	INCR	Single transfer
1	4	INCR	Single transfer
2	4	WRAP	2-beat wrapping burst
4	4	WRAP	4-beat wrapping burst
8	4	WRAP	8-beat wrapping burst
16	4	WRAP	16-beat wrapping burst
2	4	INCR	2-beat incrementing burst
3	4	INCR	3-beat incrementing burst
4	4	INCR	4-beat incrementing burst
5	4	INCR	5-beat incrementing burst
6	4	INCR	6-beat incrementing burst
7	4	INCR	7-beat incrementing burst
8	4	INCR	8-beat incrementing burst
9	4	INCR	9-beat incrementing burst
10	4	INCR	10-beat incrementing burst
11	4	INCR	11-beat incrementing burst
12	4	INCR	12-beat incrementing burst
13	4	INCR	13-beat incrementing burst
14	4	INCR	14-beat incrementing burst
15	4	INCR	15-beat incrementing burst
16	4	INCR	16-beat incrementing burst

Table 21-5. AXI to Memory Burst Splits Number

AXI Burst Type	Memory Burst Type Config.	# of accesses to X8 Memory Port size	# of accesses to X16 Memory Port size	# of accesses to X32 Memory Port size
INC16	WRAP4	16	8	4
Aligned Addr.	Cont.	1	1	1
INC16	WRAP4	17	9	5
Unaligned Addr.	Cont.	1	1	1
WRAP16 Aligned Addr.	WRAP16	4	2	1
	Cont.	1	1	1
WRAP16 Unaligned Addr.	WRAP16	5	3	1
	Cont.	2	2	2
INC8	WRAP8	4	2	1
Aligned Addr.	WRAP16	2	1	1
INC8	WRAP8	4 or 5	2 or 3	2
Unaligned Addr.	WRAP16	2 or 3	2	1 or 2
WRAP8	WRAP16	2	1	1
Aligned Addr.	Cont.	1	1	1
WRAP8 Unaligned Addr.	WRAP16	2 or 3	1	2
	Cont.	2	2	2

21.5.9 WAIT_B Signal, RWSC and WWSC bit fields Usage

Most of the external devices supporting burst mode for write or read accesses provide a signal which indicates data is valid on the memory bus (a.k.a. handshake mode). For this mode, RFL and WFL bits should be cleared and RWSC/ WWSC bit fields indicate when the controller should start sampling this signal from the external device or, in other words, how many BCLK cycles should be masked.

For devices which do not use this signal or have a fixed latency ability, the RFL and WFL bits may be set for internal calculation regarding BCLK cycles penalty until data is valid (memory initial access time). For this mode, RWSC/ WWSC indicates when the data is ready for sampling by the controller (read access) or the external device (write access). There is separation between read and write accesses wait-state control. For read access, RWSC bit field is valid and WWSC bit field is ignored; for write access, WWSC is valid and RWSC is ignored.

21.5.10 IPS Register Interface

Access to the registers of the EIM, read or write, is made with IPS protocol signals. The system should avoid changing the registers while master/memory transaction is valid, as this can cause an unknown behavior of the controller.

Register access size is 32-bit as the register size definition, other size of access (byte or half word) is not supported.

21.5.11 MRS Set for PSRAM

Memory registers of PSRAM devices can be configured according to external signal, which indicates whether the access is to a memory array or memory register domain.

When the CRE bit is set, the following transactions to the external device will assert the CRE signal. The polarity of this signal is determined by the CREP bit for active low or active high assertion of the signal.

21.5.12 EIM Access Termination

EIM is monitoring the corresponding CSx control signal every time variable latency access or dtack access is performed toward the external device.

In variable latency accesses, the Watchdog Timer (WDOG-1) counts BCLK cycles. If it reaches the wdog_limit (according to the WDOG_LIMIT bit field in the WCR) before the device signals can drive/sample new data, the controller will terminate the access and generate an error response transfer toward the Master.

In dtack access, WDOG-1 counts ACLK cycles instead of BCLK and it reaches the wdog_limit before the device asserts the dtack signal, the controller will terminate the access and generate an error response transfer toward the Master.

WDOG-1 can by disabled by WDOG_EN bit in the WCR.

21.5.13 Error Conditions

The following conditions cause an error (AXI error or IPS error) response signal:

- AXI errors
  - Access to a disabled chip select - access to a mapped chip select address space where the CSEN bit in the corresponding chip select Configuration Register is clear

- Access to a non mapped address - access to an address that is not mapped to any CS.
- User access to a supervisor-protected chip select address space (the SP bit in the corresponding chip select Configuration Register is set)
- User access in fixed mode access
- User performs write access to write protected chip select
- First write data ID and write address ID do not match. (No data is written to the memory.)
- First Write Data ID and write address ID match but one or more of the other Write data IDs does not match the First Write data ID (data is written to memory according)
- Access duration to external device from CSx signal assertion is 128/256/512/1024 cycles (access is terminated by the controller) - This error can be disabled be software.

- IPS errors
  - User read or write access to a reserved/non-valid address in the EIM Configuration Register

21.5.14 DTACK Mode

In DTACK mode, the EIM uses DTACK signal as an indication of when to end the access.

DTACK is an asynchronous edge/level sensitive signal. DTACK polarity is configurable by the DAP bit in CsxGCR2 (default value is 0).

In this case, EIM begins the access and after a few cycles (according DAPS field) and waits until DTACK (after synchronization) becomes asserted, then samples the data in read access and completes the current data access (see Figure 21-15, Figure 21-16 & Figure 21-17).

If more than one data is needed, CS will be negated between access (CSREC field is not zero) and the AXI burst access will be split into single accesses (see Figure 21-19).

21.5.15 EIM_GRANT / EIM_BUSY Handshake Description

Prior to executing command to one of the external device (chip select), EIM assert EIM_BUSY signal (1'b1) and checks the EIM_GRANT signal status.

If EIM_GRANT signal is high, it indicates external data bus is not used by other slaves (NAND Flash Controller) and EIM may start to execute the access. If EIM_GRANT is low, EIM waits until it is set (1'b1) before executing the access.

EIM keeps EIM_BUSY signal set until it completes the access toward the external device.

Once EIM_GRANT signal is set, it can not be reset until EIM_BUSY signal is cleared by EIM.

NOTE

In 16-bit Muxed EIM doesn't use the data bus, therefore there is no sharing of the data bus with NFC. EIM doesn't wait for EIM_GRANT signal from NFC and doesn't assert the EIM_BUSY signal.

21.5.16 LPMD / LPACK Handshake Description

These signals are used for frequency and/or voltage change, and for entering low power mode during normal operation of the EIM. Before any change can take place, the controller and all the relevant external devices should be in idle state, which means no access or data transfer is in process.

LPMD input signal is asserted once EIM detects the assertion of LPMD, all ready signals of the AXI channels are negated, and EIM is not sampling new accesses. It finishes all the ongoing accesses and already pending ones. When EIM is in idle state, the LPACK output signal is asserted. EIM will stay in idle state and the LPACK signal will stay asserted until the LPMD signal is negated.

21.5.17 Endianness

Big and Little endianness are supported by the controller according to the following table.

Table 21-6. EIM Out/in Data in Case AXI Out/in Data is 0xB3B2B1B0

Endian mode	AXI access	AXI address [1:0]	Port size and used bits								
			Word port				Half word port			Byte port	
			[31:24]	[23:16]	[15:8]	[7:0]	External address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([23:16]) ([15:8]) ([7:0])
Big	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
							1	0xB1	0xB0	2	0xB1
										3	0xB0
	Half Word	0			0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
		2	0xB3	0xB2			1	0xB1	0xB0	2	0xB1
										3	0xB0
	Byte	0				0xB0	0		0xB3	0	0xB3
		1			0xB1			0xB2		1	0xB2
		2		0xB2			1		0xB1	2	0xB1
		3	0xB3					0xB0		3	0xB0
Little	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
							1	0xB3	0xB2	2	0xB2
										3	0xB3
	Half Word	0			0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
		2	0xB3	0xB2			1	0xB3	0xB2	2	0xB2
										3	0xB3
	Byte	0				0xB0	0		0xB0	0	0xB0
		1			0xB1			0xB1		1	0xB1
		2		0xB2			1		0xB2	2	0xB2
		3	0xB3					0xB3		3	0xB3

21.5.18 Strobe Signal Use

The strobe signal is toggling according to address/data valid condition on the external bus for read and write accesses, and for both synchronous and asynchronous modes.

At any time point when address/data is valid on the external bus, the strobe signal will generate a positive edge, which can be used to sample the external data and control signal.

NOTE

Strobe signal for read data is active (RL + 1) cycles after data on external bus is valid.

21.6 Initialization Information

21.6.1 Booting from EIM

EIM is ready to work with CS0 after the hardware reset, but it has been configured for very slow access (for boot purposes), with additional setup and hold time.

Other CSs are disabled by hardware reset. Therefore, all CSs must be properly initialized before use in writing values to the corresponding chip select configuration registers.

DSZ[1:0] and MUM fields are set according to EIM_BOOT [2:0] block inputs.

21.7 Typical Application

Application note uses following functions to illustrate EIM and memory accesses:

- WR16(address, data) is a 16 bit write access
- WR32(address, data) is a 32 bit write access
- RD16(address, data) is a 16 bit read access
- RD32(address, data) is a 32 bit read access
- WR_I(address, data, delta, counter) is a write data sequence, there data(i+1) = data(i) + delta
- COMMAND_SEQUENCE
- CHECK_STATUS

NOTE

COMMAND_SEQUENCE and CHECK_STATUS are described in AMD Flash Utility, Intel Sibley Flash Utility, MDOC Device Utility, Samsung OneNAND Utility, and Spansion Flash Utility.

All addresses are byte addresses. "CS0" is a Chip Select 0 base address. "EIM_" is a prefix of EIM's registers. 'h is a prefix of hexadecimal constant. "//" is a comment beginning. csba[cs] is a dimension of CS base addresses. "addr" means an address offset in current CS address space. Examples use CS0 address space, but it may apply to any CS except for boot mode functionality.

Configuration examples were verfied with the memory models listed below and may require some adjustments for other family members.

21.7.1 Access to Intel Sibley Flash

The following configurations are intended to Sibley family muxed and non-muxed devices.

21.7.1.1 Intel Sibley Flash Asynchronous Mode Configuration

- WR32('EIM_CS0GCR1,'h00210081);
- WR32('EIM_CS0RCR1,'h0e020000);
- WR32('EIM_CS0RCR2,'h00000000);
- WR32('EIM_CS0WCR1,'h0704a040);

21.7.1.2 Intel Sibley Flash Synchronous Mode Configuration

Configuration used for 133 MHz synchronous access to flash:

```
// Set memory to synchronous read mode
     WR16('CS0+('h5903<<1),'h0060);
     WR16('CS0+('h5903<<1),'h0003);
     WR16('CS0+('h0000<<1),'h00ff);
// Set EIM configuration to synchronous timing
     WR32('EIM_CS0GCR1,'h50214225);                    // 133 MHz
     WR32('EIM_CS0RCR1,'h0c000000);                    // 12 cycles on memory
```

Configuration used for 66 MHz synchronous access to muxed flash:

```
// Set memory to synchronous read mode
     WR16('CS0+('h3103<<1),'h0060);
     WR16('CS0+('h3103<<1),'h0003);
     WR16('CS0+('h0000<<1),'h00ff);
//-------------------------------------------------------------------------
// Set EIM configuration to synchronous timing
     WR32('EIM_CS0GCR1,'h5021122d);                    // 66 MHz
     WR32('EIM_CS0RCR1,'h07000000);                    // 7cycles on memory
```

21.7.1.3 Intel Sibley Flash Utility

```
// Single data word programming to addr
     WR16('CS0+addr,'h0060);                 // Unlock
     WR16('CS0+addr,'h00d0);
     WR16('CS0+addr, 'h0041);
     WR16('CS0+addr, data);
     WR16('CS0+caddr, 'h0070);                 // Read Status command
     while('CS0+data[7] == 0)                              // Wait / Polling
         RD16('CS0+addr, data);              // Read status
     RD16('CS0+addr, data);                  // Read status
```

```
    WR16('CS0+'h0000,'h00ff);
// Write buffer programming
    WR16('CS0+addr,'h0060);                 // Unlock
    WR16('CS0+addr,'h00d0);
    data    = 0;
    WR16('CS0+addr, 'h0070);                // Read Status command
    while(data[7] == 0)                             // Wait
        RD16('CS0+addr, data);              // Read status
    WR16('CS0+'h0000,'h00ff);
    WR16('CS0+addr,'h00e9);           // Write Buffer command
    WR16('CS0+addr,255);              // Word counter (<256)
    for(i=0; i<'h200; i = i + 'h40)
       WR_I('CS0+addr+i, data+((i>2)*'h0010_0001), 'h0010_0001, 16); // Data
    WR16('CS0+addr,'h00d0);           // Write Confirm command
    data  = 0;
    while(data[7] == 0)                      // Wait
       RD16('CS0+addr, data);         // Read status
    RD16('CS0+addr, data);            // Read status
    WR16('CS0+'h0000,'h00ff);
```

21.7.2 Access to MDOC Device

The following configurations are intended to MDOC H3 device.

21.7.2.1 MDOC Device Boot

To boot from the MDOC device the ERRST bit should be configured to 1, so that EIM will hold the first read access to CS0 until the MDOC asserts the RDY signal.

21.7.2.2 MDOC Device Asynchronous Mode Configuration

```
// Non-muxed mode
WR32('EIM_CS0GCR1,'h00410081);
WR32('EIM_CS0RCR1,'h0e121010);
WR32('EIM_CS0RCR2,'h00000000);
WR32('EIM_CS0WCR1,'h12092492);
// Muxed mode
WR32('EIM_CS0GCR1,'h00410081);
WR32('EIM_CS0RCR1,'h0e121010);
WR32('EIM_CS0RCR2,'h00000000);
WR32('EIM_CS0WCR1,'h12092492);
```

21.7.2.3 MDOC Device Utility

```
// Read Manufacturer ID and Device ID
    RE16('CS0+'h9400,'h4833);
    RE16('CS0+'h9422,'hb7cc);
```

21.7.3 Access to Micron PSRAM

The following configurations are intended to mt45w4mw16bfb_706.

21.7.3.1 Micron PSRAM Asynchronous Mode Configuration

```
// 16 bit memory
    WR32('EIM_CS0GCR1,'h403104b1);
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h0b040040);
// 32 bit memory
    WR32('EIM_CS0GCR1,'h403304b1);
    WR32('EIM_CS0RCR1,'h0f010000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h0f040040);
```

21.7.3.2 Micron PSRAM Synchronous Mode Configuration

```
// 16 bit memory
    WR32('EIM_CS0GCR1,'h403104b1);
    WR32('EIM_CS0WCR1,'h0b040000);
    WR16('CS0+('h85947<<1),'h0040); // memory configuration
    WR32('EIM_CS0GCR1,'h4021_5487);// fixed latency memory wrap 4
    WR32('EIM_CS0RCR1,'h04000000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h04000000);
// 32 bit memory
    WR32('EIM_CS0GCR1,'h6003_04f1);
    WR32('EIM_CS0WCR1,'h0b04_0000);
    WR32('CS0+('h85947<<2),'h0040); // memory configuration
    WR32('EIM_CS0GCR1,'h4003_1487); // var latency memory inc. page size 128
    WR32('EIM_CS0RCR1,'h04000000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h04000000);
```

21.7.4 Access to Samsung OneNAND

Mentioned below are the configurations intended for Samsung OneNAND muxed and non-muxed devices.

21.7.4.1 Samsung OneNAND Boot

There are two ways to boot from Samsung OneNAND. In the first way, the ERRST bit is set to 0 and the user has to poll the interrupt status in the OneNAND interrupt register (or set interrupt handler there). In the second way, the ERRST bit is set to 1 and the user should enable the device interrupt output before the first read from CS0 access is issued.

Load sectors 2,3 to DataRAM, page 0 done in the next example:

- WR16('CS0+('hF241<<1),'h0); // Clear interrupt status
- WR16('CS0+('hF100<<1),'h0); // block[8:0] address
- WR16('CS0+('hF107<<1),'h2); // sector[1:0] and page[7:2] addresses
- WR16('CS0+('hF200<<1),'h802); // buffer[11:8] address and counter[1:0]
- WR16('CS0+('hF101<<1),'h0); // DDP choose
- WR16('CS0+('hF220<<1),'h0); // Set command

21.7.4.2 Samsung OneNAND Asynchronous Mode Configuration

```
// Non-muxed memory
    WR32('EIM_CS0GCR1,'h00410081);
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000000);
    WR32('EIM_CS0WCR1,'h0c092480);
// Muxed memory
    WR32('EIM_CS0GCR1,'h00410089);
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000000);
    WR32('EIM_CS0WCR1,'h0c092480);
```

21.7.4.3 Samsung OneNAND Synchronous Mode Configuration

Set memory and EIM to synchronous read mode is shown in the next example:

```
    WR16('CS0+('hF221<<1),'hc0e0); // Synchronous read, 4 clk latency
    WR32('EIM_CS0GCR1,'h50412405); // 44 MHz (non-muxed)
    WR32('EIM_CS0RCR1,'h05010000);
```

The muxed Samsung OneNAND supports synchronous write, too:

```
// Set memory & EIM to synchronous read and write mode
    WR16('CS0+('hF221<<1),'hc0f2); // Sync. read and write, 4 clk latency
    WR32('EIM_CS0GCR1,'h5041240f); // 44 MHz
    WR32('EIM_CS0RCR1,'h05010000);
    WR32('EIM_CS0WCR1,'h05040000);
```

21.7.4.4 Samsung OneNAND Utility

The following utility algorithms are used on the Samsung OneNAND:

```
// Unlock Block command
    WR16('CS0+('hF100<<1),'h0);        // DFS
    WR16('CS0+('hF100<<1),'h0);        // DBS
    WR16('CS0+('hF24c<<1),'h2);        // SBA - block number (2)
    WR16('CS0+('hF241<<1),'h0);        // Clear interrupt status
    WR16('CS0+('hF220<<1),'h23);       // Unlock command
    data ='h0;
    while(!(data &'h0004))             // Polling
       RD32('WIAR, data); // Read status
// Erase block command
    WR16('CS0+('hF100<<1),'h2);        // DFS and block ([8:0]) address
    WR16('CS0+('hF101<<1),'h0);        // DBS
    WR16('CS0+('hF241<<1),'h0);        // Clear interrupt status
```

```
                    WR16('CS0+('hF220<<1),'h94);            // Erase command
          data ='h0;
          while(!(data &'h0004))                    // Wait
              RD32('WIAR, data); // Read status
     // Program page command
          WR16('CS0+('hF100<<1),'h2);               // DFS and block[8:0] address
          WR16('CS0+('hF107<<1),'h0);               // sector[1:0] and page[7:2] addresses
          WR16('CS0+('hF200<<1),'h800);             // buffer[11:8] address and counter[1:0]
          WR16('CS0+('hF241<<1),'h0);               // Clear interrupt status
          WR16('CS0+('hF220<<1),'h80);              // Program command
          data ='h0;
          while(!(data &'h0004))                    // Wait
              RD32('WIAR, data); // Read status
```

21.7.5 Access to Samsung UtRAM

Below mentioned configurations are intended for Samsung UtRAM.

21.7.5.1 Samsung UtRAM Asynchronous Mode Configuration

```
          WR32('EIM_CS0GCR1,'h400104b1);
          WR32('EIM_CS0RCR1,'h0a010000);
          WR32('EIM_CS0RCR2,'h00000008);
          WR32('EIM_CS0WCR1,'h0b040040);
```

21.7.5.2 Samsung UtRAM Synchronous Mode Configuration

```
          RD16('CS0+('hff_ffff<<1),data);             // command sequence
          RD16('CS0+('hff_ffff<<1),data);
          RD16('CS0+('hff_ffff<<1),data);
          RD16('CS0+('hff_feff<<1),data);
          RD16('CS0+('h00_82a0<<1),data);             // memory sync. configuration
          WR32('EIM_CS0GCR1,'h4021_53b7);            // fixed latency memory wrap 32
          WR32('EIM_CS0RCR1,'h0500_0000);
          WR32('EIM_CS0WCR1,'h0300_0000);
```

21.7.6 Access to Spansion Flash

Below mentioned configurations are intended for Spansion Flash.

21.7.6.1 Spansion Flash Asynchronous Mode Configuration

```
          WR32('EIM_CS0GCR1,'h00410081);
          WR32('EIM_CS0RCR1,'h0a018000);
          WR32('EIM_CS0RCR2,'h00000000);
          WR32('EIM_CS0WCR1,'h0704a240);
WR16('CS0+('hF220<<1),'h94);            // Erase command
          data ='h0;
          while(!(data &'h0004))                    // Wait
              RD32('WIAR, data); // Read status
     // Program page command
```

```
    WR16('CS0+('hF100<<1),'h2);             // DFS and block[8:0] address
    WR16('CS0+('hF107<<1),'h0);             // sector[1:0] and page[7:2] addresses
    WR16('CS0+('hF200<<1),'h800);           // buffer[11:8] address and counter[1:0]
    WR16('CS0+('hF241<<1),'h0);             // Clear interrupt status
    WR16('CS0+('hF220<<1),'h80);            // Program command
    data ='h0;
    while(!(data &'h0004))                  // Wait
        RD32('WIAR, data); // Read status
```

21.7.6.2 Spansion Flash Synchronous Mode Configuration

```
    WR16('CS0+('h0555<<1),'h00aa);   // command sequence
    WR16('CS0+('h02aa<<1),'h0055);
    WR16('CS0+('h0555<<1),'hd0);
    WR16('CS0+('h0000<<1),'h1ec4);   // memory sync. configuration
    WR32('EIM_CS0GCR1,'h50411325); // 66 MHz
    WR32('EIM_CS0RCR1,'h05000000); // 5 cycles on memory
```

21.7.6.3 Spansion Flash Utility

```
// Single word programming
    COMMAND_SEQUENCE(cs,16,'ha0);           // single word programming
    WR16('CS0+addr, data);
    CHECK_STATUS('CS0+addr,data,16,1,errst);
// Write buffer programming
    COMMAND_SEQUENCE(0,16,'h25);              // write buffer programming
    WR16('CS0+addr,'h001f);                     // counter-1
    WR_I('CS0+addr, data, 'h0010_0001, 16); // data
    WR16('CS0+addr,'h0029);                     // write buffer to flash
    CHECK_STATUS('CS0+addr+'h3e,data[31:16]+'h00f0,16,1,errst);
```

There COMMAND_SEQUENCE and CHECK_STATUS are next functions:

```
task COMMAND_SEQUENCE;
    input[2:0]    cs;
    input[7:0]    port_size;
    input[31:0]   code;
begin
    if(port_size == 16)
       begin
       WR16(csba[cs]+('h0555<<1),'h00aa);
       WR16(csba[cs]+('h02aa<<1),'h0055);
       WR16(csba[cs]+('h0555<<1),code);
       end
    else
       begin
       WR32(csba[cs]+('h0555<<2),'h00aa);
       WR32(csba[cs]+('h02aa<<2),'h0055);
       WR32(csba[cs]+('h0555<<2),code);
       end
end
endtask
task     CHECK_STATUS;
    input[31:0] addr;
    input[31:0] edata;
    input[7:0]   port_size;
    input[7:0]   opcode;
    output[7:0] errst;
    reg[31:0]    data;
    reg[31:0]    data3;
begin
    errst = 0;
```

```
           data  = 0;
           data3 = 0;
while(!(data == edata) && !errst) // Wait operation
         begin: BR_EN
         RD16(addr, data);             // Read status
         if(data[7] !== edata[7])
             begin
             if(data[5] == 1)
                  begin
                  RD16(addr, data3);
                  RD16(addr, data);
                  if(data[6] !== data3[6])
                       begin
                       $display("CHECK_STATUS: Error timout on single data program");
                       errst = 1;
                       disable BR_EN;
                       end
                  end
             else
                  begin
                  if(opcode == 2)
                       if(data[1] == 1)
                           begin
                           RD16(addr, data3);
                           if(port_size == 32)
                              RD32(addr, data);
                           else
                              RD16(addr, data);
                           if(data[1] == 1 && data !== edata)
                              begin
                              $display("CHECK_STATUS: Error on write buffer");
                              errst =3;
                              disable BR_EN;
                              end
                           end
                  end
             end
         else
             begin
             RD16(addr, data3);
             if(port_size == 32)
                  RD32(addr, data);
             else
                  begin
                  RD16(addr, data);
                  edata[31:16] = 16'h0;
                  end
             if(data !== edata)
                  begin
                  $display("CHECK_STATUS: Error in data write on single data program");
                  errst =2;
                  disable BR_EN;
                  end
             end
         end
end
endtask
```

21.7.7 8 bit support

This section details the pin connections for Intel mode and Motorola mode.

Intel Mode - For intel mode use the following connection:

Table 21-7. Intel Mode pin connections

ARM platform Pin	EIM Pin	Notes
ADS#	IPP_DO_ADV_B	WAL = 1,RAL = 1
W/R	IPP_DO_BE_B	WBED = 1
WR#	WE#	
RD#	OE#	

Mot. Mode - For intel mode use the following connection:

Table 21-8. Motorola Mode pin connections

ARM platform Pin	EIM Pin	Notes
AS#	IPP_DO_CS_B	
R/W#	WE#	
LDS#	BE#	

21.8 External Bus Timing Diagrams

The following timing diagrams show the timing of accesses to memory or a peripheral with different timing parameters. All examples done for CS0, but are valid for any others chip select. BE means one from current used BE[3:0].

21.8.1 Asynchronous Read Memory Accesses Timing Diagram

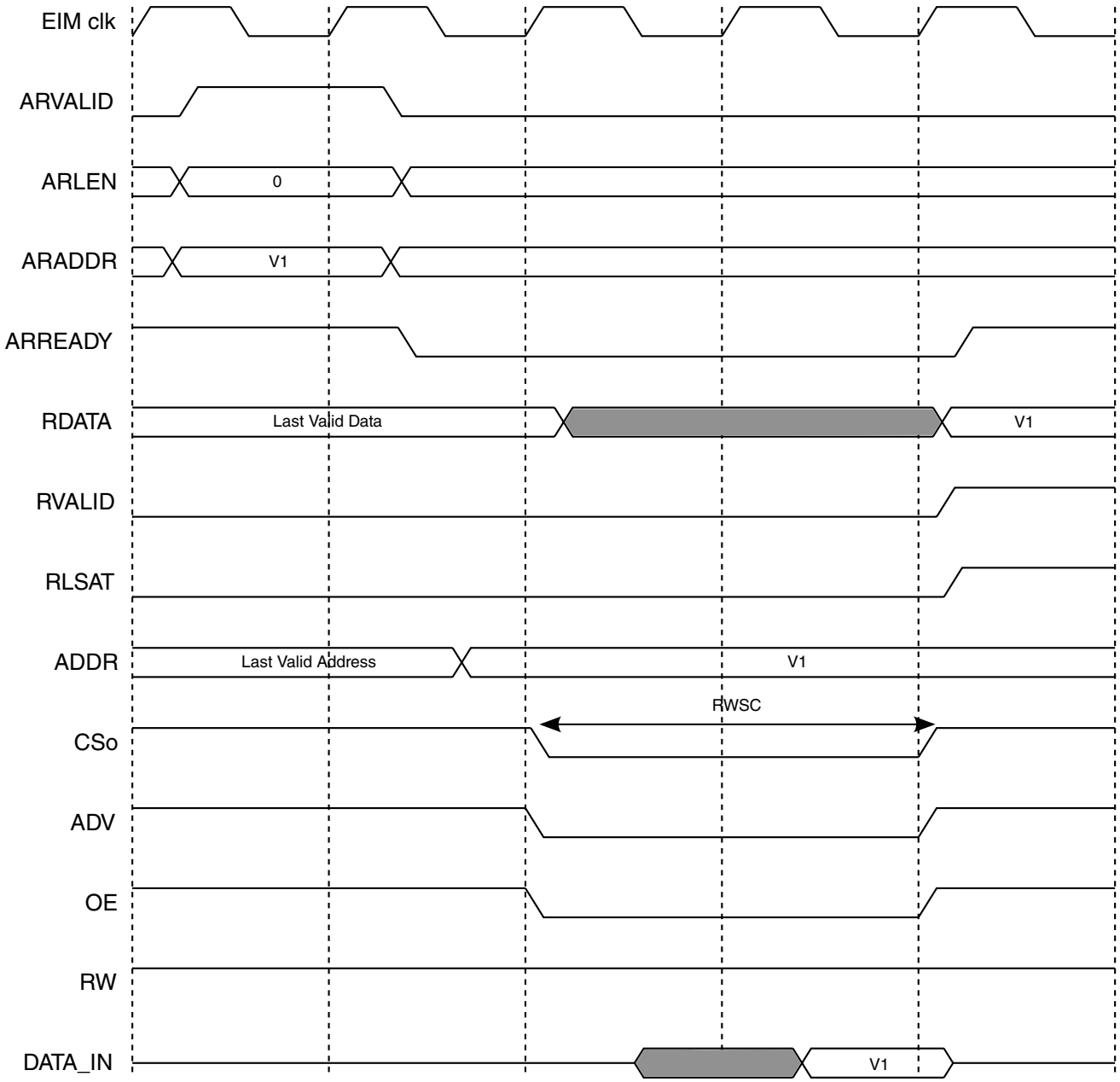


Figure 21-2. Read Access, RWSC=2,RCSA=0,OEA=0,RCSN=0,OEN=0, RAL=1

21.8.2 Asynchronous Write Memory Accesses Timing Diagram

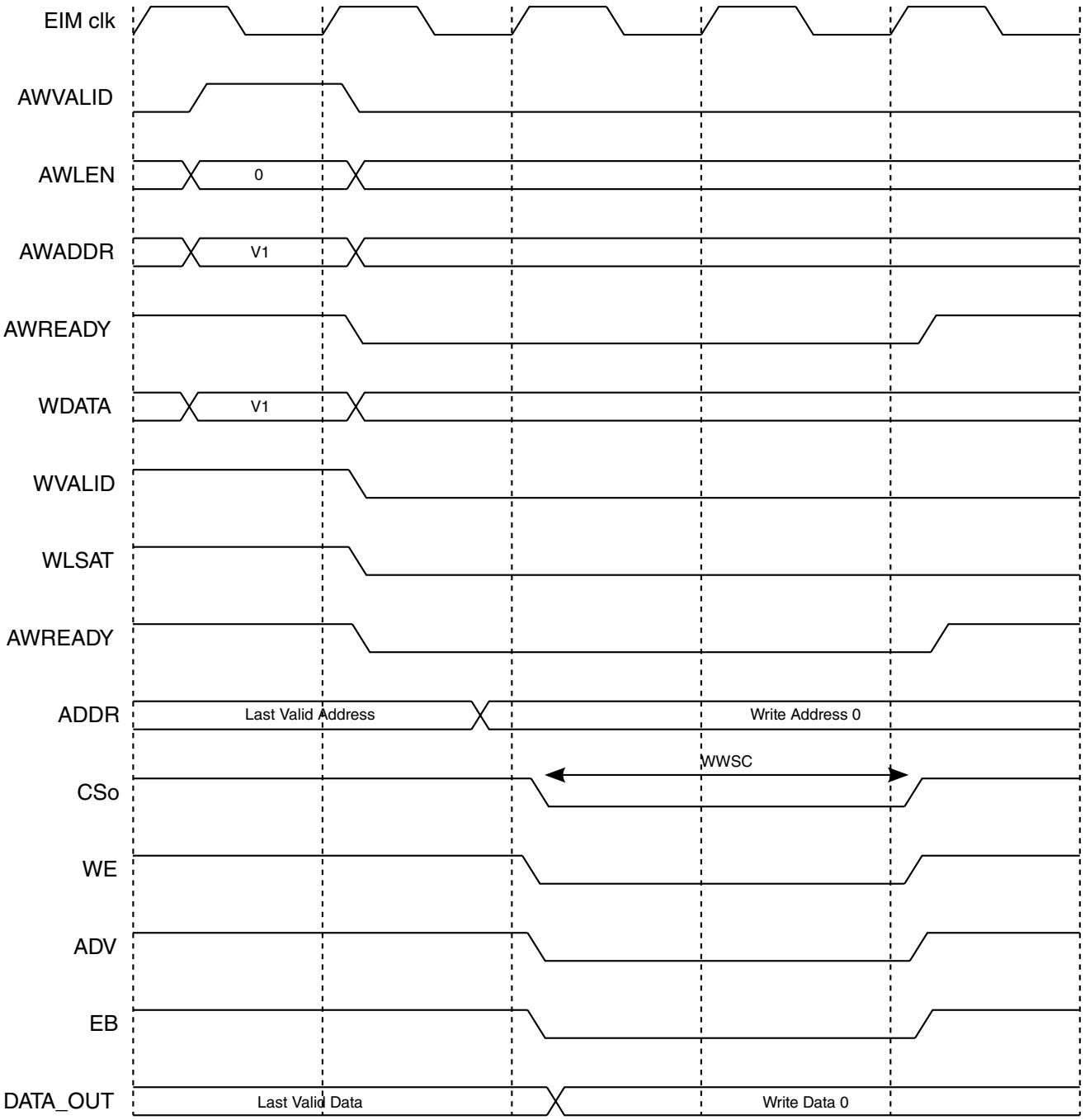


Figure 21-3. Write Access, WWSC=2,WCSA=0,WEA=0,WCSN=0,WEN=0,BEA=0, BEN=0, WAL=1

21.8.3 Asynchronous Read/Write Memory Accesses Timing Diagram

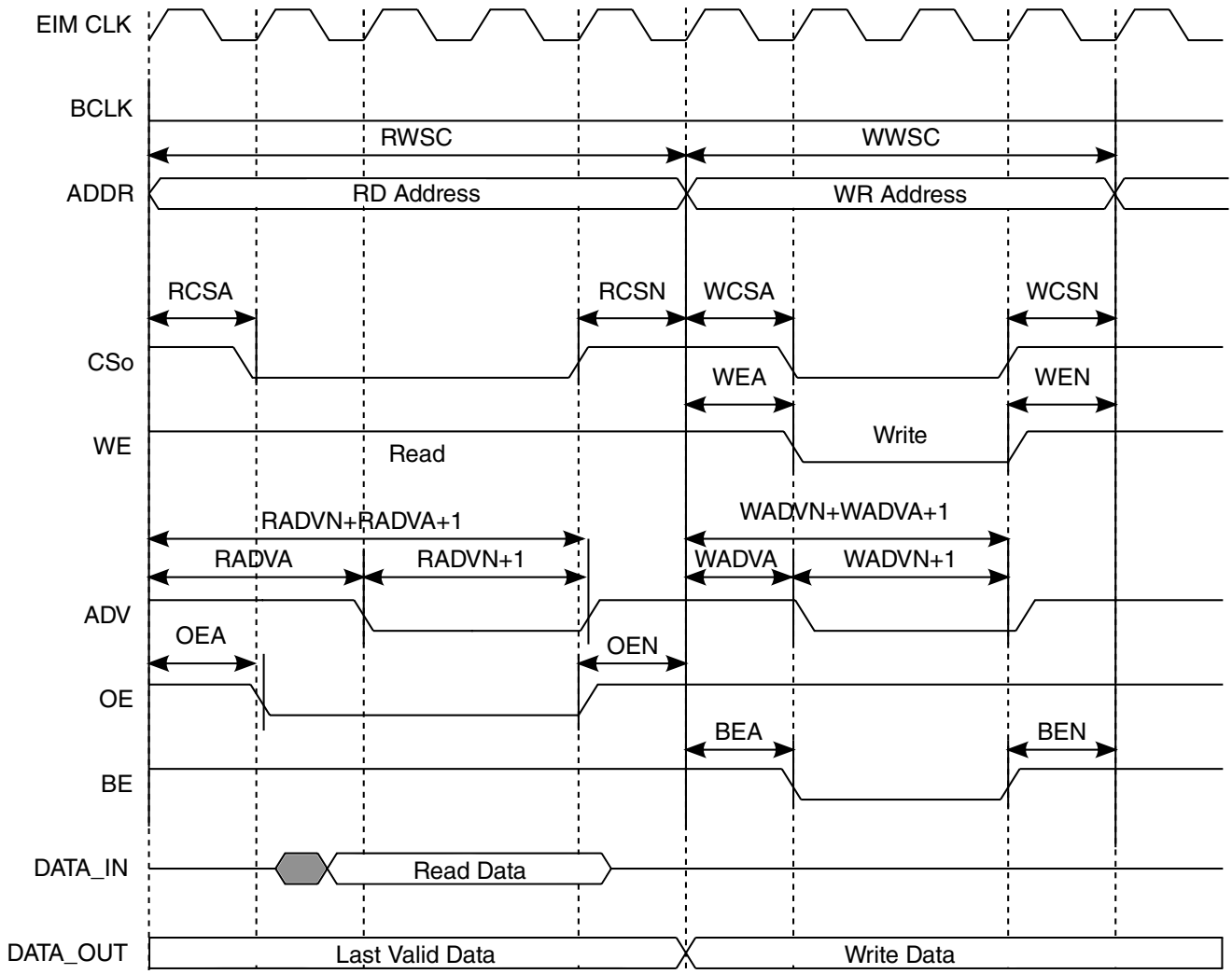


Figure 21-4.
RCSA=1,RADVA=2,OEA=1,RADVN=1,RCSN=1,OEN=1,WCSA=1,WEA=1,WADVA=1,BEA=1,WADVN=1,WCSN=1,WEN=1,BEN=1

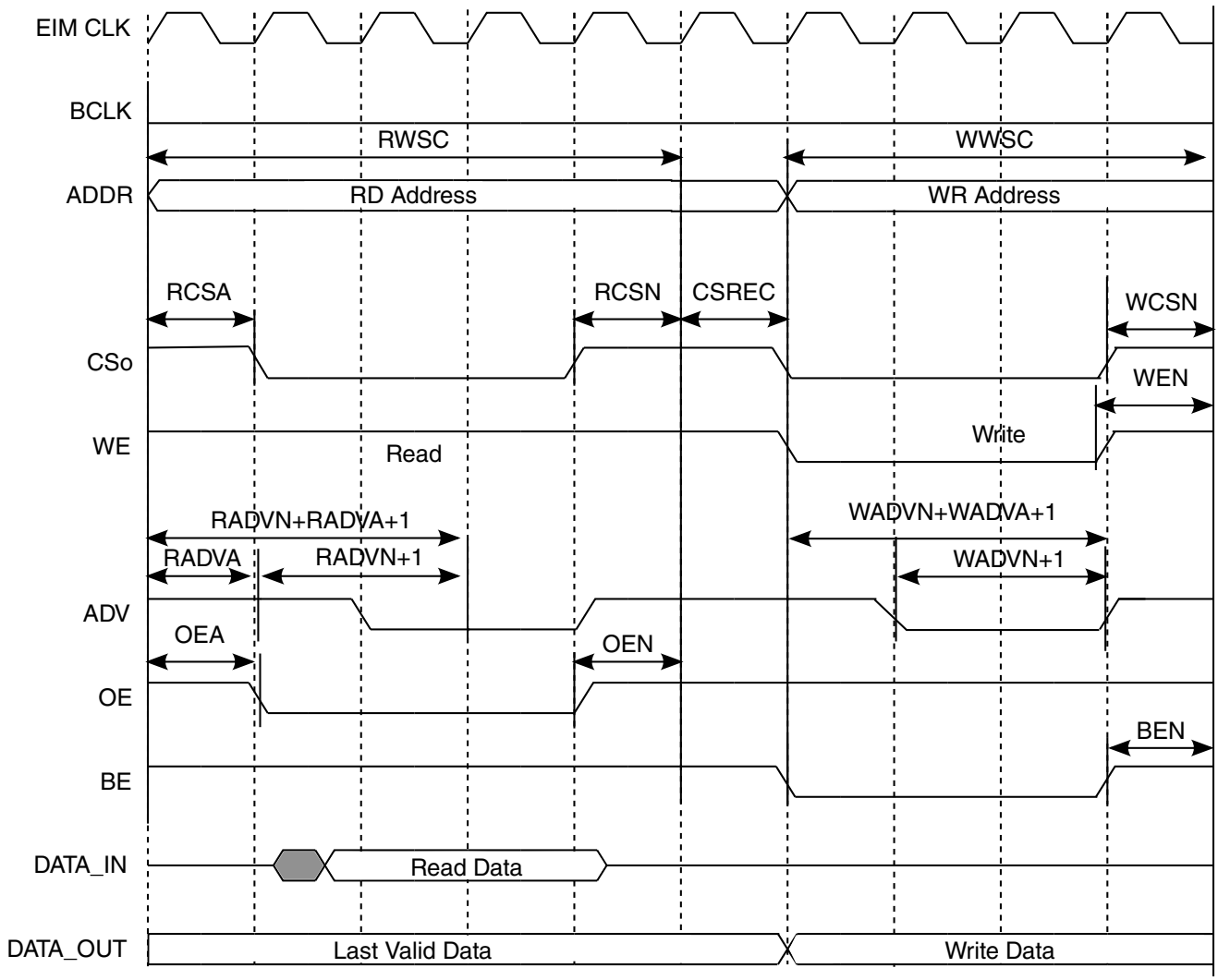


Figure 21-5.
RWSC=5,RCSA=1,RCSN=1,RADVA=1,RADVN=1,OEA=1,OEN=1,WWSC=4,WCSA=0,WCSN=1,WEA=0,WEN=1,WADVA=1,WADVN=1,BEA=0,BEN=1,CSREC=1

21.8.4 Asynchronous Read/Write Using RAL, WAL and CSREC

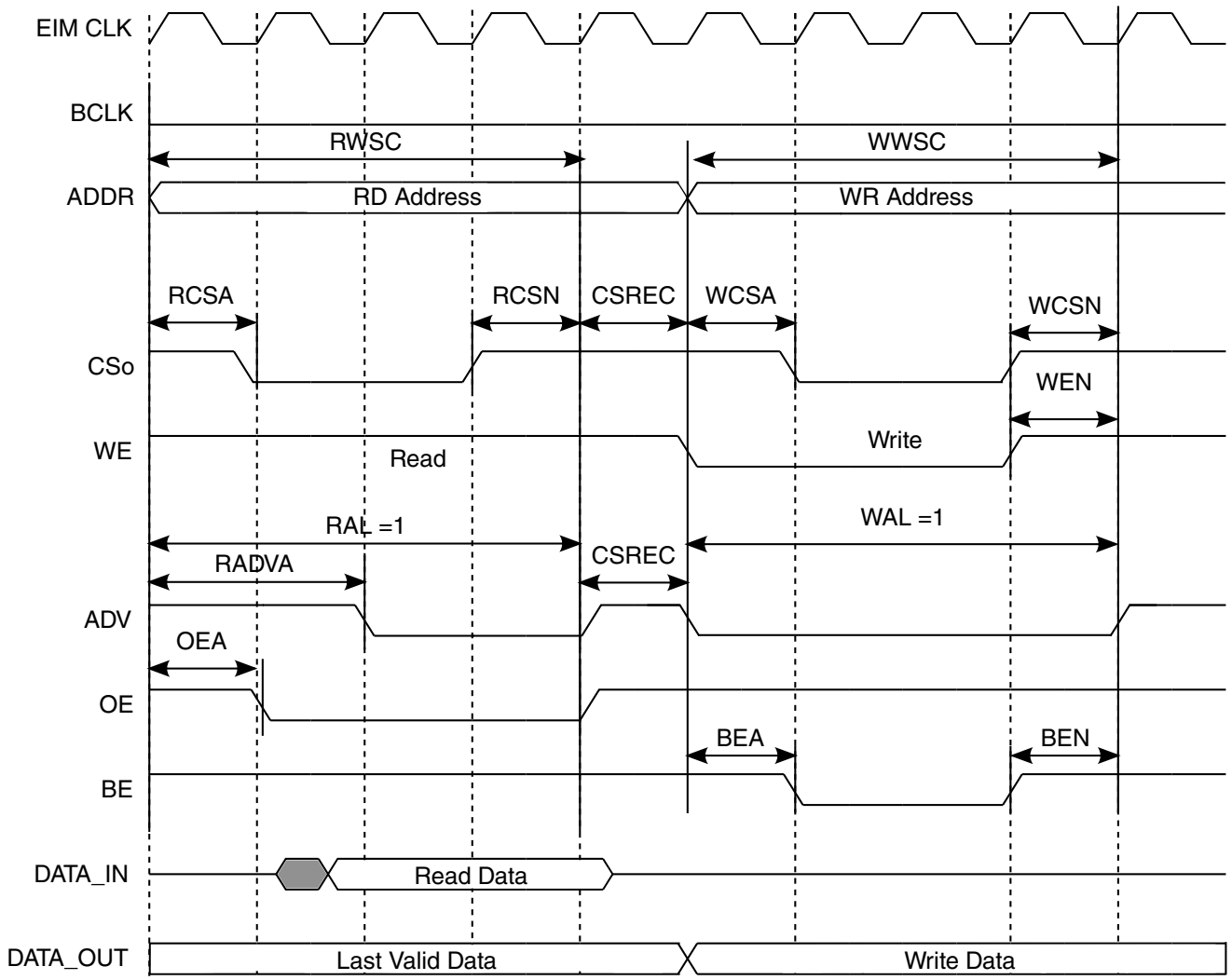


Figure 21-6.
RAL=1,RCSN=1,RADVA=2,OEA=1,RCSN=1,CSREC=1,WCSA=1,WEA=0,WADVA=0,BEA=1,WAL=1,WCSN=1,WEN=1,BEN=1

21.8.5 Consecutive Asynchronous Write Memory Accesses Timing Diagram

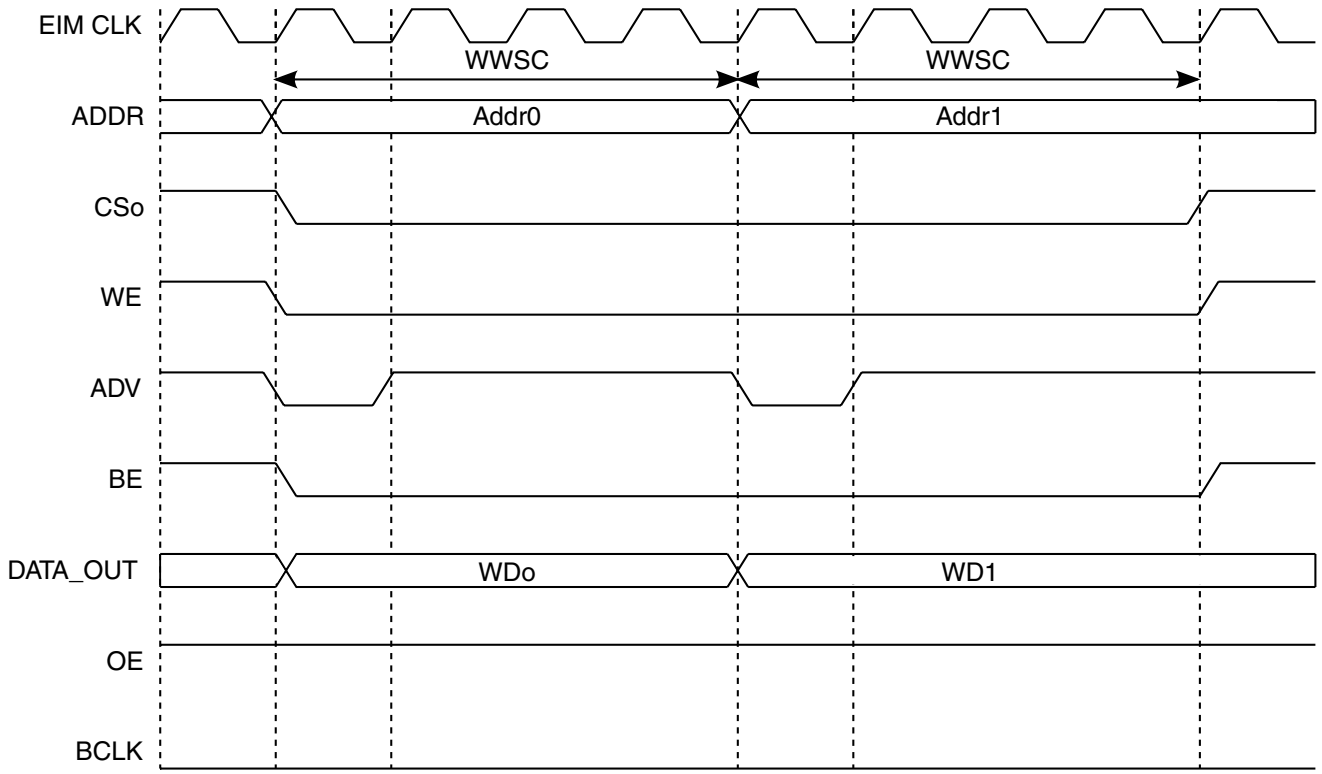


Figure 21-7. WWSC=4,WCSA=0,WEA=0,WADVA=0,BEA=0,WCSN=0,WEN=0,WADVN=0,BEN=0,CSREC=0

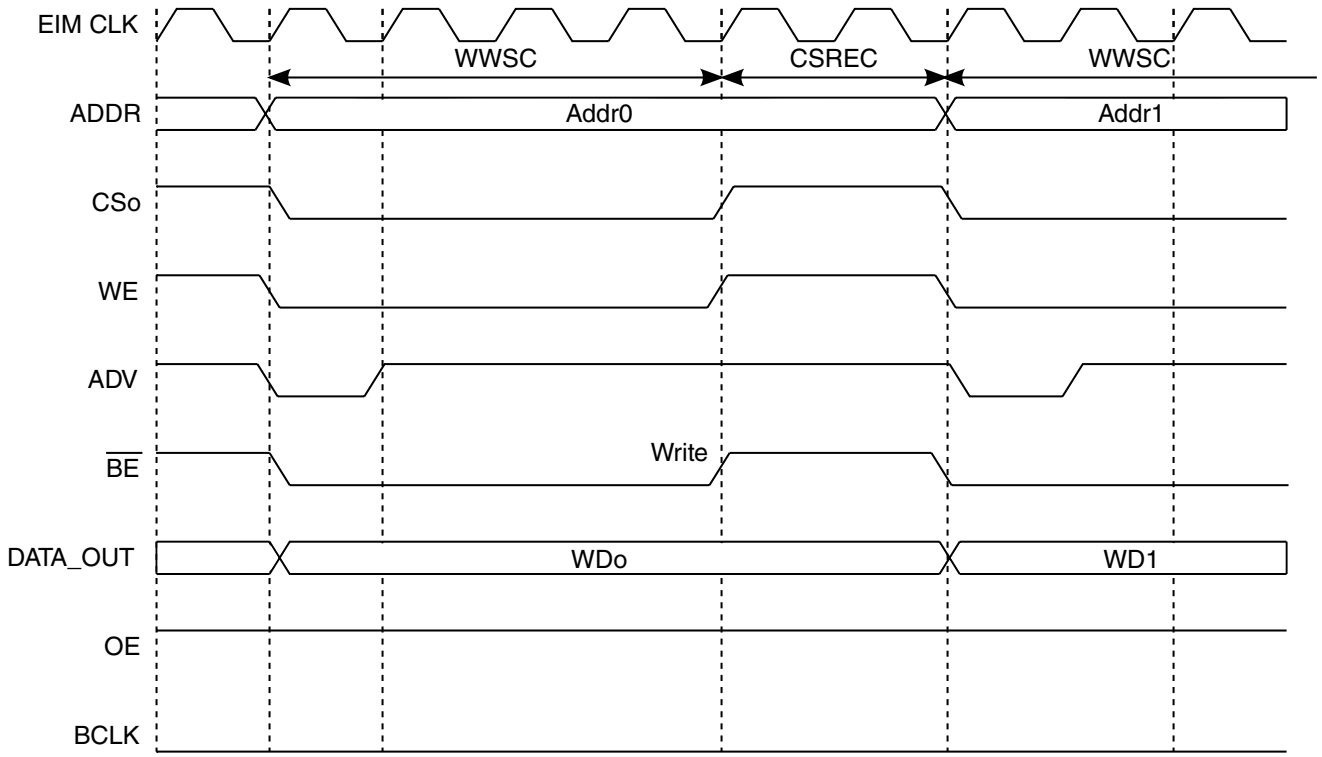


Figure 21-8. WWSC=4,WCSA=0,WEA=0,WADVA=0,BEA=0,WCSN=0,WEN=0,WADVN=0,BEN=0,CSREC=2

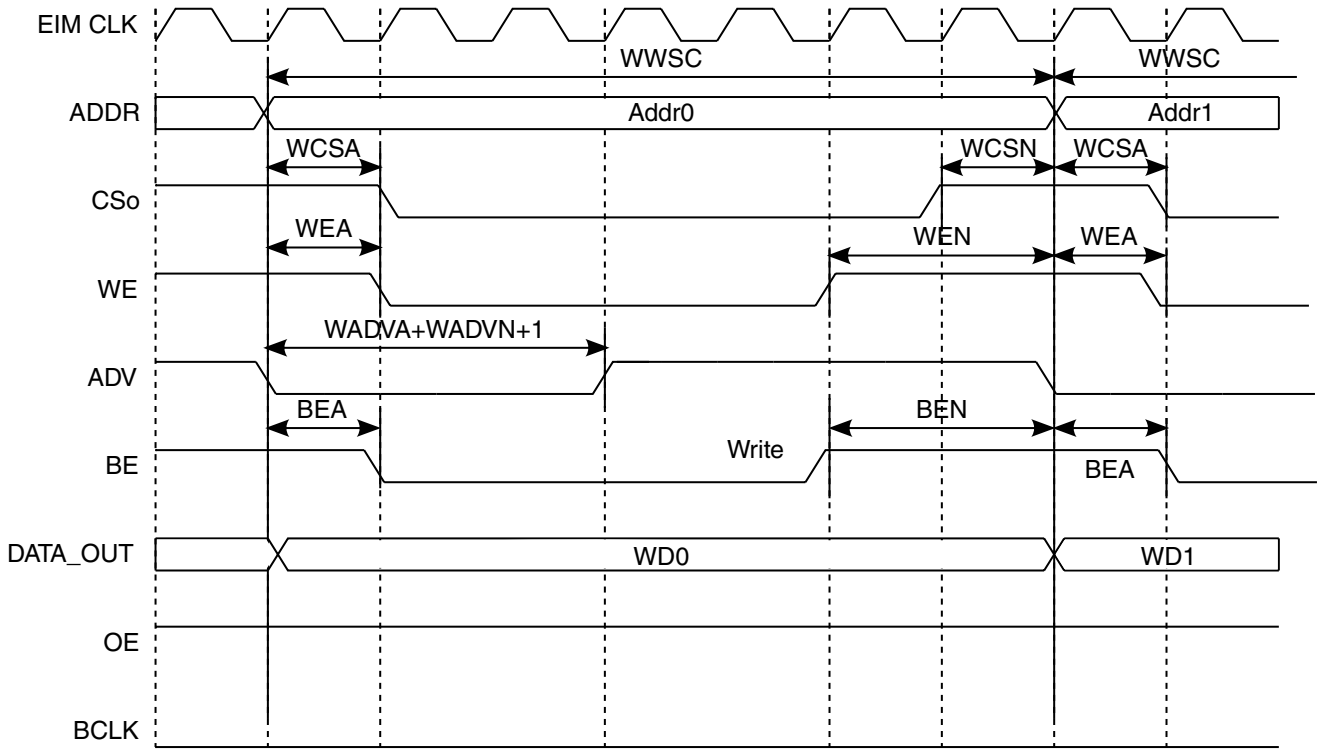


Figure 21-9.
WWSC=7,WCSA=1,WCSN=1,WEA=1,WEN=2,WADVA=0,WADVN=2,BEA=1,BEN=2

21.8.6 Consecutive Asynchronous Read Memory Accesses Timing Diagram

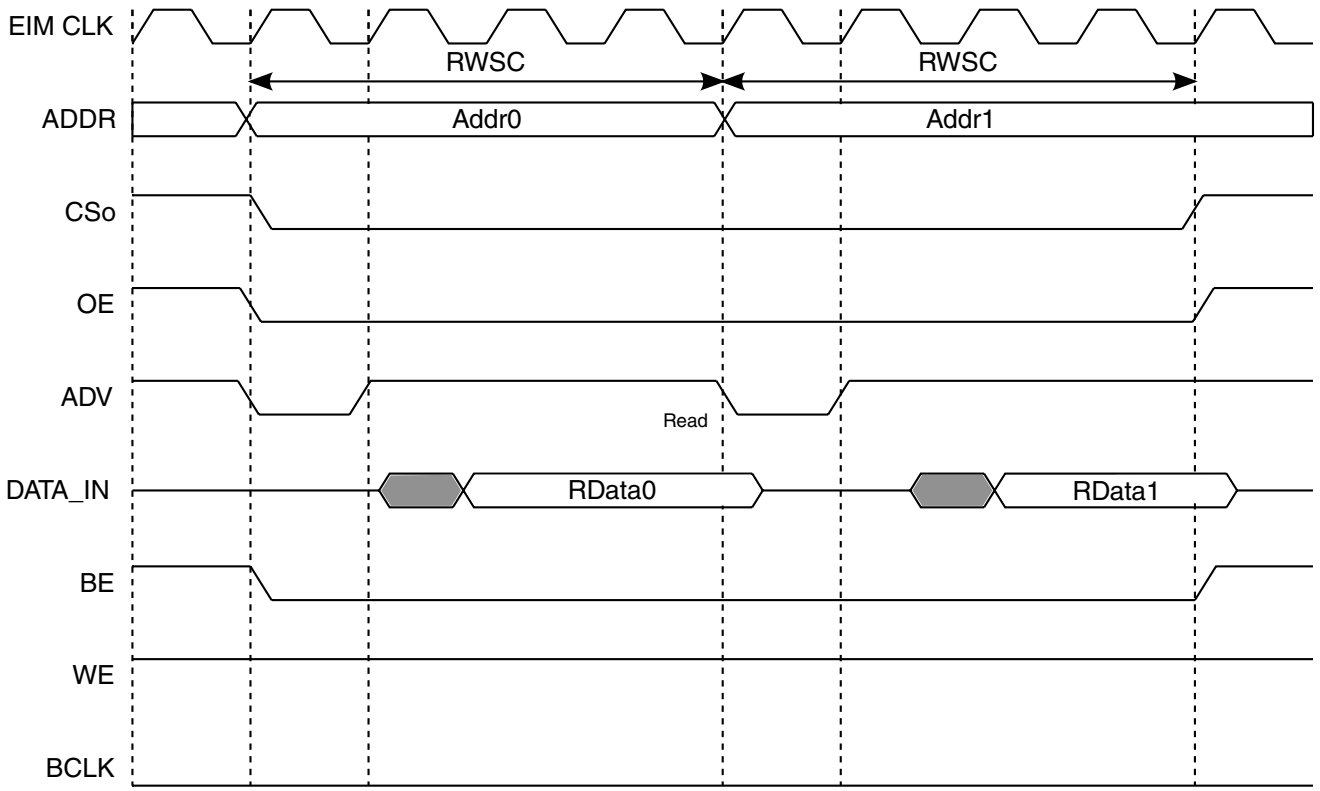


Figure 21-10. RWSC=4,RCSA=0,OEA=0,RADVA=0,RCSN=0,OEN=0,RADVN=0,CSREC=0

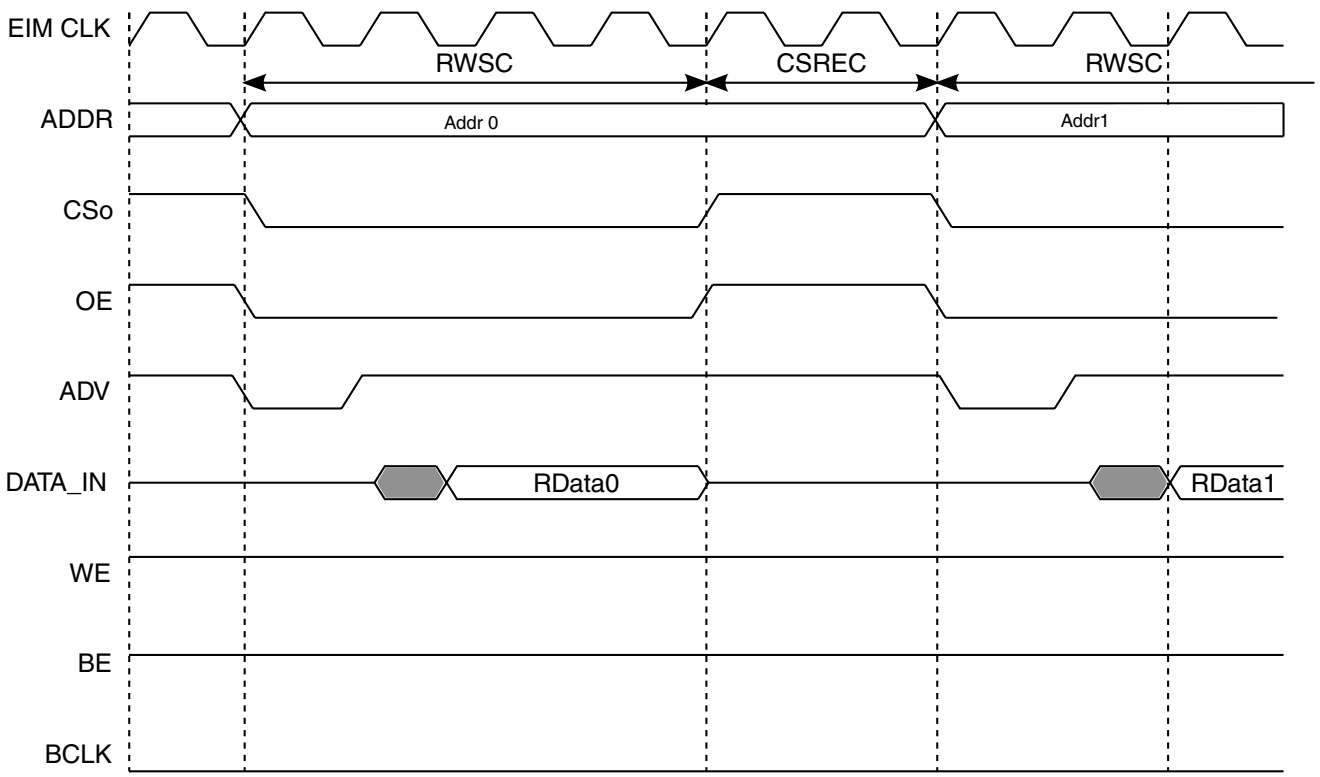


Figure 21-11. RWSC=4,RCSA=0,OEA=0,RADVA=0,RCSN=0,OEN=0,RADVN=0,CSREC=2

21.8.7 Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=0

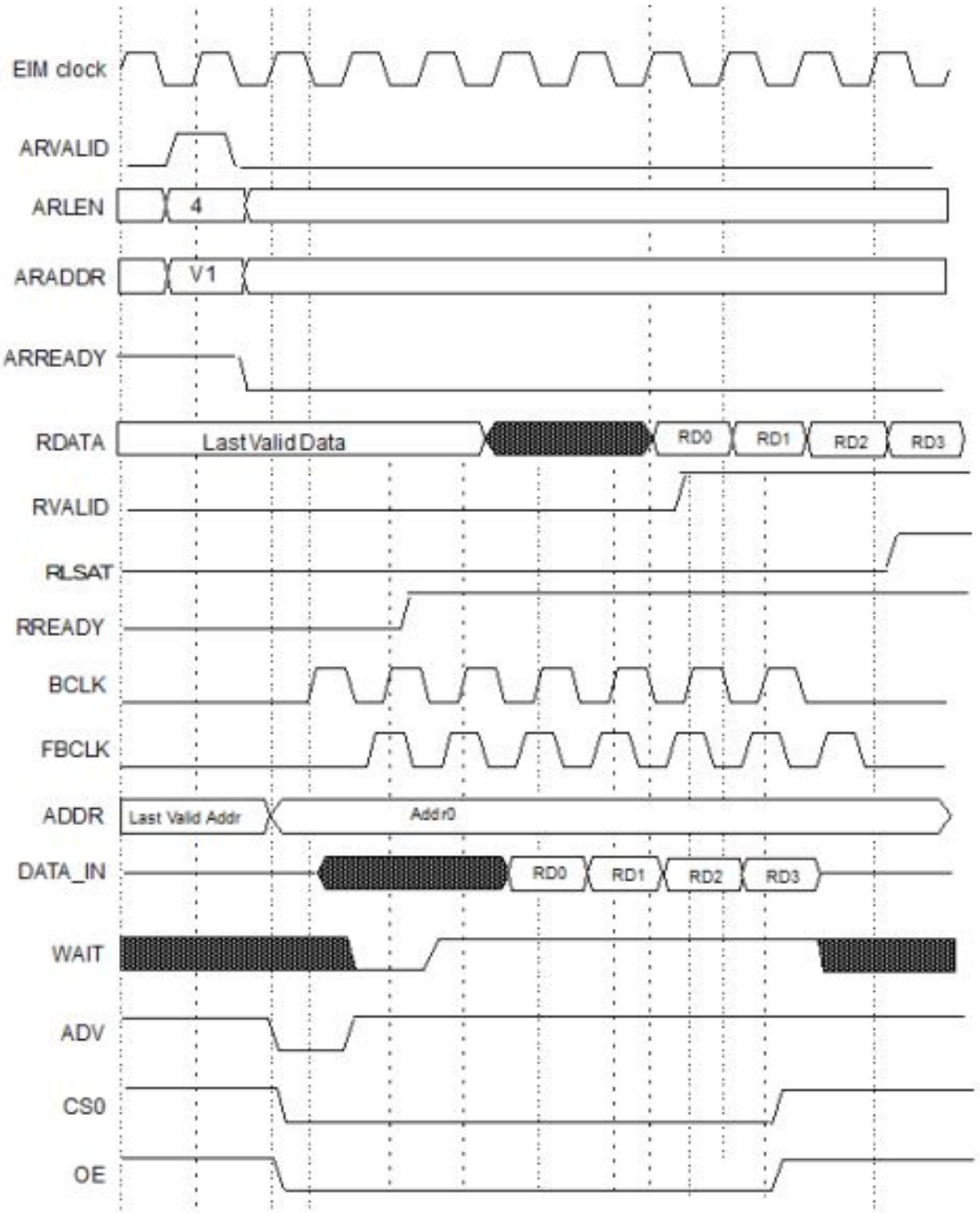


Figure 21-12. SRD=1,BCD=0,BCS=0,RWSC=1,RADVA=0,RADVN=0,RFL=0,RL=0

21.8.8 Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=1

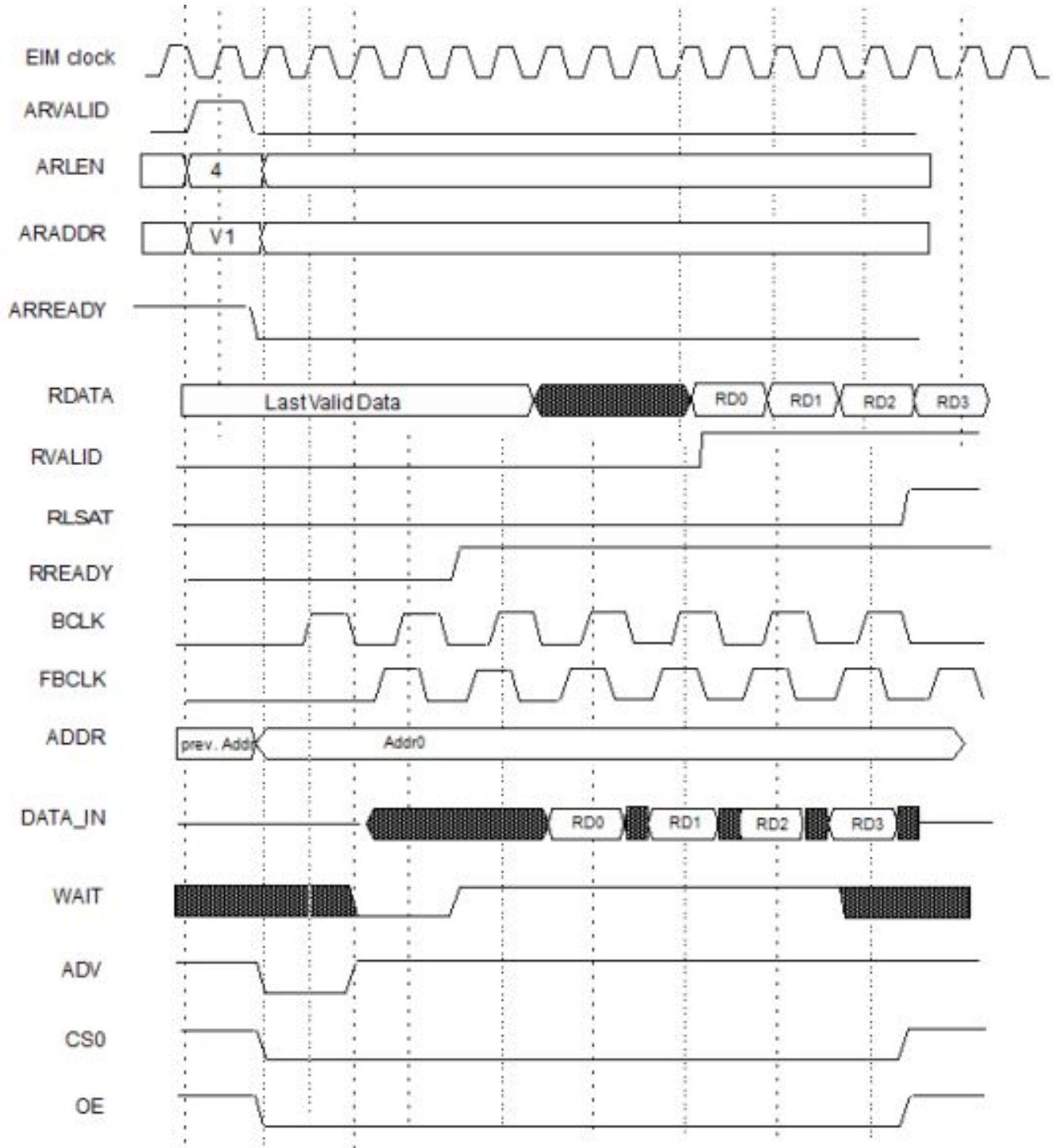
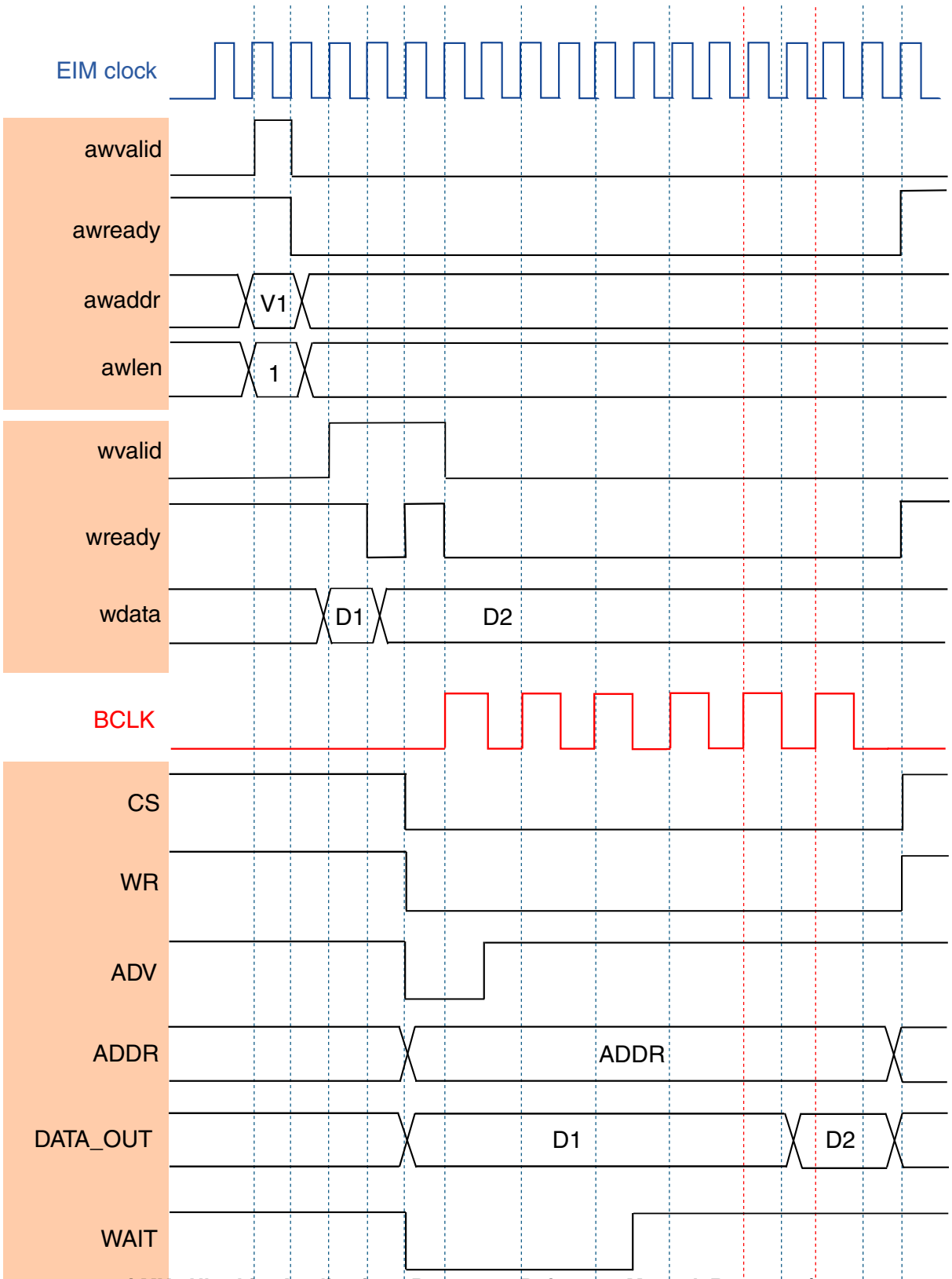


Figure 21-13. BCD=1, RL = 3

21.8.9 Burst (Synchronous Mode) Write Memory Access Timing - BCD=1



21.8.10 Asynchronous Page Mode Access

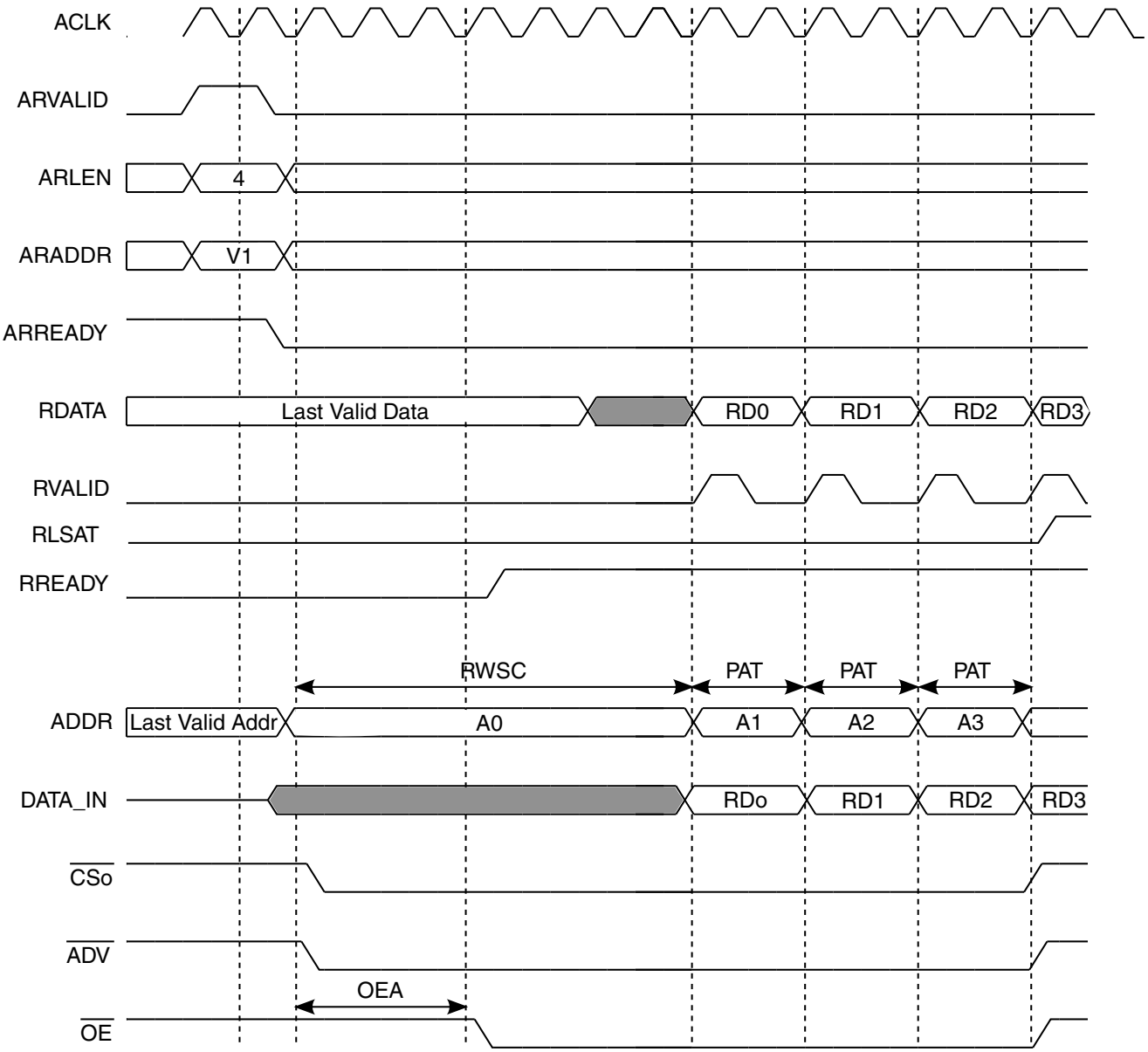


Figure 21-14. PAT = 2

21.8.11 DTACK Mode - AXI Single Access

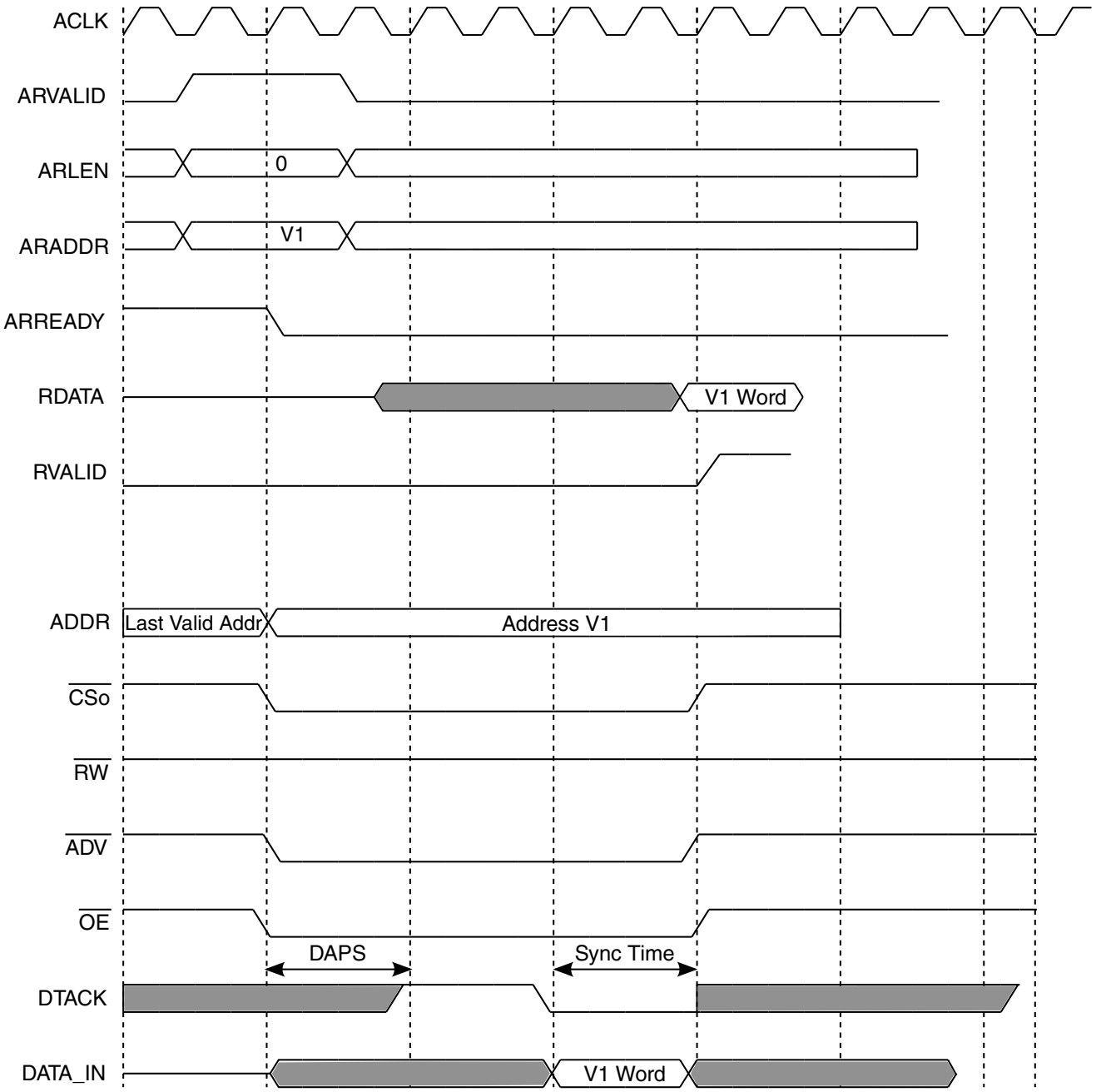


Figure 21-15. DAPS = 2

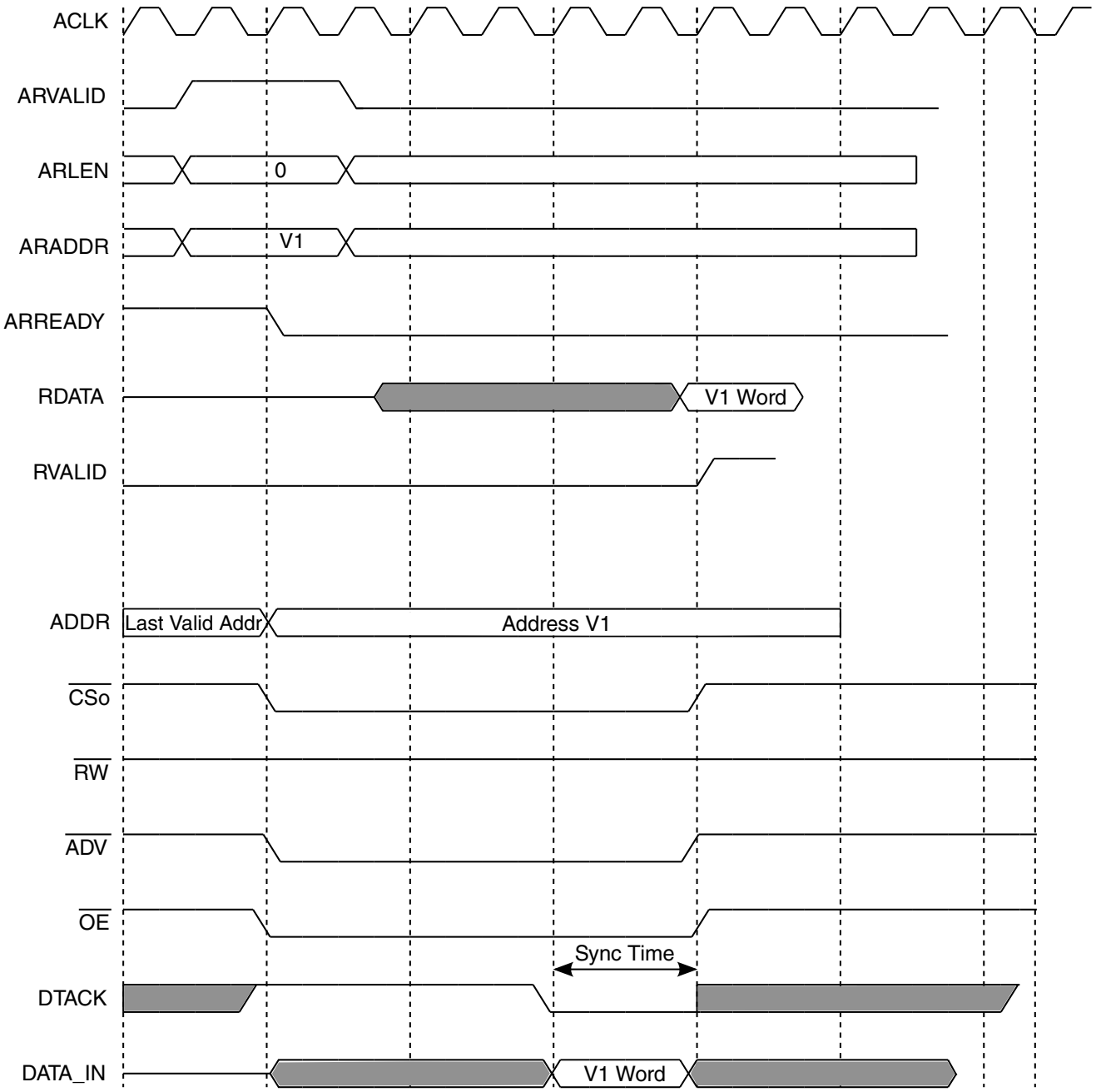


Figure 21-16. DAPS = 0

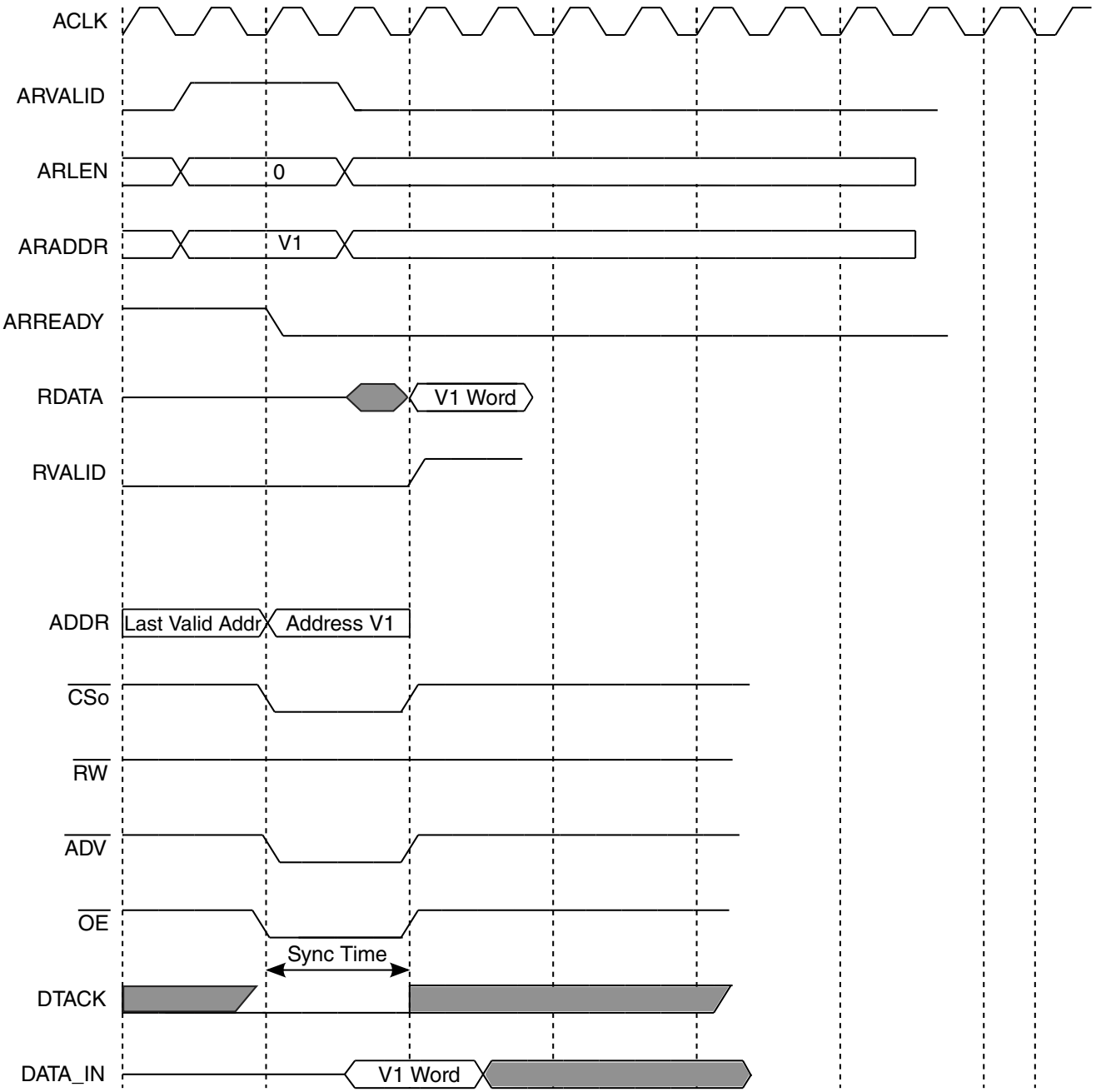


Figure 21-17. DAPS = 0

21.8.12 DTACK Mode - AXI Single Write Access

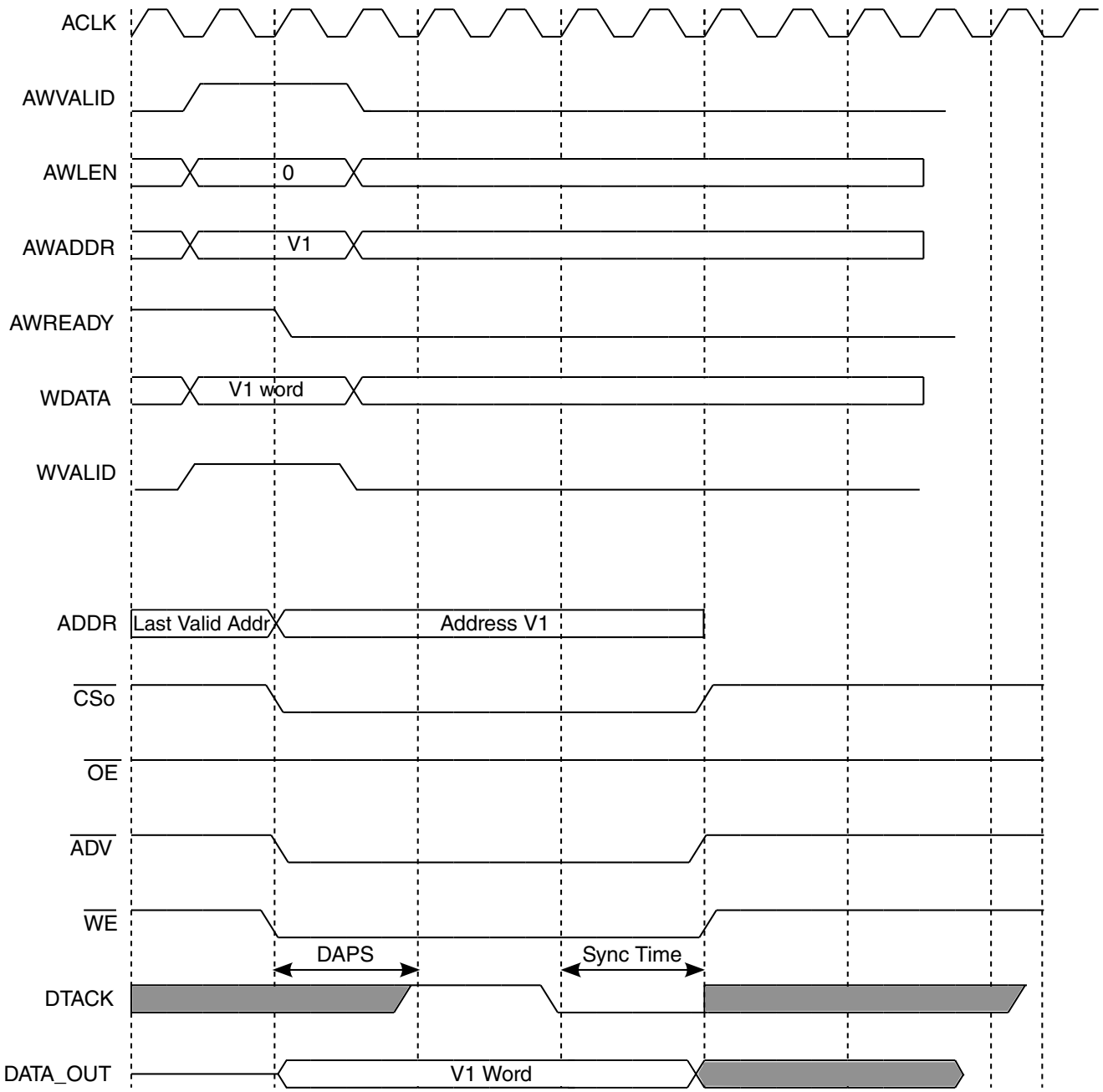


Figure 21-18. DAPS = 2

21.8.13 DTACK Mode - AXI Burst Access

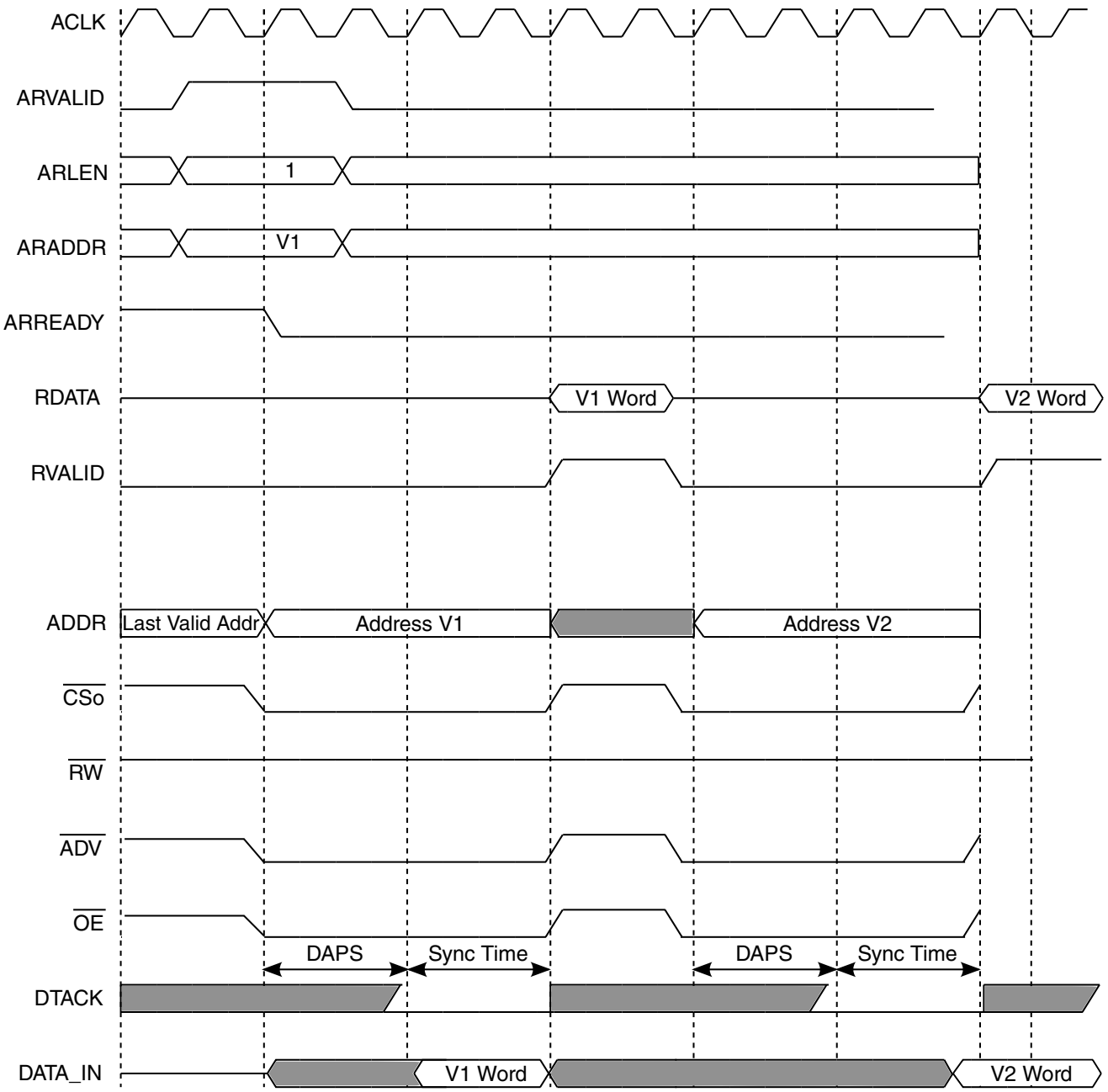


Figure 21-19. DAPS = 2 CSREC = 2

21.9 EIM Memory Map/Register Definition

The EIM includes 33 user-accessible 32-bit registers. The the EIM Configuration Register (EIM_WCR) contains control bits that configure the EIM for certain operation modes.

The 160 bits used to control Individual Chip Select are divided into five registers:

- Chip Select n General Configuration Register 1 (EIM_CSnGCR1)
- Chip Select n General Configuration Register 2 (EIM_CSnGCR2)
- Chip Select n Read Configuration Register 1 (EIM_CSnRCR1)
- Chip Select n Read Configuration Register 2 (EIM_CSnRCR2)
- Chip Select n Write Configuration Register (EIM_CSnWCR)

In addition there are 3 general registers: EIM_WCR, EIM_WIAR & EIM_EAR.

NOTE

- All EIM registers are sampled by IPG_CLK_S, therefore IPG_CLK_S must be active when accessing through IP bus.
- Read access from all registers (except EIM_WIAR & EIM_EAR) will generate one IPG_XFR_WAIT cycle.
- Read access from EIM_WIAR & EIM_EAR will generate six IPG_XFR_WAIT cycles.
- Write access to all registers (except EIM_EAR) will generate three IPG_XFR_WAIT cycles.
- Write access to EIM_EAR will generate six IPG_XFR_WAIT cycles.

EIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_8000	Chip Select n General Configuration Register 1 (EIM_CS0GCR1)	32	R/W	0061_0088h	21.9.1/849
21B_8004	Chip Select n General Configuration Register 2 (EIM_CS0GCR2)	32	R/W	0000_1010h	21.9.2/853
21B_8008	Chip Select n Read Configuration Register 1 (EIM_CS0RCR1)	32	R/W	1C00_2000h	21.9.3/854
21B_800C	Chip Select n Read Configuration Register 2 (EIM_CS0RCR2)	32	R/W	0000_0000h	21.9.4/857
21B_8010	Chip Select n Write Configuration Register 1 (EIM_CS0WCR1)	32	R/W	1C00_0000h	21.9.5/858
21B_8014	Chip Select n Write Configuration Register 2 (EIM_CS0WCR2)	32	R/W	0000_0000h	21.9.6/861
21B_8018	Chip Select n General Configuration Register 1 (EIM_CS1GCR1)	32	R/W	0061_0088h	21.9.1/849

Table continues on the next page...

EIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_801C	Chip Select n General Configuration Register 2 (EIM_CS1GCR2)	32	R/W	0000_1010h	21.9.2/853
21B_8020	Chip Select n Read Configuration Register 1 (EIM_CS1RCR1)	32	R/W	1C00_2000h	21.9.3/854
21B_8024	Chip Select n Read Configuration Register 2 (EIM_CS1RCR2)	32	R/W	0000_0000h	21.9.4/857
21B_8028	Chip Select n Write Configuration Register 1 (EIM_CS1WCR1)	32	R/W	1C00_0000h	21.9.5/858
21B_802C	Chip Select n Write Configuration Register 2 (EIM_CS1WCR2)	32	R/W	0000_0000h	21.9.6/861
21B_8030	Chip Select n General Configuration Register 1 (EIM_CS2GCR1)	32	R/W	0061_0088h	21.9.1/849
21B_8034	Chip Select n General Configuration Register 2 (EIM_CS2GCR2)	32	R/W	0000_1010h	21.9.2/853
21B_8038	Chip Select n Read Configuration Register 1 (EIM_CS2RCR1)	32	R/W	1C00_2000h	21.9.3/854
21B_803C	Chip Select n Read Configuration Register 2 (EIM_CS2RCR2)	32	R/W	0000_0000h	21.9.4/857
21B_8040	Chip Select n Write Configuration Register 1 (EIM_CS2WCR1)	32	R/W	1C00_0000h	21.9.5/858
21B_8044	Chip Select n Write Configuration Register 2 (EIM_CS2WCR2)	32	R/W	0000_0000h	21.9.6/861
21B_8048	Chip Select n General Configuration Register 1 (EIM_CS3GCR1)	32	R/W	0061_0088h	21.9.1/849
21B_804C	Chip Select n General Configuration Register 2 (EIM_CS3GCR2)	32	R/W	0000_1010h	21.9.2/853
21B_8050	Chip Select n Read Configuration Register 1 (EIM_CS3RCR1)	32	R/W	1C00_2000h	21.9.3/854
21B_8054	Chip Select n Read Configuration Register 2 (EIM_CS3RCR2)	32	R/W	0000_0000h	21.9.4/857
21B_8058	Chip Select n Write Configuration Register 1 (EIM_CS3WCR1)	32	R/W	1C00_0000h	21.9.5/858
21B_805C	Chip Select n Write Configuration Register 2 (EIM_CS3WCR2)	32	R/W	0000_0000h	21.9.6/861
21B_8090	EIM Configuration Register (EIM_WCR)	32	R/W	See section	21.9.7/862

21.9.1 Chip Select n General Configuration Register 1 (EIM_CS*n*GCR1)

Address: 21B_8000h base + 0h offset + (24d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PSZ				WP	GBC			AUS	CSREC			SP	DSZ		
Reset	0	0	0	0	0	0	0	0	0				0			

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BCS		BCD		WC	BL			CREP	CRE	RFL	WFL	MUM	SRD	SWR	CSEN
Reset	0	0	0	0	0	0	0	0	1	0	0	0		0	0	0

EIM_CS*n*GCR1 field descriptions

Field	Description
31–28 PSZ	Page Size. This bit field indicates memory page size in words (word is defined by the DSZ field). PSZ is used when fix latency mode is applied, WFL=1 for sync. write accesses, RFL=1 for sync. Read accesses. When working in fix latency mode WAIT signal from the external device is not being monitored, PSZ is used to determine if page boundary is reached and renewal of access is preformed. This bit field is ignored when sync. Mode is disabled or fix latency mode is not being used for write or read access separately. It can be valid for both access type, read or write, or only for one type, according to configuration. PSZ is cleared by a hardware reset. 0000 8 words page size 0001 16 words page size 0010 32 words page size 0011 64 words page size 0100 128 words page size 0101 256 words page size 0110 512 words page size 0111 1024 (1k) words page size 1000 2048 (2k) words page size 1001 - 1111 Reserved
27 WP	Write Protect. This bit prevents writes to the address range defined by the corresponding chip select. WP is cleared by a hardware reset. 0 Writes are allowed in the memory range defined by chip. 1 Writes are prohibited. All attempts to write to an address mapped by this chip select result in a error response and no assertion of the chip select output.
26–24 GBC	Gap Between Chip Selects. This bit field, according to the settings shown below, determines the minimum time between end of access to the current chip select and start of access to different chip select. GBC is cleared by a hardware reset.

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	Example settings: 000 minimum of 0 EIM clock cycles before next access from different chip select (async. mode only) 001 minimum of 1 EIM clock cycles before next access from different chip select 010 minimum of 2 EIM clock cycles before next access from different chip select 111 minimum of 7 EIM clock cycles before next access from different chip select
23 AUS	Address UnShifted. This bit indicates an unshifted mode for address assertion for the relevant chip select accesses. AUS bit is cleared by hardware reset. 0 Address shifted according to port size (DSZ config) (128 Mbyte maximum supported memory density). 1 Address unshifted (32 Mbyte maximum supported memory density).
22–20 CSREC	CS Recovery. This bit field, according to the settings shown below, determines the minimum pulse width of CS, OE, and WE control signals before executing a new back to back access to the same chip select. CSREC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0GCR1, CSREC[2:0] is 0b110. For EIM_CS1GCR1 - EIM_CS5GCR, the reset value is 0b000. Example settings: 000 0 EIM clock cycles minimum width of CS, OE and WE signals (read async. mode only) 001 1 EIM clock cycles minimum width of CS, OE and WE signals 010 2 EIM clock cycles minimum width of CS, OE and WE signals 111 7 EIM clock cycles minimum width of CS, OE and WE signals
19 SP	Supervisor Protect. This bit prevents accesses to the address range defined by the corresponding chip select when the access is attempted in the User mode. SP is cleared by a hardware reset. 0 User mode accesses are allowed in the memory range defined by chip select. 1 User mode accesses are prohibited. All attempts to access an address mapped by this chip select in User mode results in an error response and no assertion of the chip select output.
18–16 DSZ	Data Port Size. This bit field defines the width of an external device's data port as shown below. **NOTE:** Only async. access supported for 8 bit port. **NOTE:** The reset value for EIM_CS0GCR1, DSZ[2] = 0, DSZ[1:0] = EIM_BOOT[1:0]. For EIM_CS1GCR1 - EIM_CS5GCR1, the reset value is 0b001. 000 Reserved. 001 16 bit port resides on DATA[15:0] 010 16 bit port resides on DATA[31:16] 011 32 bit port resides on DATA[31:0] 100 8 bit port resides on DATA[7:0] 101 8 bit port resides on DATA[15:8] 110 8 bit port resides on DATA[23:16] 111 8 bit port resides on DATA[31:24]
15–14 BCS	Burst Clock Start. When SRD=1 or SWR=1,this bit field determines the number of EIM clock cycles delay from start of access before the first rising edge of BCLK is generated. When BCD=0 value of BCS=0 results in a half clock delay after the start of access. For other values of BCD a one clock delay after the start of access is applied, not an immediate assertion. BCS is cleared by a hardware reset. 00 0 EIM clock cycle additional delay

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	01 1 EIM clock cycle additional delay 10 2 EIM clock cycle additional delay 11 3 EIM clock cycle additional delay
13–12 BCD	Burst Clock Divisor. This bit field contains the value used to program the burst clock divisor for BCLK generation. It is used to divide the internal EIMbus frequency. BCD is cleared by a hardware reset. **NOTE:** For other then the mentioned below frequency such as 104 MHz, EIM clock (input clock) should be adjust accordingly. 00 Divide EIM clock by 1 01 Divide EIM clock by 2 10 Divide EIM clock by 3 11 Divide EIM clock by 4
11 WC	Write Continuous. The WI bit indicates that write access to the memory are always continuous accesses regardless of the BL field value. WI is cleared by hardware reset. 0 Write access burst length occurs according to BL value. 1 Write access burst length is continuous.
10–8 BL	Burst Length. The BL bit field indicates memory burst length in words (word is defined by the DSZ field) and should be properly initialized for mixed wrap/increment accesses support. Continuous BL value corresponds to continuous burst length setting of the external memory device. For fix memory burst size, type is always wrap. In case not matching wrap boundaries in both the memory (BL field) and Master access on the current address, EIM update address on the external device address bus and regenerates the access. BL is cleared by a hardware reset. When APR=1, Page Read Mode is applied, BL determine the number of words within the read page burst. BL is cleared by a hardware reset for EIM_CS0GCR1 - EIM_CS5GCR1. 000 4 words Memory wrap burst length (read page burst size when APR = 1) 001 8 words Memory wrap burst length (read page burst size when APR = 1) 010 16 words Memory wrap burst length (read page burst size when APR = 1) 011 32 words Memory wrap burst length (read page burst size when APR = 1) 100 Continuous burst length (2 words read page burst size when APR = 1) 101 Reserved 110 Reserved 111 Reserved
7 CREP	Configuration Register Enable Polarity. This bit indicates CRE memory pin assertion state, active-low or active-high, while executing a memory register set command to the external device (PSRAM memory type). CREP is set by a hardware reset. **NOTE:** Whenever PSRAM is connected the CREP value must be correct also for accesses where CRE is disabled. For Non-PSRAM memory CREP value should be 1. 0 CRE signal is active low 1 CRE signal is active high
6 CRE	Configuration Register Enable. This bit indicates CRE memory pin state while executing a memory register set command to PSRAM external device. CRE is cleared by a hardware reset.

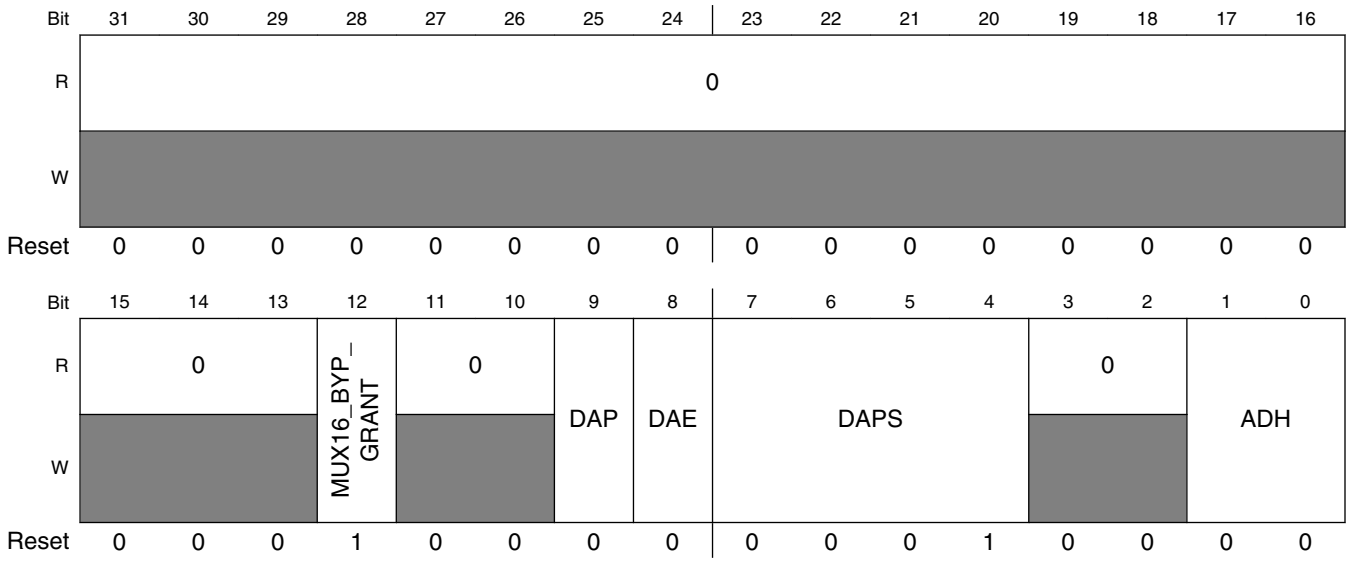
Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	0 CRE signal use is disable 1 CRE signal use is enable
5 RFL	Read Fix Latency. This bit field determine if the controller is monitoring the WAIT signal from the External device connected to the chip select (handshake mode - fix or variable data latency) or if it start sampling data according to RWSC field, it only valid in synchronous mode. RFL is cleared by a hardware reset. When RFL=1 Burst access is terminated on page boundary and resume on the following page according to BL bit field configuration, because WAIT signal is not monitored from the external device. 0 the External device WAIT signal is being monitored, and it reflect the external data bus state 1 the state of the External devices is determined internally (Fix latency mode only)
4 WFL	Write Fix Latency. This bit field determine if the controller is monitoring the WAIT signal from the External device connected to the chip select (handshake mode - fix or variable data latency) or if it start data transfer according to WWSC field, it only valid in synchronous mode. WFL is cleared by a hardware reset. When WFL=1 Burst access is terminated on page boundary and resume on the following page according to BL bit field configuration, because WAIT signal is not monitored from the external device 0 the External device WAIT signal is being monitored, and it reflect the external data bus state 1 the state of the External devices is determined internally (Fix latency mode only)
3 MUM	Multiplexed Mode. This bit determines the address/data multiplexed mode for asynchronous and synchronous accesses for 8 bit, 16 bit or 32 bit devices (DSZ config. dependent). **NOTE:** The reset value for EIM_CS0GCR1[MUM] = EIM_BOOT[2]. For EIM_CS1GCR1 - EIM_CS5GCR1 the reset value is 0. 0 Multiplexed Mode disable 1 Multiplexed Mode enable
2 SRD	Synchronous Read Data. This bit field determine the read accesses mode to the External device of the chip select. The External device should be configured to the same mode as this bit implicates. SRD is cleared by a hardware reset. **NOTE:** Sync. accesses supported only for 16/32 bit port. 0 read accesses are in Asynchronous mode 1 read accesses are in Synchronous mode
1 SWR	Synchronous Write Data. This bit field determine the write accesses mode to the External device of the chip select. The External device should be configured to the same mode as this bit implicates. SWR is cleared by a hardware reset. **NOTE:** Sync. accesses supported only for 16/32 bit port. 0 write accesses are in Asynchronous mode 1 write accesses are in Synchronous mode
0 CSEN	CS Enable. This bit controls the operation of the chip select pin. CSEN is set by a hardware reset for CSGCR0 to allow external boot operation. CSEN is cleared by a hardware reset to CSGCR1-CSGCR5. **NOTE:** Reset value for EIM_CS0GCR1 for CSEN is 1. For EIM_CS1GCR1-CS1GCR5 reset value is 0. 0 Chip select function is disabled; attempts to access an address mapped by this chip select results in an error respond and no assertion of the chip select output 1 Chip select is enabled, and is asserted when presented with a valid access.

21.9.2 Chip Select n General Configuration Register 2 (EIM_CS*n*GCR2)

Address: 21B_8000h base + 4h offset + (24d × i), where i=0d to 3d



EIM_CS*n*GCR2 field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 MUX16_BYP_GRANT	Muxed 16 bypass grant. This bit when asserted causes EIM to bypass the grant/ack. arbitration with NFC (only for 16 bit muxed mode accesses). 0 EIM waits for grant before driving a 16 bit muxed mode access to the memory. 1 EIM ignores the grant signal and immediately drives a 16 bit muxed mode access to the memory.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9 DAP	Data Acknowledge Polarity. This bit indicates DTACK memory pin assertion state, active-low or active-high, while executing an async access using DTACK signal from the external device. DAP is cleared by a hardware reset. 0 DTACK signal is active high 1 DTACK signal is active low
8 DAE	Data Acknowledge Enable. This bit indicates external device is using DTACK pin as strobe/terminator of an async. access. DTACK signal may be used only in asynchronous single read (APR=0) or write accesses. DTACK poling start point is set by DAPS bit field. polarity of DTACK is set by DAP bit field. DAE is cleared by a hardware reset. 0 DTACK signal use is disable 1 DTACK signal use is enable
7–4 DAPS	Data Acknowledge Poling Start. This bit field determine the starting point of DTACK input signal polling. DAPS is used only in asynchronous single read or write accesses.

Table continues on the next page...

EIM_CS*n*GCR2 field descriptions (continued)

Field	Description
	NOTE: Since DTACK is an async. signal the start point of DTACK signal polling is at least 3 cycles after the start of access. DAPS is cleared by a hardware reset. Example settings: 0000 3 EIM clk cycle between start of access and first $\overline{\text{DTACK}}$ check 0001 4 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 0010 5 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 0111 10 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 1011 14 EIM clk cycles between start of access and first DTACK check 1111 18 EIM clk cycles between start of access and first DTACK check
3–2 Reserved	This read-only field is reserved and always has the value 0.
ADH	Address hold time - This bit field determine the address hold time after ADV negation when mum = 1 (muxed mode). When mum = 0 this bit has no effect. For read accesses the field determines when the pads direction will be switched. **NOTE:** Reset value for EIM_CS0GCR2 for ADH is 0b10. For EIM_CS1GCR2-EIM_CS5GCR2 reset value is 0b00. 00 0 cycle after ADV negation 01 1 cycle after ADV negation 10 2 cycle after ADV negation 11 Reserved

21.9.3 Chip Select n Read Configuration Register 1 (EIM_CS*n*RCR1)

Address: 21B_8000h base + 8h offset + (24d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		RWSC						0	RADVA			RAL	RADVN		
W																
Reset	0	0							0	0	0	0	0	0	0	0

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

EIM_CS*n*RCR1 field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

EIM_CS*n*RCR1 field descriptions (continued)

Field	Description
29–24 RWSC	Read Wait State Control. This bit field programs the number of wait-states, according to the settings shown below, for synchronous or asynchronous read access to the external device connected to the chip select. When SRD=1 and RFL=0, RWSC indicates the number of burst clock (BCLK) cycles from the start of an access, before the controller can start sample data.Since WAIT signal can be asserted one cycle before the first data can be sampled, the controller starts evaluating the WAIT signal state one cycle before, this is referred as handshake mode or variable latency mode. When SRD=1 and RFL=1, RWSC indicates the number of burst clock (BCLK) cycles from the start of an access, until the external device is ready for data transfer, this is referred as fix latency mode. When SRD=0, RFL bit is ignored, RWSC indicates the asynchronous access length and the number of EIM clock cycles from the start of access until the external device is ready for data transfer. RWSC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0RCR1, RWSC[5:0] = 0b011100. For CG1RCR1 - CS1RCR5 the reset value is 0b000000. Example settings: 000000 Reserved 000001 RWSC value is 1 000010 RWSC value is 2 111101 RWSC value is 61 111110 RWSC value is 62 111111 RWSC value is 63
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 RADVA	ADV Assertion. This bit field determines when ADV signal is asserted for synchronous or asynchronous read modes according to the settings shown below. RADVA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of access and ADV assertion 001 1 EIM clock cycles between beginning of access and ADV assertion 010 2 EIM clock cycles between beginning of access and ADV assertion 111 7 EIM clock cycles between beginning of access and ADV assertion
19 RAL	Read ADV Low. This bit field determine ADV signal negation time. When RAL=1, RADVN bit field is ignored and ADV signal will stay asserted until end of access. When RAL=0 negation of ADV signal is according to RADVN bit field configuration.
18–16 RADVN	ADV Negation. This bit field determines when ADV signal to memory is negated during read accesses. When SRD=1 (synchronous read mode), ADV negation occurs according to the following formula: (RADVN + RADVA + BCD + BCS + 1) EIM clock cycles from start of access. When asynchronous read mode is applied (SRD=0) and RAL=0 ADV negation occurs according to the following formula: (RADVN + RADVA + 1) EIM clock cycles from start of access. RADVN is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0RCR1[RADVN] = 0b010. For EIM_CS1RCR1 - EIM_CS5RCR1, the reset value is 0b000. **NOTE:** This field should be configured so ADV negation will occur before the end of access. For ADV negation at the same time with the end of access user should RAL bit.

Table continues on the next page...

EIM_CS*n*RCR1 field descriptions (continued)

Field	Description
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 OEA	OE Assertion. This bit field determines when OE signal are asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. OEA is cleared by a hardware reset. In muxed mode OE assertion occurs (OEA + RADVN + RADVA + ADH +1) EIM clock cycles from start of access. **NOTE:** The reset value for EIM_CS0RCR1[OEA] is 0b000 if EIM_BOOT[2] = 0. If EIM_BOOT[2] is 1, the reset value for EIM_CS0RCR1[OEA] is 0b010. The reset value of this field for EIM_CS1RCR1 - EIM_CS5RCR1 is 0b000. Example settings: 000 0 EIM clock cycles between beginning of access and OE assertion 001 1 EIM clock cycles between beginning of access and OE assertion 010 2 EIM clock cycles between beginning of access and OE assertion 111 7 EIM clock cycles between beginning of access and OE assertion
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 OEN	OE Negation. This bit field determines when OE signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR = 0), according to the settings shown below. This bit field is ignored when SRD=1. OEN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of access and OE negation 001 1 EIM clock cycles between end of access and OE negation 010 2 EIM clock cycles between end of access and OE negation 111 7 EIM clock cycles between end of access and OE negation
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 RCSA	Read CS Assertion. This bit field determines when CS signal is asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. RCSA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of read access and CS assertion 001 1 EIM clock cycles between beginning of read access and CS assertion 010 2 EIM clock cycles between beginning of read access and CS assertion 111 7 EIM clock cycles between beginning of read access and CS assertion
3 Reserved	This read-only field is reserved and always has the value 0.
RCSN	Read CS Negation. This bit field determines when CS signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR = 0), according to the settings shown below. This bit field is ignored when SRD=1. RCSN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and CS negation 001 1 EIM clock cycles between end of read access and CS negation 010 2 EIM clock cycles between end of read access and CS negation 111 7 EIM clock cycles between end of read access and CS negation

21.9.4 Chip Select n Read Configuration Register 2 (EIM_CS*n*RCR2)

Address: 21B_8000h base + Ch offset + (24d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APR	PAT			0		RL		0	RBEA			RBE	RBEN		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*RCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 APR	Asynchronous Page Read. This bit field determine the asynchronous read mode to the external device. When APR=0, the async. read access is done as single word (where word is defined by the DSZ field). when APR=1, the async. read access executed as page read. page size is according to BL field config., RCSN,RBEN,OEN and RADVN are being ignored. APR is cleared by a hardware reset for EIM_CS1GCR1 - EIM_CS5GCR1. **NOTE:** SRD=0 and MUM=0 must apply when APR=1
14–12 PAT	Page Access Time. This bit field is used in Asynchronous Page Read mode only (APR=1). the initial access is set by RWSC as in regular asynchronous mode. the consecutive address assertions width determine by PAT field according to the settings shown below. when APR=0 this field is ignored. PAT is cleared by a hardware reset for EIM_CS1GCR1 - EIM_CS5GCR1. 000 Address width is 2 EIM clock cycles 001 Address width is 3 EIM clock cycles 010 Address width is 4 EIM clock cycles 011 Address width is 5 EIM clock cycles 100 Address width is 6 EIM clock cycles 101 Address width is 7 EIM clock cycles 110 Address width is 8 EIM clock cycles 111 Address width is 9 EIM clock cycles
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 RL	Read Latency. This bit field indicates cycle latency when executing a synchronous read operation. The fields holds the feedback clock loop delay in aclk cycle units. This field is cleared by a hardware reset. 00 Feedback clock loop delay is up to 1 cycle for BCD = 0 or 1.5 cycles for BCD != 0 01 Feedback clock loop delay is up to 2 cycles for BCD = 0 or 2.5 cycles for BCD != 0

Table continues on the next page...

EIM_CS*n*RCR2 field descriptions (continued)

Field	Description
	10 Feedback clock loop delay is up to 3 cycles for BCD = 0 or 3.5 cycles for BCD != 0 11 Feedback clock loop delay is up to 4 cycles for BCD = 0 or 4.5 cycles for BCD != 0
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 RBEA	Read BE Assertion. This bit field determines when BE signal is asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. RBEA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of read access and BE assertion 001 1 EIM clock cycles between beginning of read access and BE assertion 010 2 EIM clock cycles between beginning of read access and BE assertion 111 7 EIM clock cycles between beginning of read access and BE assertion
3 RBE	Read BE enable. This bit field determines if BE will be asserted during read access. 0 BE are disabled during read access. 1 BE are enable during read access according to value of RBEA & RBEN bit fields.
RBEN	Read BE Negation. This bit field determines when BE signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR=0), according to the settings shown below. This bit field is ignored when SRD=1. RBEN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and BE negation 001 1 EIM clock cycles between end of read access and BE negation 010 2 EIM clock cycles between end of read access and BE negation 111 7 EIM clock cycles between end of read access and BE negation

21.9.5 Chip Select n Write Configuration Register 1 (EIM_CS*n*WCR1)

Address: 21B_8000h base + 10h offset + (24d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WAL	WBED	WWSC						WADVA			WADVN			WBEA	
Reset	0	0							0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WBEA	WBEN			WEA				WEN			WCSA			WCSN	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*WCR1 field descriptions

Field	Description
31 WAL	Write ADV Low. This bit field determine ADV signal negation time in write accesses. When WAL=1, WADVN bit field is ignored and ADV signal will stay asserted until end of access. When WAL=0 negation of ADV signal is according to WADVN bit field configuration.
30 WBED	Write Byte Enable Disable. When asserted this bit prevent from IPP_DO_BE_B[x] to be asserted during write accesses.This bit is cleared by hardware reset.
29–24 WWSC	Write Wait State Control. This bit field programs the number of wait-states, according to the settings shown below, for synchronous or asynchronous write access to the external device connected to the chip select. When SWR=1 and WFL=0, WWSC indicates the number of burst clock (BCLK) cycles from the start of an access, before the memory can sample the first data.Since WAIT signal can be asserted one cycle before the first data can be sampled, the controller starts evaluating the WAIT signal state one cycle before, this is referred as handshake mode or variable latency mode. When SWR=1 and WFL=1, WWSC indicates the number of burst clock (BCLK) cycles from the start of an access, until the external device is ready for data transfer, this is referred as fix latency mode. When SWR=0, WFL bit is ignored, WWSC indicates the asynchronous access length and the number of EIM clock cycles from the start of access until the external device is ready for data transfer. WWSC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0WCR1, WWSC[5:0] = 0b011100. For EIM_CS1WCR1 - EIM_CS5WCR1, the reset value of this field is 0b000000. Example settings: 000000 Reserved 000001 WWSC value is 1 000010 WWSC value is 2 000011 WWSC value is 3 111111 WWSC value is 63
23–21 WADVA	ADV Assertion. This bit field determines when ADV signal is asserted for synchronous or asynchronous write modes according to the settings shown below. WADVA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of access and ADV assertion 001 1 EIM clock cycles between beginning of access and ADV assertion 010 2 EIM clock cycles between beginning of access and ADV assertion 111 7 EIM clock cycles between beginning of access and ADV assertion
20–18 WADVN	ADV Negation. This bit field determines when ADV signal to memory is negated during write accesses. When SWR=1 (synchronous write mode), ADV negation occurs according to the following formula: (WADVN + WADVA + BCD + BCS + 1) EIM clock cycles. When asynchronous read mode is applied (SWR=0) ADV negation occurs according to the following formula: (WADVN + WADVA + 1) EIM clock cycles. **NOTE:** Reset value for EIM_CS0WCR for WADVN is 0b010. For EIM_CS1WCR - EIM_CS5WCR reset value is 0b000. **NOTE:** This field should be configured so ADV negation will occur before the end of access. For ADV negation at the same time as the end of access, S/W should set the WAL bit.
17–15 WBEA	BE Assertion. This bit field determines when BE signal is asserted during write cycles in async. mode only (SWR=0), according to the settings shown below. BEA is cleared by a hardware reset.

Table continues on the next page...

EIM_CS*n*WCR1 field descriptions (continued)

Field	Description
	NOTE: Reset value for EIM_CS0WCR for WBEA is 0b010. For EIM_CS1WCR - EIM_CS5WCR reset value is 0b000. Example settings: 000 0 EIM clock cycles between beginning of access and BE assertion 001 1 EIM clock cycles between beginning of access and BE assertion 010 2 EIM clock cycles between beginning of access and BE assertion 111 7 EIM clock cycles between beginning of access and BE assertion
14–12 WBEN	BE[3:0] Negation. This bit field determines when BE[3:0] bus signal is negated during write cycles in async. mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. BEN is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WBEN is 0b010. For EIM_CS1WCR - EIM_CS5WCR reset value is 0b000. Example settings: 000 0 EIM clock cycles between end of access and WE negation 001 1 EIM clock cycles between end of access and WE negation 010 2 EIM clock cycles between end of access and WE negation 111 7 EIM clock cycles between end of access and WE negation
11–9 WEA	WE Assertion. This bit field determines when WE signal is asserted during write cycles (synchronous or asynchronous mode), according to the settings shown below. This bit field is ignored when executing a read access to the external device. WEA is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WEA is 0b010. For EIM_CS1WCR - EIM_CS5WCR reset value is 0b000. Example settings: 000 0 EIM clock cycles between beginning of access and WE assertion 001 1 EIM clock cycles between beginning of access and WE assertion 010 2 EIM clock cycles between beginning of access and WE assertion 111 7 EIMclock cycles between beginning of access and WE assertion
8–6 WEN	WE Negation. This bit field determines when WE signal is negated during write cycles in asynchronous mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. WEN is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WEN is 0b010. For EIM_CS1WCR - EIM_CS5WCR reset value is 0b000. Example settings: 000 0 EIM clock cycles between beginning of access and WE assertion 001 1 EIM clock cycles between beginning of access and WE assertion 010 2 EIM clock cycles between beginning of access and WE assertion 111 7 EIM clock cycles between beginning of access and WE assertion
5–3 WCSA	Write CS Assertion. This bit field determines when CS signal is asserted during write cycles (synchronous or asynchronous mode), according to the settings shown below.this bit field is ignored when executing a read access to the external device. WCSA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of write access and CS assertion

Table continues on the next page...

EIM_CS*n*WCR1 field descriptions (continued)

Field	Description
	001 1 EIM clock cycles between beginning of write access and CS assertion 010 2 EIM clock cycles between beginning of write access and CS assertion 111 7 EIMclock cycles between beginning of write access and CS assertion
WCSN	Write CS Negation. This bit field determines when CS signal is negated during write cycles in asynchronous mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. WCSN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and CS negation 001 1 EIM clock cycles between end of read access and CS negation 010 2 EIM clock cycles between end of read access and CS negation 111 7 EIM clock cycles between end of read access and CS negation

21.9.6 Chip Select n Write Configuration Register 2 (EIM_CS*n*WCR2)

Address: 21B_8000h base + 14h offset + (24d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

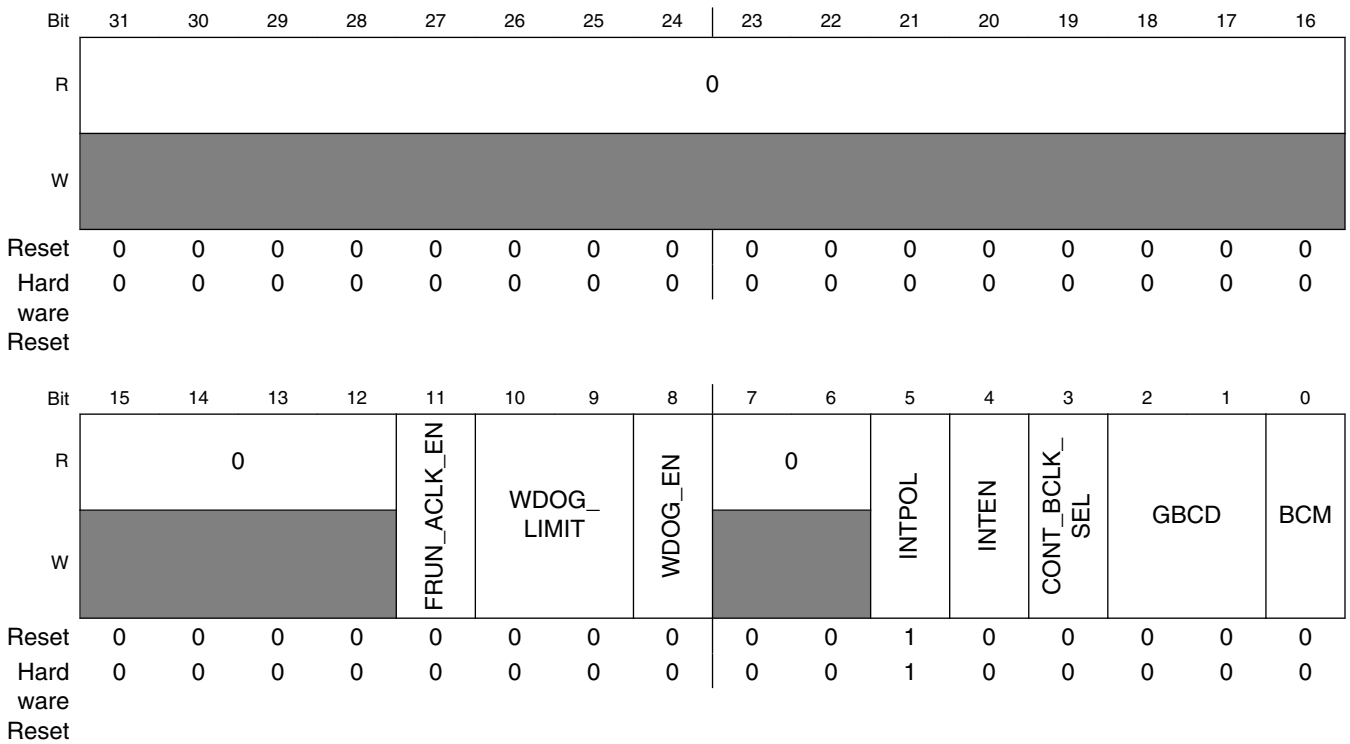
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															WBCDD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*WCR2 field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 WBCDD	Write Burst Clock Divisor Decrement. If this bit is asserted and BCD value is 0 sync. write access will be preformed as if BCD value is 1.When this bit is negated or BCD value is not 0 this bit has no affect. This bit is cleared by hardware reset.

21.9.7 EIM Configuration Register (EIM_WCR)

Address: 21B_8000h base + 90h offset = 21B_8090h



EIM_WCR field descriptions

Field	Description
31–12 Reserved	Reserved This read-only field is reserved and always has the value 0.
11 FRUN_ACLK_EN	Free run ACLK enable
10–9 WDOG_LIMIT	Memory Watchdog (WDOG) cycle limit. This bit field determines the number of BCLK cycles (ACLK cycles in dtack mode) before the WDOG counter terminates the access and send an error response to the master. 00 128 BCLK cycles 01 256 BCLK cycles 10 512 BCLK cycles 11 1024 BCLK cycles
8 WDOG_EN	Memory WDOG enable. This bit controls the operation of the wdog counter that terminates the EIM access. 0 Memory WDOG is Disabled 1 Memory WDOG is Enabled

Table continues on the next page...

EIM_WCR field descriptions (continued)

Field	Description
7–6 Reserved	This read-only field is reserved and always has the value 0.
5 INTPOL	Interrupt Polarity. This bit field determines the polarity of the external device interrupt. 0 External interrupt polarity is active low 1 External interrupt polarity is active high
4 INTEN	Interrupt Enable. When this bit is set the External signal RDY_INT as active interrupt. When interrupt occurs, INT bit at the WCR will be set and t EIM_EXT_INT signal will be asserted correspondingly. This bit is cleared by a hardware reset. 0 External interrupt Disable 1 External interrupt Enable
3 CONT_BCLK_ SEL	Continuous BCLK select When this bit is set BCLK pin output continuous clock. Otherwize, BCLK will output clock only when nesserary. 0 BCLK When nesserary 1 BCLK Continuous
2–1 GBCD	General Burst Clock Divisor. When BCM bit is set, this bit field contains the value used to program the burst clock divisor for Continuous BCLK generation. The other BCD bit fields for each chip select are ignored. It is used to divide the internal AXI bus frequency. When BCM=0 GBCD bit field has no influence. GBCD is cleared by a hardware reset. 00 Divide EIM clock by 1 01 Divide EIM clock by 2 10 Divide EIM clock by 3 11 Divide EIM clock by 4
0 BCM	Burst Clock Mode. This bit selects the burst clock mode of operation. It is used for system debug mode. BCM is cleared by a hardware reset. **NOTE:** The BCLK frequency in this mode is according to GBCD bit field. **NOTE:** The BCLK phase is opposite to the EIM clock in this mode if GBCD is 0. **NOTE:** This bit should be used only in async. accesses. No sync access can be executed if this bit is set. **NOTE:** When this bit is set bcd field shouldn't be configured to 0. 0 The burst clock runs only when accessing a chip select range with the SWR/SRD bits set. When the burst clock is not running it remains in a logic 0 state. When the burst clock is running it is configured by the BCD and BCS bit fields in the chip select Configuration Register. 1 The burst clock runs whenever ACLK is active (independent of chip select configuration)

Chapter 22
10/100-Mbps Ethernet MAC (ENET)

22.1 Introduction

The MAC-NET core, in conjunction with a 10/100-Mbit/s MAC, implements layer 3 network acceleration functions. These functions are designed to accelerate the processing of various common networking protocols, such as IP, TCP, UDP, and ICMP, providing wire speed services to client applications.

22.2 Overview

The core implements a dual-speed 10/100-Mbit/s Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with half- or full-duplex 10/100-Mbit/s Ethernet LANs.

The MAC operation is fully programmable and can be used in Network Interface Card (NIC), bridging, or switching applications. The core implements the remote network monitoring (RMON) counters according to IETF RFC 2819.

The core also implements a hardware acceleration block to optimize the performance of network controllers providing TCP/IP, UDP, and ICMP protocol services. The acceleration block performs critical functions in hardware, which are typically implemented with large software overhead.

The core implements programmable embedded FIFOs that can provide buffering on the receive path for lossless flow control.

Advanced power management features are available with magic packet detection and programmable power-down modes.

A unified DMA (uDMA), internal to the ENET module, optimizes data transfer between the ENET core and the SoC, and supports an enhanced buffer descriptor programming model to support IEEE 1588 functionality.

The programmable Ethernet MAC with IEEE 1588 integrates a standard IEEE 802.3 Ethernet MAC with a time-stamping module. The IEEE 1588 standard provides accurate clock synchronization for distributed control nodes for industrial automation applications.

22.2.1 Features

The MAC-NET core includes the following features.

22.2.1.1 Ethernet MAC features

- Implements the full 802.3 specification with preamble/SFD generation, frame padding generation, CRC generation and checking
- Supports zero-length preamble
- Dynamically configurable to support 10/100-Mbit/s operation
- Supports 10/100 Mbit/s full-duplex and configurable half-duplex operation
- Compliant with the AMD magic packet detection with interrupt for node remote power management
- Seamless interface to commercial ethernet PHY devices via one of the following:
  - a 4-bit Media Independent Interface (MII) operating at 2.5/25 MHz.
  - a 4-bit non-standard MII-Lite (MII without the CRS and COL signals) operating at 2.5/25 MHz.
  - a 2-bit Reduced MII (RMII) operating at 50 MHz.
- Simple 64-Bit FIFO user-application interface
- CRC-32 checking at full speed with optional forwarding of the frame check sequence (FCS) field to the client
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- In full-duplex mode:
  - Implements automated pause frame (802.3 x31A) generation and termination, providing flow control without user application intervention
  - Pause quanta used to form pause frames — dynamically programmable
  - Pause frame generation additionally controllable by user application offering flexible traffic flow control
  - Optional forwarding of received pause frames to the user application
  - Implements standard flow-control mechanism
- In half-duplex mode: provides full collision support, including jamming, backoff, and automatic retransmission
- Supports VLAN-tagged frames according to IEEE 802.1Q
- Programmable MAC address: Insertion on transmit; discards frames with mismatching destination address on receive (except broadcast and pause frames)

- Programmable promiscuous mode support to omit MAC destination address checking on receive
- Multicast and unicast address filtering on receive based on 64-entry hash table, reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length
- Statistics indicators for frame traffic and errors (alignment, CRC, length) and pause frames providing for IEEE 802.3 basic and mandatory management information database (MIB) package and remote network monitoring (RFC 2819)
- Simple handshake user application FIFO interface with fully programmable depth and threshold levels
- Provides separate status word for each received frame on the user interface providing information such as frame length, frame type, VLAN tag, and error information
- Multiple internal loopback options
- MDIO master interface for PHY device configuration and management supports two programmable MDIO base addresses, and standard (IEEE 802.3 Clause 22) and extended (Clause 45) MDIO frame formats
- Supports legacy FEC buffer descriptors
- Interrupt coalescing reduces the number of interrupts generated by the MAC, reducing CPU loading

22.2.1.2 IP protocol performance optimization features

- Operates on TCP/IP and UDP/IP and ICMP/IP protocol data or IP header only
- Enables wire-speed processing
- Supports IPv4 and IPv6
- Transparent passing of frames of other types and protocols
- Supports VLAN tagged frames according to IEEE 802.1q with transparent forwarding of VLAN tag and control field
- Automatic IP-header and payload (protocol specific) checksum calculation and verification on receive
- Automatic IP-header and payload (protocol specific) checksum generation and automatic insertion on transmit configurable on a per-frame basis
- Supports IP and TCP, UDP, ICMP data for checksum generation and checking
- Supports full header options for IPv4 and TCP protocol headers

- Provides IPv6 support to datagrams with base header only — datagrams with extension headers are passed transparently unmodifed/unchecked
- Provides statistics information for received IP and protocol errors
- Configurable automatic discard of erroneous frames
- Configurable automatic host-to-network (RX) and network-to-host (TX) byte order conversion for IP and TCP/UDP/ICMP headers within the frame
- Configurable padding remove for short IP datagrams on receive
- Configurable Ethernet payload alignment to allow for 32-bit word-aligned header and payload processing
- Programmable store-and-forward operation with clock and rate decoupling FIFOs

22.2.1.3 IEEE 1588 features

- Supports all IEEE 1588 frames.
- Allows reference clock to be chosen independently of network speed.
- Software-programmable precise time-stamping of ingress and egress frames
- Timer monitoring capabilities for system calibration and timing accuracy management
- Precise time-stamping of external events with programmable interrupt generation
- Programmable event and interrupt generation for external system control
- Supports hardware- and software-controllable timer synchronization.
- Provides a 4-channel IEEE 1588 timer. Each channel supports input capture and output compare using the 1588 counter.

22.2.2 Block diagram

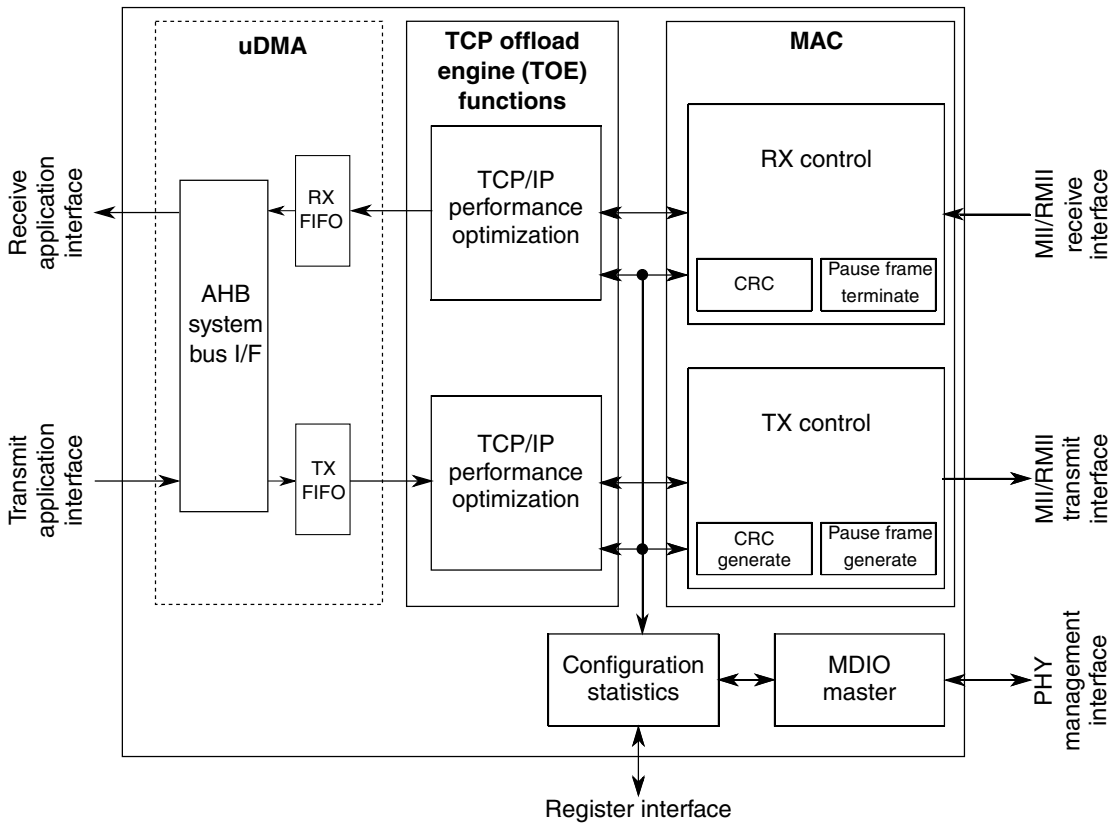


Figure 22-1. Ethernet MAC-NET core block diagram

22.3 External Signals

The table found here describes the external signals of ENET.

Table 22-1. ENET1 External Signals

Signal	Description	Mode	Pad	Alt Mode	Direction
ENET1_1588_EVENT0_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in	MII / RMII / RGMII	SD3_DATA7	ALT6	IO

Table continues on the next page...

Table 22-1. ENET1 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.				
ENET1_1588_EVENT0_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	SD3_DATA6	ALT6	IO
ENET1_1588_EVENT1_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	SD1_DATA0	ALT6	IO
ENET1_1588_EVENT1_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	SD1_DATA1	ALT6	IO

Table continues on the next page...

Table 22-1. ENET1 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
ENET1_1588_EVENT2_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	LCD1_CLK	ALT3	IO
ENET1_1588_EVENT2_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	LCD1_DATA20	ALT3	IO
ENET1_1588_EVENT3_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	LCD1_ENABLE	ALT3	IO
ENET1_1588_EVENT3_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is	MII / RMII / RGMII	LCD1_DATA21	ALT3	IO

Table continues on the next page...

Table 22-1. ENET1 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.				
ENET1_COL	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for fullduplex mode.	MII	ENET1_COL	ALT0	IO
ENET1_CRS	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the ENET_RX_EN pin.	MII	ENET1_CRS	ALT0	IO
ENET1_MDC	Output clock provides a timing reference to the PHY for data transfers on the MDIO signal.	MII / RMII	ENET1_MDC	ALT0	IO
			ENET2_COL	ALT1	
			GPIO1_IO04	ALT2	
ENET1_MDIO	Transfers control information between the external PHY and the mediaaccess controller. Data is synchronous to MDC. This signal is an input after reset.	MII / RMII	ENET1_MDIO	ALT0	IO
			ENET2_CRS	ALT1	
			GPIO1_IO05	ALT2	
ENET1_REF_CLK1	In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.	RMII	ENET1_TX_CLK	ALT1	IO
			GPIO1_IO05	ALT4	
ENET1_REF_CLK_25M	25 MHz Reference Clock	-	ENET1_RX_CLK	ALT1	IO
			GPIO1_IO03	ALT2	
ENET1_RGMII_RXC	In MII mode, provides a timing reference for RX_EN, RX_DATA[3:0], and RX_ER. In RGMII mode, provides a timing reference for RX_DATA[3:0] and RX_CTL.	RGMII	RGMII1_RXC	ALT0	IO
ENET1_RGMII_TXC	Serial output Ethernet data. Only valid during TX_EN assertion.	RGMII	RGMII1_TXC	ALT0	IO
ENET1_RX_CLK	In MII mode, provides a timing reference for RX_EN, RX_DATA[3:0], and RX_ER. In RGMII mode, provides a timing reference for RX_DATA[3:0] and RX_CTL.	MII	ENET1_RX_CLK	ALT0	IO

Table continues on the next page...

Table 22-1. ENET1 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
ENET1_RX_DATA0	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII / RMII / RGMII	RGMII1_RD0	ALT0	IO
ENET1_RX_DATA1	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII / RMII / RGMII	RGMII1_RD1	ALT0	IO
ENET1_RX_DATA2	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII / RGMII	RGMII1_RD2	ALT0	IO
ENET1_RX_DATA3	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII / RGMII	RGMII1_RD3	ALT0	IO
ENET1_RX_EN	Asserting this input indicates the PHY has valid nibbles present on the MII. RX_EN must remain asserted from the first recovered nibble of the frame through to the last nibble. Asserting RX_EN must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal. In RGMII mode, contains RXDV on the rising edge of RX_CLK, and a logical derivative of RX_EV and RX_ER (RX_EV XOR RX_ER) on the falling edge of RX_CLK.	MII / RMII / RGMII	RGMII1_RX_CTL	ALT0	IO
ENET1_RX_ER	When asserted with RXDV, indicates the PHY detects an error in the current frame.	MII / RMII	RGMII1_RXC	ALT1	IO
ENET1_TX_CLK	Input clock, which provides a timing reference for TX_EN, TX_DATA[3:0], and TX_ER.	MII	ENET1_TX_CLK	ALT0	IO
ENET1_TX_DATA0	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RMII / RGMII	RGMII1_TD0	ALT0	IO
ENET1_TX_DATA1	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RMII / RGMII	RGMII1_TD1	ALT0	IO
ENET1_TX_DATA2	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RGMII	RGMII1_TD2	ALT0	IO
ENET1_TX_DATA3	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RGMII	RGMII1_TD3	ALT0	IO
ENET1_TX_EN	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is deasserted before the first TX_CLK following the final nibble of the frame. In RGMII mode,	MII / RMII / RGMII	RGMII1_TX_CTL	ALT0	IO

Table continues on the next page...

Table 22-1. ENET1 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	contains TX_EN on the rising edge of TX_CTL, and a logical derivative of TX_EN and TX_ER (TX_EN XOR TX_ER) on the falling edge of TX_CTL.				
ENET1_TX_ER	When asserted for one or more clock cycles while TXEN is also asserted, PHY sends one or more illegal symbols.	MII	RGMII1_TXC	ALT1	IO

Table 22-2. ENET2 External Signals

Signal	Description	Mode	Pad	Alt Mode	Direction
ENET2_1588_EVENT0_IN	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	SD3_DATA4	ALT6	IO
ENET2_1588_EVENT0_OUT	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	SD3_DATA5	ALT6	IO
ENET2_1588_EVENT1_IN	Capture/compare block input/ output event bus signal. When configured for capture and a rising	MII / RMII / RGMII	SD1_CMD	ALT6	IO

Table continues on the next page...

Table 22-2. ENET2 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.				
ENET2_1588_EVENT1_OUT	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	SD1_CLK	ALT6	IO
ENET2_1588_EVENT2_IN	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	LCD1_HSYNC	ALT3	IO
ENET2_1588_EVENT2_OUT	Capture/compare block input/ output event bus signal. When configured for capture and a rising	MII / RMII / RGMII	LCD1_DATA22	ALT3	IO

Table continues on the next page...

Table 22-2. ENET2 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.				
ENET2_1588_EVENT3_IN	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	LCD1_VSYNC	ALT3	IO
ENET2_1588_EVENT3_OUT	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII / RMII / RGMII	LCD1_DATA23	ALT3	IO
ENET2_COL	Asserted upon detection of a collision and remains asserted while the collision persists. This	MII	ENET2_COL	ALT0	IO

Table continues on the next page...

Table 22-2. ENET2 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	signal is not defined for full duplex mode.				
ENET2_CRS	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the ENET_RX_EN pin.	MII	ENET2_CRS	ALT0	IO
ENET2_MDC	Output clock provides a timing reference to the PHY for data transfers on the MDIO signal.	MII / RMII	ENET1_COL	ALT1	IO
			ENET1_MDC	ALT1	
			GPIO1_IO06	ALT2	
			KEY_COL4	ALT1	
ENET2_MDIO	Transfers control information between the external PHY and the mediaaccess controller. Data is synchronous to MDC. This signal is an input after reset.	MII / RMII	ENET1_CRS	ALT1	IO
			ENET1_MDIO	ALT1	
			GPIO1_IO07	ALT2	
			KEY_ROW4	ALT1	
ENET2_REF_CLK2	In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.	RMII	ENET2_TX_CLK	ALT1	IO
			GPIO1_IO04	ALT4	
ENET2_REF_CLK_25M	25M Reference Clock	-	ENET2_RX_CLK	ALT1	IO
ENET2_RGMII_RXC	In MII mode, provides a timing reference for RX_EN, RX_DATA[3:0], and RX_ER. In RGMII mode, provides a timing reference for RX_DATA[3:0] and RX_CTL.	RGMII	RGMII2_RXC	ALT0	IO
ENET2_RGMII_TXC	Serial output Ethernet data. Only valid during TX_EN assertion.	RGMII	RGMII2_TXC	ALT0	IO
ENET2_RX_CLK	In MII mode, provides a timing reference for RX_EN, RX_DATA[3:0], and RX_ER. In RGMII mode, provides a timing reference for RX_DATA[3:0] and RX_CTL.	MII	ENET2_RX_CLK	ALT0	IO
ENET2_RX_DATA0	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII / RMII / RGMII	RGMII2_RD0	ALT0	IO
ENET2_RX_DATA1	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII / RMII / RGMII	RGMII2_RD1	ALT0	IO
ENET2_RX_DATA2	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII / RGMII	RGMII2_RD2	ALT0	IO
ENET2_RX_DATA3	Contains the Ethernet input data transferred from the PHY to the	MII / RGMII	RGMII2_RD3	ALT0	IO

Table continues on the next page...

Table 22-2. ENET2 External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	media-access controller when RX_EN is asserted.				
ENET2_RX_EN	Asserting this input indicates the PHY has valid nibbles present on the MII. RX_EN must remain asserted from the first recovered nibble of the frame through to the last nibble. Asserting RX_EN must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal. In RGMII mode, contains RXDV on the rising edge of RX_CLK, and a logical derivative of RX_EV and RX_ER (RX_EV XOR RX_ER) on the falling edge of RX_CLK.	MII / RMII / RGMII	RGMII2_RX_CTL	ALT0	IO
ENET2_RX_ER	When asserted with RXDV, indicates the PHY detects an error in the current frame.	MII / RMII	RGMII2_RXC	ALT1	IO
ENET2_TX_CLK	Input clock, which provides a timing reference for TX_EN, TX_DATA[3:0], and TX_ER.	MII	ENET2_TX_CLK	ALT0	IO
ENET2_TX_DATA0	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RMII / RGMII	RGMII2_TD0	ALT0	IO
ENET2_TX_DATA1	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RMII / RGMII	RGMII2_TD1	ALT0	IO
ENET2_TX_DATA2	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RGMII	RGMII2_TD2	ALT0	IO
ENET2_TX_DATA3	Serial output Ethernet data. Only valid during TX_EN assertion.	MII / RGMII	RGMII2_TD3	ALT0	IO
ENET2_TX_EN	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is deasserted before the first TX_CLK following the final nibble of the frame. In RGMII mode, contains TX_EN on the rising edge of TX_CTL, and a logical derivative of TX_EN and TX_ER (TX_EN XOR TX_ER) on the falling edge of TX_CTL.	MII / RMII / RGMII	RGMII2_TX_CTL	ALT0	IO
ENET2_TX_ER	When asserted for one or more clock cycles while TXEN is also asserted, PHY sends one or more illegal symbols.	MII	RGMII2_TXC	ALT1	IO

22.4 Clocks

The table found here describes the clock sources for ENET. Please see for clock setting, configuration and gating information.

22.5 Memory map/register definition

ENET registers must be read or written with 32-bit accesses. Non-32 bit accesses will terminate with an error.

Reserved bits should be written with 0 and ignored on read. Unused registers read zero and a write has no effect.

This table shows Ethernet registers organization.

Table 22-3. Register map summary

Offset Address	Section	Description
0x0000 – 0x01FF	Configuration	Core control and status registers
0x0200 – 0x03FF	Statistics counters	MIB and Remote Network Monitoring (RFC 2819) registers
0x0400 – 0x0430	1588 control	1588 adjustable timer (TSM) and 1588 frame control
0x0600 – 0x07FC	Capture/Compare block	Registers for the Capture/Compare block

ENET memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_4004	Interrupt Event Register (ENET2_EIR)	32	w1c	0000_0000h	22.5.1/888
20B_4008	Interrupt Mask Register (ENET2_EIMR)	32	R/W	0000_0000h	22.5.2/891
20B_4010	Receive Descriptor Active Register (ENET2_RDAR)	32	R/W	0000_0000h	22.5.3/894
20B_4014	Transmit Descriptor Active Register (ENET2_TDAR)	32	R/W	0000_0000h	22.5.4/894
20B_4024	Ethernet Control Register (ENET2_ECR)	32	R/W	See section	22.5.5/896
20B_4040	MII Management Frame Register (ENET2_MMFR)	32	R/W	0000_0000h	22.5.6/898
20B_4044	MII Speed Control Register (ENET2_MSCR)	32	R/W	0000_0000h	22.5.7/898
20B_4064	MIB Control Register (ENET2_MIBC)	32	R/W	C000_0000h	22.5.8/901
20B_4084	Receive Control Register (ENET2_RCR)	32	R/W	05EE_0001h	22.5.9/902
20B_40C4	Transmit Control Register (ENET2_TCR)	32	R/W	0000_0000h	22.5.10/ 905
20B_40E4	Physical Address Lower Register (ENET2_PALR)	32	R/W	0000_0000h	22.5.11/ 907
20B_40E8	Physical Address Upper Register (ENET2_PAUR)	32	R/W	0000_8808h	22.5.12/ 907

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_40EC	Opcode/Pause Duration Register (ENET2_OPD)	32	R/W	0001_0000h	22.5.13/ 908
20B_40F0	Transmit Interrupt Coalescing Register (ENET2_TXIC)	32	R/W	0000_0000h	22.5.14/ 908
20B_4100	Receive Interrupt Coalescing Register (ENET2_RXIC)	32	R/W	0000_0000h	22.5.15/ 909
20B_4118	Descriptor Individual Upper Address Register (ENET2_IAUR)	32	R/W	0000_0000h	22.5.16/ 910
20B_411C	Descriptor Individual Lower Address Register (ENET2_IALR)	32	R/W	0000_0000h	22.5.17/ 911
20B_4120	Descriptor Group Upper Address Register (ENET2_GAUR)	32	R/W	0000_0000h	22.5.18/ 911
20B_4124	Descriptor Group Lower Address Register (ENET2_GALR)	32	R/W	0000_0000h	22.5.19/ 912
20B_4144	Transmit FIFO Watermark Register (ENET2_TFWR)	32	R/W	0000_0000h	22.5.20/ 912
20B_4180	Receive Descriptor Ring Start Register (ENET2_RDSR)	32	R/W	0000_0000h	22.5.21/ 913
20B_4184	Transmit Buffer Descriptor Ring Start Register (ENET2_TDSR)	32	R/W	0000_0000h	22.5.22/ 914
20B_4188	Maximum Receive Buffer Size Register (ENET2_MRBR)	32	R/W	0000_0000h	22.5.23/ 915
20B_4190	Receive FIFO Section Full Threshold (ENET2_RSFL)	32	R/W	0000_0000h	22.5.24/ 916
20B_4194	Receive FIFO Section Empty Threshold (ENET2_RSEM)	32	R/W	0000_0000h	22.5.25/ 916
20B_4198	Receive FIFO Almost Empty Threshold (ENET2_RAEM)	32	R/W	0000_0004h	22.5.26/ 917
20B_419C	Receive FIFO Almost Full Threshold (ENET2_RAFL)	32	R/W	0000_0004h	22.5.27/ 917
20B_41A0	Transmit FIFO Section Empty Threshold (ENET2_TSEM)	32	R/W	0000_0000h	22.5.28/ 918
20B_41A4	Transmit FIFO Almost Empty Threshold (ENET2_TAEM)	32	R/W	0000_0004h	22.5.29/ 918
20B_41A8	Transmit FIFO Almost Full Threshold (ENET2_TAFL)	32	R/W	0000_0008h	22.5.30/ 919
20B_41AC	Transmit Inter-Packet Gap (ENET2_TIPG)	32	R/W	0000_000Ch	22.5.31/ 919
20B_41B0	Frame Truncation Length (ENET2_FTRL)	32	R/W	0000_07FFh	22.5.32/ 920
20B_41C0	Transmit Accelerator Function Configuration (ENET2_TACC)	32	R/W	0000_0000h	22.5.33/ 920
20B_41C4	Receive Accelerator Function Configuration (ENET2_RACC)	32	R/W	0000_0000h	22.5.34/ 921

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_4200	Reserved Statistic Register (ENET2_RMON_T_DROP)	32	R	0000_0000h	22.5.35/ 922
20B_4204	Tx Packet Count Statistic Register (ENET2_RMON_T_PACKETS)	32	R	0000_0000h	22.5.36/ 923
20B_4208	Tx Broadcast Packets Statistic Register (ENET2_RMON_T_BC_PKT)	32	R	0000_0000h	22.5.37/ 923
20B_420C	Tx Multicast Packets Statistic Register (ENET2_RMON_T_MC_PKT)	32	R	0000_0000h	22.5.38/ 924
20B_4210	Tx Packets with CRC/Align Error Statistic Register (ENET2_RMON_T_CRC_ALIGN)	32	R	0000_0000h	22.5.39/ 924
20B_4214	Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET2_RMON_T_UNDERSIZE)	32	R	0000_0000h	22.5.40/ 924
20B_4218	Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET2_RMON_T_OVERSIZE)	32	R	0000_0000h	22.5.41/ 925
20B_421C	Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET2_RMON_T_FRAG)	32	R	0000_0000h	22.5.42/ 925
20B_4220	Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET2_RMON_T_JAB)	32	R	0000_0000h	22.5.43/ 926
20B_4224	Tx Collision Count Statistic Register (ENET2_RMON_T_COL)	32	R	0000_0000h	22.5.44/ 926
20B_4228	Tx 64-Byte Packets Statistic Register (ENET2_RMON_T_P64)	32	R	0000_0000h	22.5.45/ 927
20B_422C	Tx 65- to 127-byte Packets Statistic Register (ENET2_RMON_T_P65TO127)	32	R	0000_0000h	22.5.46/ 927
20B_4230	Tx 128- to 255-byte Packets Statistic Register (ENET2_RMON_T_P128TO255)	32	R	0000_0000h	22.5.47/ 928
20B_4234	Tx 256- to 511-byte Packets Statistic Register (ENET2_RMON_T_P256TO511)	32	R	0000_0000h	22.5.48/ 928
20B_4238	Tx 512- to 1023-byte Packets Statistic Register (ENET2_RMON_T_P512TO1023)	32	R	0000_0000h	22.5.49/ 929
20B_423C	Tx 1024- to 2047-byte Packets Statistic Register (ENET2_RMON_T_P1024TO2047)	32	R	0000_0000h	22.5.50/ 929
20B_4240	Tx Packets Greater Than 2048 Bytes Statistic Register (ENET2_RMON_T_P_GTE2048)	32	R	0000_0000h	22.5.51/ 930
20B_4244	Tx Octets Statistic Register (ENET2_RMON_T_OCTETS)	32	R	0000_0000h	22.5.52/ 930
20B_4248	Reserved Statistic Register (ENET2_IEEE_T_DROP)	32	R	0000_0000h	22.5.53/ 930
20B_424C	Frames Transmitted OK Statistic Register (ENET2_IEEE_T_FRAME_OK)	32	R	0000_0000h	22.5.54/ 931
20B_4250	Frames Transmitted with Single Collision Statistic Register (ENET2_IEEE_T_1COL)	32	R	0000_0000h	22.5.55/ 931
20B_4254	Frames Transmitted with Multiple Collisions Statistic Register (ENET2_IEEE_T_MCOL)	32	R	0000_0000h	22.5.56/ 932

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_4258	Frames Transmitted after Deferral Delay Statistic Register (ENET2_IEEE_T_DEF)	32	R	0000_0000h	22.5.57/ 932
20B_425C	Frames Transmitted with Late Collision Statistic Register (ENET2_IEEE_T_LCOL)	32	R	0000_0000h	22.5.58/ 932
20B_4260	Frames Transmitted with Excessive Collisions Statistic Register (ENET2_IEEE_T_EXCOL)	32	R	0000_0000h	22.5.59/ 933
20B_4264	Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET2_IEEE_T_MACERR)	32	R	0000_0000h	22.5.60/ 933
20B_4268	Frames Transmitted with Carrier Sense Error Statistic Register (ENET2_IEEE_T_CSERR)	32	R	0000_0000h	22.5.61/ 934
20B_426C	Reserved Statistic Register (ENET2_IEEE_T_SQE)	32	R (reads 0)	0000_0000h	22.5.62/ 934
20B_4270	Flow Control Pause Frames Transmitted Statistic Register (ENET2_IEEE_T_FDXFC)	32	R	0000_0000h	22.5.63/ 934
20B_4274	Octet Count for Frames Transmitted w/o Error Statistic Register (ENET2_IEEE_T_OCTETS_OK)	32	R	0000_0000h	22.5.64/ 935
20B_4284	Rx Packet Count Statistic Register (ENET2_RMON_R_PACKETS)	32	R	0000_0000h	22.5.65/ 935
20B_4288	Rx Broadcast Packets Statistic Register (ENET2_RMON_R_BC_PKT)	32	R	0000_0000h	22.5.66/ 936
20B_428C	Rx Multicast Packets Statistic Register (ENET2_RMON_R_MC_PKT)	32	R	0000_0000h	22.5.67/ 936
20B_4290	Rx Packets with CRC/Align Error Statistic Register (ENET2_RMON_R_CRC_ALIGN)	32	R	0000_0000h	22.5.68/ 936
20B_4294	Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET2_RMON_R_UNDERSIZE)	32	R	0000_0000h	22.5.69/ 937
20B_4298	Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET2_RMON_R_OVERSIZE)	32	R	0000_0000h	22.5.70/ 937
20B_429C	Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET2_RMON_R_FRAG)	32	R	0000_0000h	22.5.71/ 938
20B_42A0	Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET2_RMON_R_JAB)	32	R	0000_0000h	22.5.72/ 938
20B_42A4	Reserved Statistic Register (ENET2_RMON_R_RESVD_0)	32	R (reads 0)	0000_0000h	22.5.73/ 938
20B_42A8	Rx 64-Byte Packets Statistic Register (ENET2_RMON_R_P64)	32	R	0000_0000h	22.5.74/ 939
20B_42AC	Rx 65- to 127-Byte Packets Statistic Register (ENET2_RMON_R_P65TO127)	32	R	0000_0000h	22.5.75/ 939
20B_42B0	Rx 128- to 255-Byte Packets Statistic Register (ENET2_RMON_R_P128TO255)	32	R	0000_0000h	22.5.76/ 940
20B_42B4	Rx 256- to 511-Byte Packets Statistic Register (ENET2_RMON_R_P256TO511)	32	R	0000_0000h	22.5.77/ 940
20B_42B8	Rx 512- to 1023-Byte Packets Statistic Register (ENET2_RMON_R_P512TO1023)	32	R	0000_0000h	22.5.78/ 940

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_42BC	Rx 1024- to 2047-Byte Packets Statistic Register (ENET2_RMON_R_P1024TO2047)	32	R	0000_0000h	22.5.79/ 941
20B_42C0	Rx Packets Greater than 2048 Bytes Statistic Register (ENET2_RMON_R_P_GTE2048)	32	R	0000_0000h	22.5.80/ 941
20B_42C4	Rx Octets Statistic Register (ENET2_RMON_R_OCTETS)	32	R	0000_0000h	22.5.81/ 942
20B_42C8	Frames not Counted Correctly Statistic Register (ENET2_IEEE_R_DROP)	32	R	0000_0000h	22.5.82/ 942
20B_42CC	Frames Received OK Statistic Register (ENET2_IEEE_R_FRAME_OK)	32	R	0000_0000h	22.5.83/ 942
20B_42D0	Frames Received with CRC Error Statistic Register (ENET2_IEEE_R_CRC)	32	R	0000_0000h	22.5.84/ 943
20B_42D4	Frames Received with Alignment Error Statistic Register (ENET2_IEEE_R_ALIGN)	32	R	0000_0000h	22.5.85/ 943
20B_42D8	Receive FIFO Overflow Count Statistic Register (ENET2_IEEE_R_MACERR)	32	R	0000_0000h	22.5.86/ 944
20B_42DC	Flow Control Pause Frames Received Statistic Register (ENET2_IEEE_R_FDXFC)	32	R	0000_0000h	22.5.87/ 944
20B_42E0	Octet Count for Frames Received without Error Statistic Register (ENET2_IEEE_R_OCTETS_OK)	32	R	0000_0000h	22.5.88/ 944
20B_4400	Adjustable Timer Control Register (ENET2_ATCR)	32	R/W	0000_0000h	22.5.89/ 945
20B_4404	Timer Value Register (ENET2_ATVR)	32	R/W	0000_0000h	22.5.90/ 947
20B_4408	Timer Offset Register (ENET2_ATOFF)	32	R/W	0000_0000h	22.5.91/ 947
20B_440C	Timer Period Register (ENET2_ATPER)	32	R/W	3B9A_CA00h	22.5.92/ 947
20B_4410	Timer Correction Register (ENET2_ATCOR)	32	R/W	0000_0000h	22.5.93/ 948
20B_4414	Time-Stamping Clock Period Register (ENET2_ATINC)	32	R/W	0000_0000h	22.5.94/ 948
20B_4418	Timestamp of Last Transmitted Frame (ENET2_ATSTMP)	32	R	0000_0000h	22.5.95/ 949
20B_4604	Timer Global Status Register (ENET2_TGSR)	32	R/W	0000_0000h	22.5.96/ 949
20B_4608	Timer Control Status Register (ENET2_TCSR0)	32	R/W	0000_0000h	22.5.97/ 950
20B_460C	Timer Compare Capture Register (ENET2_TCCR0)	32	R/W	0000_0000h	22.5.98/ 952
20B_4610	Timer Control Status Register (ENET2_TCSR1)	32	R/W	0000_0000h	22.5.97/ 950
20B_4614	Timer Compare Capture Register (ENET2_TCCR1)	32	R/W	0000_0000h	22.5.98/ 952

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_4618	Timer Control Status Register (ENET2_TCSR2)	32	R/W	0000_0000h	22.5.97/ 950
20B_461C	Timer Compare Capture Register (ENET2_TCCR2)	32	R/W	0000_0000h	22.5.98/ 952
20B_4620	Timer Control Status Register (ENET2_TCSR3)	32	R/W	0000_0000h	22.5.97/ 950
20B_4624	Timer Compare Capture Register (ENET2_TCCR3)	32	R/W	0000_0000h	22.5.98/ 952
218_8004	Interrupt Event Register (ENET1_EIR)	32	w1c	0000_0000h	22.5.1/888
218_8008	Interrupt Mask Register (ENET1_EIMR)	32	R/W	0000_0000h	22.5.2/891
218_8010	Receive Descriptor Active Register (ENET1_RDAR)	32	R/W	0000_0000h	22.5.3/894
218_8014	Transmit Descriptor Active Register (ENET1_TDAR)	32	R/W	0000_0000h	22.5.4/894
218_8024	Ethernet Control Register (ENET1_ECR)	32	R/W	See section	22.5.5/896
218_8040	MII Management Frame Register (ENET1_MMFR)	32	R/W	0000_0000h	22.5.6/898
218_8044	MII Speed Control Register (ENET1_MSCR)	32	R/W	0000_0000h	22.5.7/898
218_8064	MIB Control Register (ENET1_MIBC)	32	R/W	C000_0000h	22.5.8/901
218_8084	Receive Control Register (ENET1_RCR)	32	R/W	05EE_0001h	22.5.9/902
218_80C4	Transmit Control Register (ENET1_TCR)	32	R/W	0000_0000h	22.5.10/ 905
218_80E4	Physical Address Lower Register (ENET1_PALR)	32	R/W	0000_0000h	22.5.11/ 907
218_80E8	Physical Address Upper Register (ENET1_PAUR)	32	R/W	0000_8808h	22.5.12/ 907
218_80EC	Opcode/Pause Duration Register (ENET1_OPD)	32	R/W	0001_0000h	22.5.13/ 908
218_80F0	Transmit Interrupt Coalescing Register (ENET1_TXIC)	32	R/W	0000_0000h	22.5.14/ 908
218_8100	Receive Interrupt Coalescing Register (ENET1_RXIC)	32	R/W	0000_0000h	22.5.15/ 909
218_8118	Descriptor Individual Upper Address Register (ENET1_IAUR)	32	R/W	0000_0000h	22.5.16/ 910
218_811C	Descriptor Individual Lower Address Register (ENET1_IALR)	32	R/W	0000_0000h	22.5.17/ 911
218_8120	Descriptor Group Upper Address Register (ENET1_GAUR)	32	R/W	0000_0000h	22.5.18/ 911
218_8124	Descriptor Group Lower Address Register (ENET1_GALR)	32	R/W	0000_0000h	22.5.19/ 912
218_8144	Transmit FIFO Watermark Register (ENET1_TFWR)	32	R/W	0000_0000h	22.5.20/ 912
218_8180	Receive Descriptor Ring Start Register (ENET1_RDSR)	32	R/W	0000_0000h	22.5.21/ 913
218_8184	Transmit Buffer Descriptor Ring Start Register (ENET1_TDSR)	32	R/W	0000_0000h	22.5.22/ 914

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_8188	Maximum Receive Buffer Size Register (ENET1_MRBR)	32	R/W	0000_0000h	22.5.23/ 915
218_8190	Receive FIFO Section Full Threshold (ENET1_RSFL)	32	R/W	0000_0000h	22.5.24/ 916
218_8194	Receive FIFO Section Empty Threshold (ENET1_RSEM)	32	R/W	0000_0000h	22.5.25/ 916
218_8198	Receive FIFO Almost Empty Threshold (ENET1_RAEM)	32	R/W	0000_0004h	22.5.26/ 917
218_819C	Receive FIFO Almost Full Threshold (ENET1_RAFL)	32	R/W	0000_0004h	22.5.27/ 917
218_81A0	Transmit FIFO Section Empty Threshold (ENET1_TSEM)	32	R/W	0000_0000h	22.5.28/ 918
218_81A4	Transmit FIFO Almost Empty Threshold (ENET1_TAEM)	32	R/W	0000_0004h	22.5.29/ 918
218_81A8	Transmit FIFO Almost Full Threshold (ENET1_TAFL)	32	R/W	0000_0008h	22.5.30/ 919
218_81AC	Transmit Inter-Packet Gap (ENET1_TIPG)	32	R/W	0000_000Ch	22.5.31/ 919
218_81B0	Frame Truncation Length (ENET1_FTRL)	32	R/W	0000_07FFh	22.5.32/ 920
218_81C0	Transmit Accelerator Function Configuration (ENET1_TACC)	32	R/W	0000_0000h	22.5.33/ 920
218_81C4	Receive Accelerator Function Configuration (ENET1_RACC)	32	R/W	0000_0000h	22.5.34/ 921
218_8200	Reserved Statistic Register (ENET1_RMON_T_DROP)	32	R	0000_0000h	22.5.35/ 922
218_8204	Tx Packet Count Statistic Register (ENET1_RMON_T_PACKETS)	32	R	0000_0000h	22.5.36/ 923
218_8208	Tx Broadcast Packets Statistic Register (ENET1_RMON_T_BC_PKT)	32	R	0000_0000h	22.5.37/ 923
218_820C	Tx Multicast Packets Statistic Register (ENET1_RMON_T_MC_PKT)	32	R	0000_0000h	22.5.38/ 924
218_8210	Tx Packets with CRC/Align Error Statistic Register (ENET1_RMON_T_CRC_ALIGN)	32	R	0000_0000h	22.5.39/ 924
218_8214	Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET1_RMON_T_UNDERSIZE)	32	R	0000_0000h	22.5.40/ 924
218_8218	Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET1_RMON_T_OVERSIZE)	32	R	0000_0000h	22.5.41/ 925
218_821C	Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET1_RMON_T_FRAG)	32	R	0000_0000h	22.5.42/ 925
218_8220	Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET1_RMON_T_JAB)	32	R	0000_0000h	22.5.43/ 926
218_8224	Tx Collision Count Statistic Register (ENET1_RMON_T_COL)	32	R	0000_0000h	22.5.44/ 926

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_8228	Tx 64-Byte Packets Statistic Register (ENET1_RMON_T_P64)	32	R	0000_0000h	22.5.45/ 927
218_822C	Tx 65- to 127-byte Packets Statistic Register (ENET1_RMON_T_P65TO127)	32	R	0000_0000h	22.5.46/ 927
218_8230	Tx 128- to 255-byte Packets Statistic Register (ENET1_RMON_T_P128TO255)	32	R	0000_0000h	22.5.47/ 928
218_8234	Tx 256- to 511-byte Packets Statistic Register (ENET1_RMON_T_P256TO511)	32	R	0000_0000h	22.5.48/ 928
218_8238	Tx 512- to 1023-byte Packets Statistic Register (ENET1_RMON_T_P512TO1023)	32	R	0000_0000h	22.5.49/ 929
218_823C	Tx 1024- to 2047-byte Packets Statistic Register (ENET1_RMON_T_P1024TO2047)	32	R	0000_0000h	22.5.50/ 929
218_8240	Tx Packets Greater Than 2048 Bytes Statistic Register (ENET1_RMON_T_P_GTE2048)	32	R	0000_0000h	22.5.51/ 930
218_8244	Tx Octets Statistic Register (ENET1_RMON_T_OCTETS)	32	R	0000_0000h	22.5.52/ 930
218_8248	Reserved Statistic Register (ENET1_IEEE_T_DROP)	32	R	0000_0000h	22.5.53/ 930
218_824C	Frames Transmitted OK Statistic Register (ENET1_IEEE_T_FRAME_OK)	32	R	0000_0000h	22.5.54/ 931
218_8250	Frames Transmitted with Single Collision Statistic Register (ENET1_IEEE_T_1COL)	32	R	0000_0000h	22.5.55/ 931
218_8254	Frames Transmitted with Multiple Collisions Statistic Register (ENET1_IEEE_T_MCOL)	32	R	0000_0000h	22.5.56/ 932
218_8258	Frames Transmitted after Deferral Delay Statistic Register (ENET1_IEEE_T_DEF)	32	R	0000_0000h	22.5.57/ 932
218_825C	Frames Transmitted with Late Collision Statistic Register (ENET1_IEEE_T_LCOL)	32	R	0000_0000h	22.5.58/ 932
218_8260	Frames Transmitted with Excessive Collisions Statistic Register (ENET1_IEEE_T_EXCOL)	32	R	0000_0000h	22.5.59/ 933
218_8264	Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET1_IEEE_T_MACERR)	32	R	0000_0000h	22.5.60/ 933
218_8268	Frames Transmitted with Carrier Sense Error Statistic Register (ENET1_IEEE_T_CSERR)	32	R	0000_0000h	22.5.61/ 934
218_826C	Reserved Statistic Register (ENET1_IEEE_T_SQE)	32	R (reads 0)	0000_0000h	22.5.62/ 934
218_8270	Flow Control Pause Frames Transmitted Statistic Register (ENET1_IEEE_T_FDXFC)	32	R	0000_0000h	22.5.63/ 934
218_8274	Octet Count for Frames Transmitted w/o Error Statistic Register (ENET1_IEEE_T_OCTETS_OK)	32	R	0000_0000h	22.5.64/ 935
218_8284	Rx Packet Count Statistic Register (ENET1_RMON_R_PACKETS)	32	R	0000_0000h	22.5.65/ 935
218_8288	Rx Broadcast Packets Statistic Register (ENET1_RMON_R_BC_PKT)	32	R	0000_0000h	22.5.66/ 936

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_828C	Rx Multicast Packets Statistic Register (ENET1_RMON_R_MC_PKT)	32	R	0000_0000h	22.5.67/ 936
218_8290	Rx Packets with CRC/Align Error Statistic Register (ENET1_RMON_R_CRC_ALIGN)	32	R	0000_0000h	22.5.68/ 936
218_8294	Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET1_RMON_R_UNDERSIZE)	32	R	0000_0000h	22.5.69/ 937
218_8298	Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET1_RMON_R_OVERSIZE)	32	R	0000_0000h	22.5.70/ 937
218_829C	Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET1_RMON_R_FRAG)	32	R	0000_0000h	22.5.71/ 938
218_82A0	Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET1_RMON_R_JAB)	32	R	0000_0000h	22.5.72/ 938
218_82A4	Reserved Statistic Register (ENET1_RMON_R_RESVD_0)	32	R (reads 0)	0000_0000h	22.5.73/ 938
218_82A8	Rx 64-Byte Packets Statistic Register (ENET1_RMON_R_P64)	32	R	0000_0000h	22.5.74/ 939
218_82AC	Rx 65- to 127-Byte Packets Statistic Register (ENET1_RMON_R_P65TO127)	32	R	0000_0000h	22.5.75/ 939
218_82B0	Rx 128- to 255-Byte Packets Statistic Register (ENET1_RMON_R_P128TO255)	32	R	0000_0000h	22.5.76/ 940
218_82B4	Rx 256- to 511-Byte Packets Statistic Register (ENET1_RMON_R_P256TO511)	32	R	0000_0000h	22.5.77/ 940
218_82B8	Rx 512- to 1023-Byte Packets Statistic Register (ENET1_RMON_R_P512TO1023)	32	R	0000_0000h	22.5.78/ 940
218_82BC	Rx 1024- to 2047-Byte Packets Statistic Register (ENET1_RMON_R_P1024TO2047)	32	R	0000_0000h	22.5.79/ 941
218_82C0	Rx Packets Greater than 2048 Bytes Statistic Register (ENET1_RMON_R_P_GTE2048)	32	R	0000_0000h	22.5.80/ 941
218_82C4	Rx Octets Statistic Register (ENET1_RMON_R_OCTETS)	32	R	0000_0000h	22.5.81/ 942
218_82C8	Frames not Counted Correctly Statistic Register (ENET1_IEEE_R_DROP)	32	R	0000_0000h	22.5.82/ 942
218_82CC	Frames Received OK Statistic Register (ENET1_IEEE_R_FRAME_OK)	32	R	0000_0000h	22.5.83/ 942
218_82D0	Frames Received with CRC Error Statistic Register (ENET1_IEEE_R_CRC)	32	R	0000_0000h	22.5.84/ 943
218_82D4	Frames Received with Alignment Error Statistic Register (ENET1_IEEE_R_ALIGN)	32	R	0000_0000h	22.5.85/ 943
218_82D8	Receive FIFO Overflow Count Statistic Register (ENET1_IEEE_R_MACERR)	32	R	0000_0000h	22.5.86/ 944
218_82DC	Flow Control Pause Frames Received Statistic Register (ENET1_IEEE_R_FDXFC)	32	R	0000_0000h	22.5.87/ 944
218_82E0	Octet Count for Frames Received without Error Statistic Register (ENET1_IEEE_R_OCTETS_OK)	32	R	0000_0000h	22.5.88/ 944

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_8400	Adjustable Timer Control Register (ENET1_ATCR)	32	R/W	0000_0000h	22.5.89/ 945
218_8404	Timer Value Register (ENET1_ATVR)	32	R/W	0000_0000h	22.5.90/ 947
218_8408	Timer Offset Register (ENET1_ATOFF)	32	R/W	0000_0000h	22.5.91/ 947
218_840C	Timer Period Register (ENET1_ATPER)	32	R/W	3B9A_CA00h	22.5.92/ 947
218_8410	Timer Correction Register (ENET1_ATCOR)	32	R/W	0000_0000h	22.5.93/ 948
218_8414	Time-Stamping Clock Period Register (ENET1_ATINC)	32	R/W	0000_0000h	22.5.94/ 948
218_8418	Timestamp of Last Transmitted Frame (ENET1_ATSTMP)	32	R	0000_0000h	22.5.95/ 949
218_8604	Timer Global Status Register (ENET1_TGSR)	32	R/W	0000_0000h	22.5.96/ 949
218_8608	Timer Control Status Register (ENET1_TCSR0)	32	R/W	0000_0000h	22.5.97/ 950
218_860C	Timer Compare Capture Register (ENET1_TCCR0)	32	R/W	0000_0000h	22.5.98/ 952
218_8610	Timer Control Status Register (ENET1_TCSR1)	32	R/W	0000_0000h	22.5.97/ 950
218_8614	Timer Compare Capture Register (ENET1_TCCR1)	32	R/W	0000_0000h	22.5.98/ 952
218_8618	Timer Control Status Register (ENET1_TCSR2)	32	R/W	0000_0000h	22.5.97/ 950
218_861C	Timer Compare Capture Register (ENET1_TCCR2)	32	R/W	0000_0000h	22.5.98/ 952
218_8620	Timer Control Status Register (ENET1_TCSR3)	32	R/W	0000_0000h	22.5.97/ 950
218_8624	Timer Compare Capture Register (ENET1_TCCR3)	32	R/W	0000_0000h	22.5.98/ 952

22.5.1 Interrupt Event Register (ENETx_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

NOTE

TxBD[INT] and RxBD[INT] must be set to 1 to allow setting the corresponding EIR register flags in enhanced mode,

ENET_ECR[EN1588] = 1. Legacy mode does not require these flags to be enabled.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS_TIMER															
W	w1c	0		0	0			0	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_EIR field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 BABR	Babbling Receive Error Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
29 BABT	Babbling Transmit Error Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused when a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
28 GRA	Graceful Stop Complete This interrupt is asserted after the transmitter is put into a pause state after completion of the frame currently being transmitted. See Graceful Transmit Stop (GTS) for conditions that lead to graceful stop. **NOTE:** The GRA interrupt is asserted only when the TX transitions into the stopped state. If this bit is cleared by writing 1 and the TX is still stopped, the bit is not set again.
27 TXF	Transmit Frame Interrupt Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
26 TXB	Transmit Buffer Interrupt Indicates a transmit buffer descriptor has been updated.

Table continues on the next page...

ENETx_EIR field descriptions (continued)

Field	Description
25 RXF	Receive Frame Interrupt Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
24 RXB	Receive Buffer Interrupt Indicates a receive buffer descriptor is not the last in the frame has been updated.
23 MII	MII Interrupt. Indicates that the MII has completed the data transfer requested.
22 EBERR	Ethernet Bus Error Indicates a system bus error occurred when a uDMA transaction is underway. When this bit is set, ECR[ETHEREN] is cleared, halting frame processing by the MAC. When this occurs, software must ensure proper actions, possibly resetting the system, to resume normal operation.
21 LC	Late Collision Indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
20 RL	Collision Retry Limit Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half-duplex mode.
19 UN	Transmit FIFO Underrun Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
18 PLR	Payload Receive Error Indicates a frame was received with a payload length error. See Frame Length/Type Verification: Payload Length Check for more information.
17 WAKEUP	Node Wakeup Request Indication Read-only status bit to indicate that a magic packet has been detected. Will act only if ECR[MAGICEN] is set.
16 TS_AVAIL	Transmit Timestamp Available Indicates that the timestamp of the last transmitted timing frame is available in the ATSTMP register.
15 TS_TIMER	Timestamp Timer The adjustable timer reached the period event. A period event interrupt can be generated if ATCR[PEREN] is set and the timer wraps according to the periodic setting in the ATPER register. Set the timer period value before setting ATCR[PEREN].
14–13 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
12 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
11–9 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
8 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

Table continues on the next page...

ENETx_EIR field descriptions (continued)

Field	Description
Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

22.5.2 Interrupt Mask Register (ENETx_EIMR)

EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR field (write 1 to clear) or a 0 is written to the EIMR field.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_EIMR field descriptions

Field	Description
31 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30 BABR	BABR Interrupt Mask Corresponds to interrupt source EIR[BABR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
29 BABT	BABT Interrupt Mask Corresponds to interrupt source EIR[BABT] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABT field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

ENETx_EIMR field descriptions (continued)

Field	Description
	0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
28 GRA	GRA Interrupt Mask Corresponds to interrupt source EIR[GRA] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR GRA field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
27 TXF	TXF Interrupt Mask Corresponds to interrupt source EIR[TXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
26 TXB	TXB Interrupt Mask Corresponds to interrupt source EIR[TXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
25 RXF	RXF Interrupt Mask Corresponds to interrupt source EIR[RXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
24 RXB	RXB Interrupt Mask Corresponds to interrupt source EIR[RXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXB field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
23 MII	MII Interrupt Mask Corresponds to interrupt source EIR[MII] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR MII field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
22 EBERR	EBERR Interrupt Mask Corresponds to interrupt source EIR[EBERR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR EBERR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

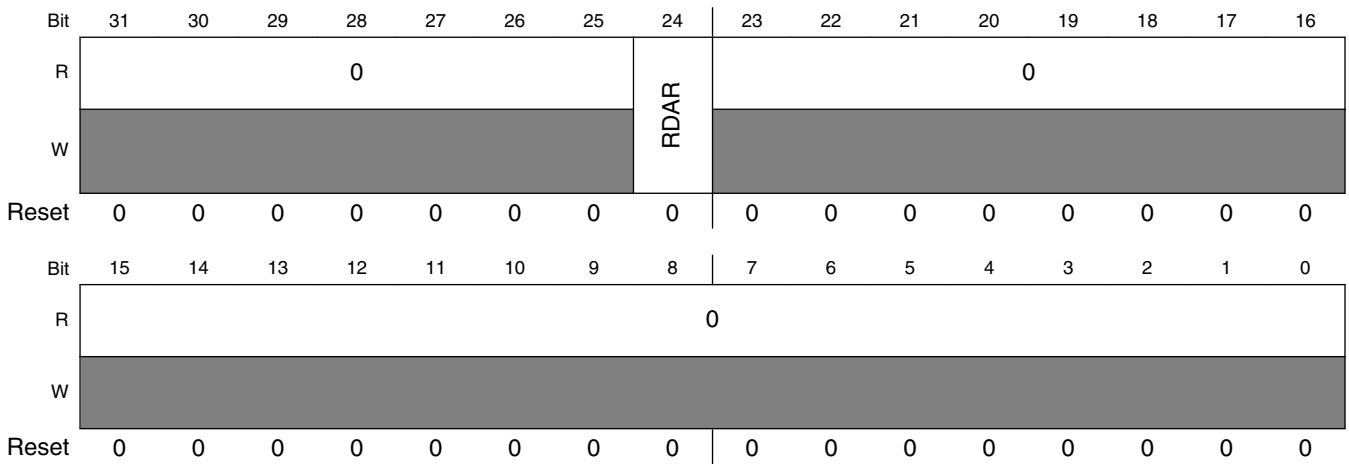
ENETx_EIMR field descriptions (continued)

Field	Description
21 LC	LC Interrupt Mask Corresponds to interrupt source EIR[LC] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR LC field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
20 RL	RL Interrupt Mask Corresponds to interrupt source EIR[RL] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
19 UN	UN Interrupt Mask Corresponds to interrupt source EIR[UN] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR UN field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
18 PLR	PLR Interrupt Mask Corresponds to interrupt source EIR[PLR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR PLR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
17 WAKEUP	WAKEUP Interrupt Mask Corresponds to interrupt source EIR[WAKEUP] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR WAKEUP field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
16 TS_AVAIL	TS_AVAIL Interrupt Mask Corresponds to interrupt source EIR[TS_AVAIL] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_AVAIL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
15 TS_TIMER	TS_TIMER Interrupt Mask Corresponds to interrupt source EIR[TS_TIMER] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_TIMER field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
14–13 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
12 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
11–9 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
8 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

22.5.3 Receive Descriptor Active Register (ENETx_RDAR)

RDAR is a command register, written by the user, to indicate that the receive descriptor ring has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: Base address + 10h offset



ENETx_RDAR field descriptions

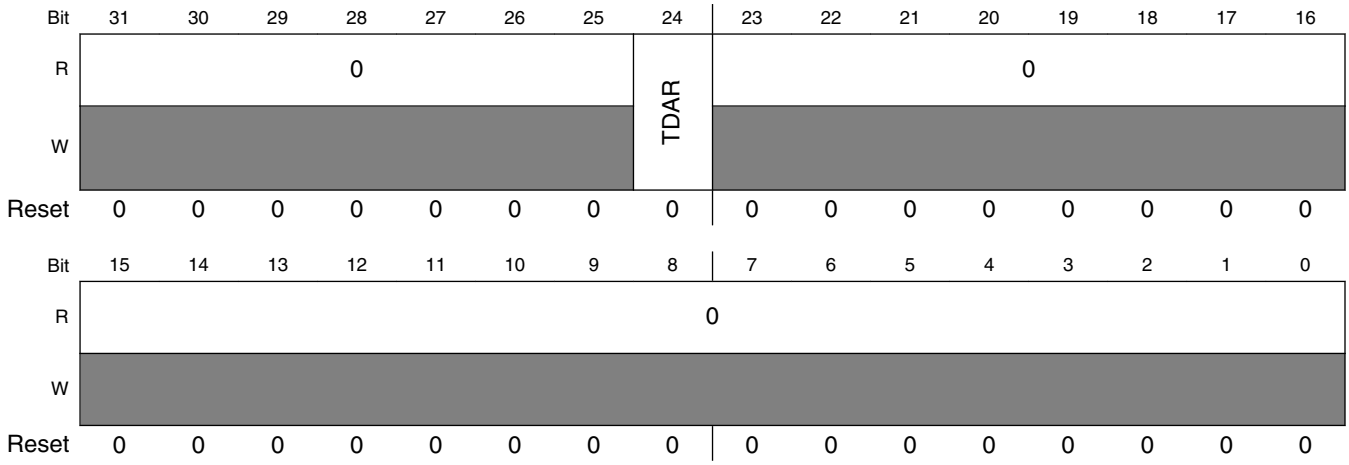
Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.5.4 Transmit Descriptor Active Register (ENETx_TDAR)

The TDAR is a command register that the user writes to indicate that the transmit descriptor ring has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

Address: Base address + 14h offset



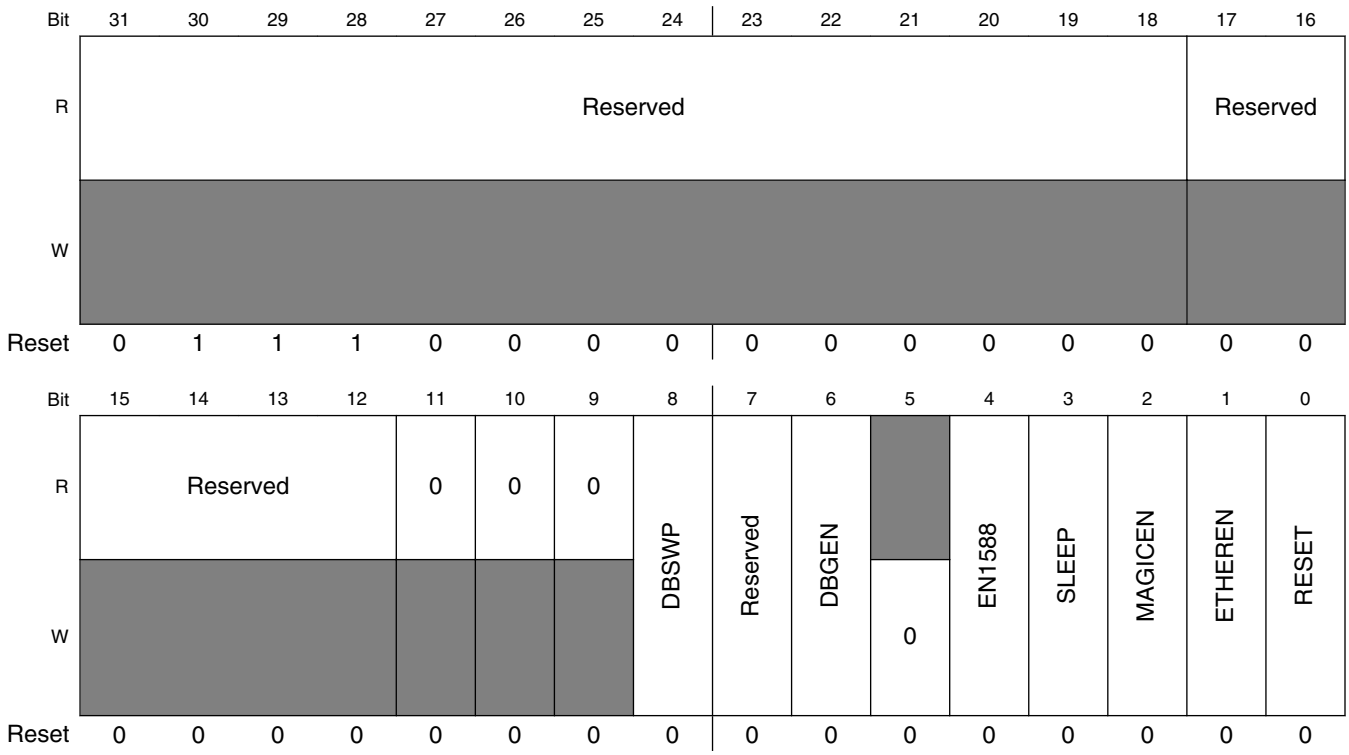
ENETx_TDAR field descriptions

Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.5.5 Ethernet Control Register (ENETx_ECR)

ECR is a read/write user register, though hardware may also alter fields in this register. It controls many of the high level features of the Ethernet MAC, including legacy FEC support through the EN1588 field.

Address: Base address + 24h offset



ENETx_ECR field descriptions

Field	Description
31–18 Reserved	This field is reserved. This field must be set to 01110000000000b.
17–12 Reserved	This field is reserved. This field must be set to 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DBSWP	Descriptor Byte Swapping Enable Swaps the byte locations of the buffer descriptors. **NOTE:** This field must be written to 1 after reset.

Table continues on the next page...

ENETx_ECR field descriptions (continued)

Field	Description
	0 The buffer descriptor bytes are not swapped to support big-endian devices. 1 The buffer descriptor bytes are swapped to support little-endian devices.
7 Reserved	This field is reserved. This field must be set to zero.
6 DBGEN	Debug Enable Enables the MAC to enter hardware freeze mode when the device enters debug mode. 0 MAC continues operation in debug mode. 1 MAC enters hardware freeze mode when the processor is in debug mode.
5 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
4 EN1588	EN1588 Enable Enables enhanced functionality of the MAC. 0 Legacy FEC buffer descriptors and functions enabled. 1 Enhanced frame time-stamping functions enabled.
3 SLEEP	Sleep Mode Enable 0 Normal operating mode. 1 Sleep mode.
2 MAGICEN	Magic Packet Detection Enable Enables/disables magic packet detection. **NOTE:** MAGICEN is relevant only if the SLEEP field is set. If MAGICEN is set, changing the SLEEP field enables/disables sleep mode and magic packet detection. 0 Magic detection logic disabled. 1 The MAC core detects magic packets and asserts EIR[WAKEUP] when a frame is detected.
1 ETHEREN	Ethernet Enable Enables/disables the Ethernet MAC. When the MAC is disabled, the buffer descriptors for an aborted transmit frame are not updated. The uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware clears this field under the following conditions: • RESET is set by software • An error condition causes the EBERR field to set. **NOTE:** • ETHEREN must be set at the very last step during ENET configuration/setup/initialization, only *after* all other ENET-related registers have been configured. • If ETHEREN is cleared to 0 by software then next time ETHEREN is set, the EIR interrupts must cleared to 0 due to previous pending interrupts. 0 Reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. 1 MAC is enabled, and reception and transmission are possible.
0 RESET	Ethernet MAC Reset When this field is set, it clears the ETHEREN field.

22.5.6 MII Management Frame Register (ENETx_MMFR)

Writing to MMFR triggers a management frame transaction to the PHY device unless MSCR is programmed to zero.

If MSCR is changed from zero to non-zero during a write to MMFR, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the EIR[MII] interrupt indication to avoid writing to the MMFR register while frame generation is in progress.

Address: Base address + 40h offset

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

ENETx_MMFR field descriptions

Field	Description
31–30 ST	Start Of Frame Delimiter See Table 22-41 (Clause 22) or Table 22-43 (Clause 45) for correct value.
29–28 OP	Operation Code See Table 22-41 (Clause 22) or Table 22-43 (Clause 45) for correct value.
27–23 PA	PHY Address See Table 22-41 (Clause 22) or Table 22-43 (Clause 45) for correct value.
22–18 RA	Register Address See Table 22-41 (Clause 22) or Table 22-43 (Clause 45) for correct value.
17–16 TA	Turn Around This field must be programmed to 10 to generate a valid MII management frame.
DATA	Management Frame Data This is the field for data to be written to or read from the PHY register.

22.5.7 MII Speed Control Register (ENETx_MSCR)

MSCR provides control of the MII clock (MDC pin) frequency and allows a preamble drop on the MII management frame.

The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be cleared to turn off MDC. The MDC signal generated has a 50% duty cycle except when MII_SPEED changes during operation. This change takes effect following a rising or falling edge of MDC.

For example, if the internal module clock (i.e., IPS bus clock) is 25 MHz, programming MII_SPEED to 0x4 results in an MDC as given in the following equation:

MII clock frequency = 25 MHz / ((4 + 1) x 2) = 2.5 MHz

The following table shows the optimum values for MII_SPEED as a function of IPS bus clock frequency.

Table 22-4. Programming Examples for MSCR

Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
25 MHz	0x4	2.50 MHz
33 MHz	0x6	2.36 MHz
40 MHz	0x7	2.50 MHz
50 MHz	0x9	2.50 MHz
66 MHz	0xD	2.36 MHz

Address: Base address + 44h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_MSCR field descriptions

Field	Description
31–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 HOLDTIME	Hold time On MDIO Output IEEE802.3 clause 22 defines a minimum of 10 ns for the hold time on the MDIO output. Depending on the host bus frequency, the setting may need to be increased. 000 1 internal module clock cycle 001 2 internal module clock cycles

Table continues on the next page...

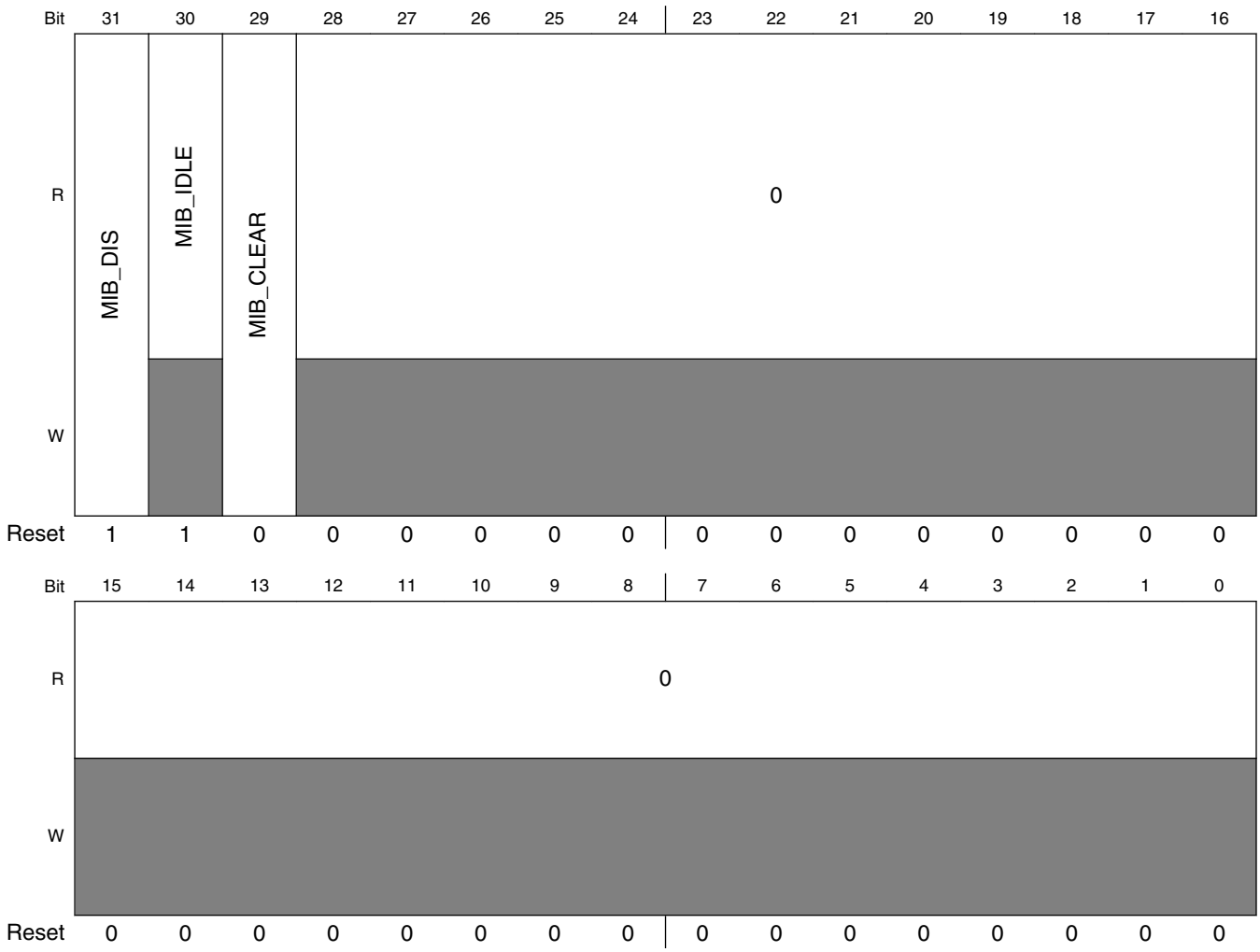
ENETx_MSCR field descriptions (continued)

Field	Description
	010 3 internal module clock cycles 111 8 internal module clock cycles
7 DIS_PRE	Disable Preamble Enables/disables prepending a preamble to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY devices do not require it. 0 Preamble enabled. 1 Preamble (32 ones) is not prepended to the MII management frame.
6–1 MII_SPEED	MII Speed Controls the frequency of the MII management interface clock (MDC) relative to the internal module clock. A value of 0 in this field turns off MDC and leaves it in low voltage state. Any non-zero value results in the MDC frequency of: 1/((MII_SPEED + 1) x 2) of the internal module clock frequency
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.5.8 MIB Control Register (ENETx_MIBC)

MIBC is a read/write register controlling and observing the state of the MIB block. Access this register to disable the MIB block operation or clear the MIB counters. The MIB_DIS field resets to 1.

Address: Base address + 64h offset



ENETx_MIBC field descriptions

Field	Description
31 MIB_DIS	Disable MIB Logic If this control field is set, 0 MIB logic is enabled. 1 MIB logic is disabled. The MIB logic halts and does not update any MIB counters.
30 MIB_IDLE	MIB Idle

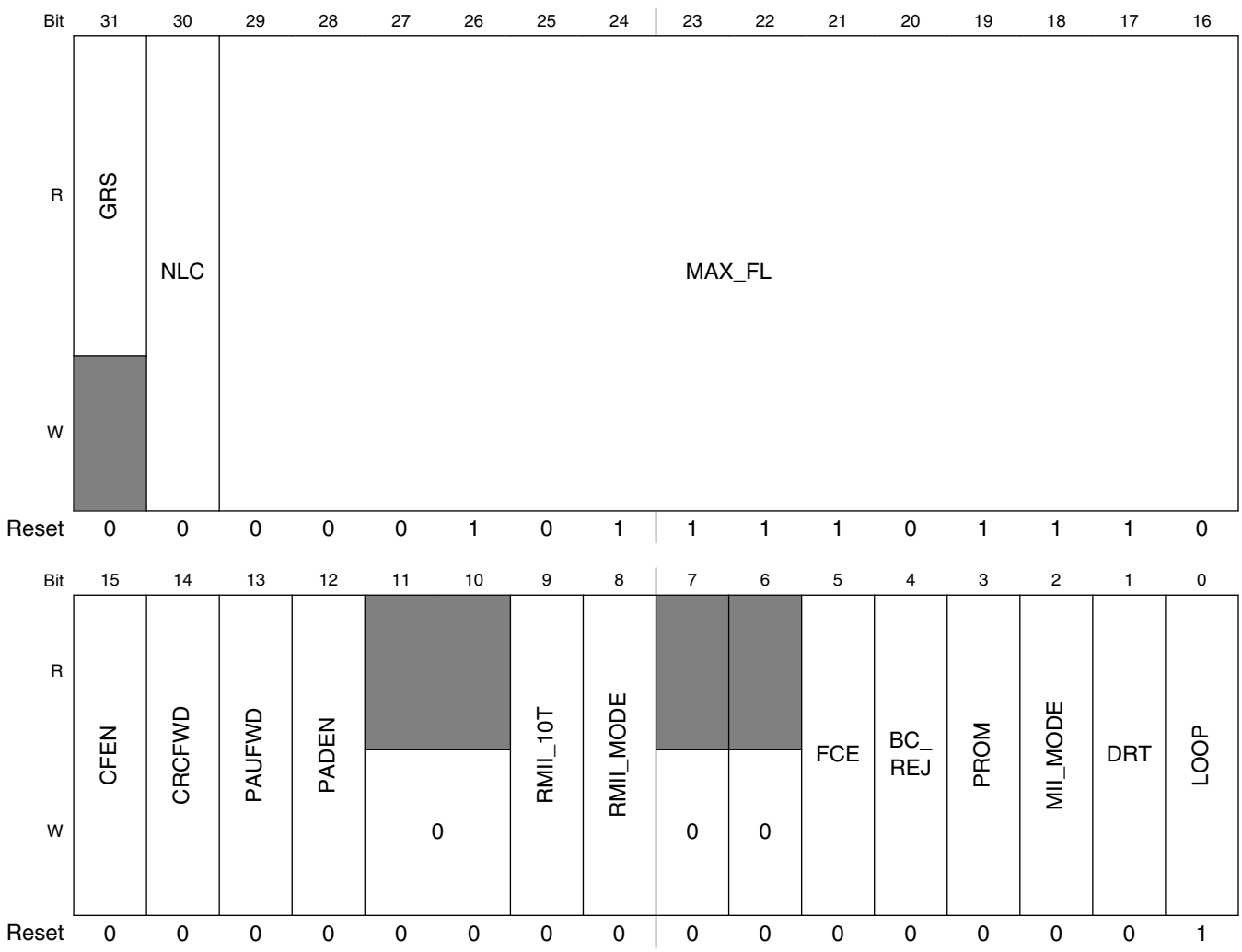
Table continues on the next page...

ENETx_MIBC field descriptions (continued)

Field	Description
	0 The MIB block is updating MIB counters. 1 The MIB block is not currently updating any MIB counters.
29 MIB_CLEAR	MIB Clear **NOTE:** This field is not self-clearing. To clear the MIB counters set and then clear this field. 0 See note above. 1 All statistics counters are reset to 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.5.9 Receive Control Register (ENETx_RCR)

Address: Base address + 84h offset



ENETx_RCR field descriptions

Field	Description
31 GRS	Graceful Receive Stopped Read-only status indicating that the MAC receive datapath is stopped.
30 NLC	Payload Length Check Disable Enables/disables a payload length check. 0 The payload length check is disabled. 1 The core checks the frame's payload length with the frame length/type field. Errors are indicated in the EIR[PLC] field.
29–16 MAX_FL	Maximum Frame Length Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL cause the BABT interrupt to occur. Receive frames longer than MAX_FL cause the BABR interrupt to occur and set the LG field in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
15 CFEN	MAC Control Frame Enable Enables/disables the MAC control frame. 0 MAC control frames with any opcode other than 0x0001 (pause frame) are accepted and forwarded to the client interface. 1 MAC control frames with any opcode other than 0x0001 (pause frame) are silently discarded.
14 CRCFWD	Terminate/Forward Received CRC Specifies whether the CRC field of received frames is transmitted or stripped. **NOTE:** If padding function is enabled (PADEN = 1), CRCFWD is ignored and the CRC field is checked and always terminated and removed. 0 The CRC field of received frames is transmitted to the user application. 1 The CRC field is stripped from the frame.
13 PAUFWD	Terminate/Forward Pause Frames Specifies whether pause frames are terminated or forwarded. 0 Pause frames are terminated and discarded in the MAC. 1 Pause frames are forwarded to the user application.
12 PADEN	Enable Frame Padding Remove On Receive Specifies whether the MAC removes padding from received frames. 0 No padding is removed on receive by the MAC. 1 Padding is removed from received frames.
11–10 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
9 RMII_10T	Enables 10-Mbit/s mode of the RMII . 0 100-Mbit/s operation. 1 10-Mbit/s operation.

Table continues on the next page...

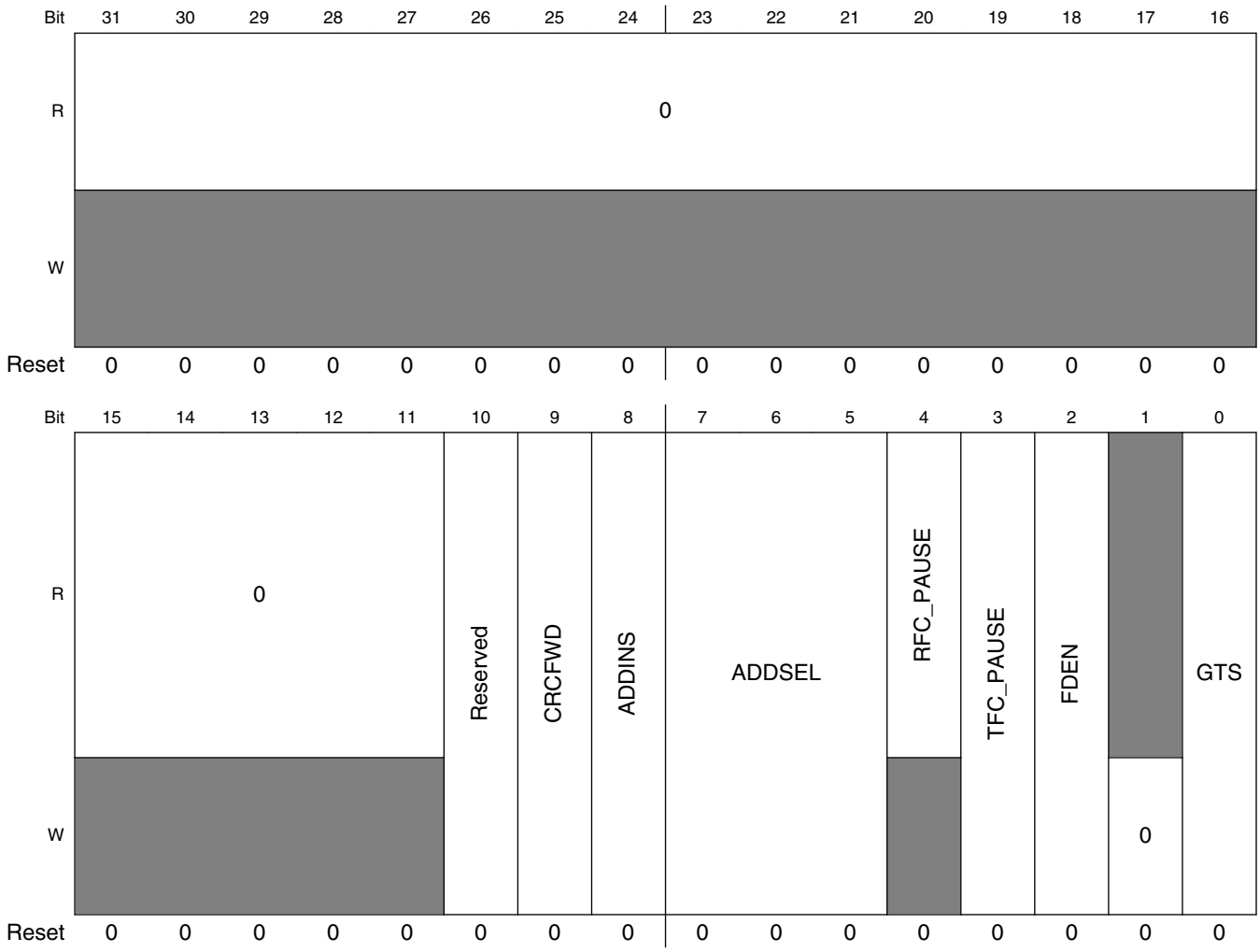
ENETx_RCR field descriptions (continued)

Field	Description
8 RMII_MODE	RMII Mode Enable Specifies whether the MAC is configured for MII mode or RMII operation . 0 MAC configured for MII mode. 1 MAC configured for RMII operation.
7 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
6 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
5 FCE	Flow Control Enable If set, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
4 BC_REJ	Broadcast Frame Reject If set, frames with destination address (DA) equal to 0xFFFF_FFFF_FFFF are rejected unless the PROM field is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the MISS (M) is set in the receive buffer descriptor.
3 PROM	Promiscuous Mode All frames are accepted regardless of address matching. 0 Disabled. 1 Enabled.
2 MII_MODE	Media Independent Interface Mode This field must always be set. 0 Reserved. 1 MII or RMII mode, as indicated by the RMII_MODE field.
1 DRT	Disable Receive On Transmit 0 Receive path operates independently of transmit (i.e., full-duplex mode). Can also be used to monitor transmit activity in half-duplex mode. 1 Disable reception of frames while transmitting. (Normally used for half-duplex mode.)
0 LOOP	Internal Loopback This is an MII internal loopback, therefore MII_MODE must be written to 1 and RMII_MODE must be written to 0. 0 Loopback disabled. 1 Transmitted frames are looped back internal to the device and transmit MII output signals are not asserted. DRT must be cleared.

22.5.10 Transmit Control Register (ENETx_TCR)

TCR is read/write and configures the transmit block. This register is cleared at system reset. FDEN can only be modified when ECR[ETHEREN] is cleared.

Address: Base address + C4h offset



ENETx_TCR field descriptions

Field	Description
31–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	This field is reserved. This field is read/write and must be set to 0.
9 CRCFWD	Forward Frame From Application With CRC 0 TxBD[TC] controls whether the frame has a CRC from the application. 1 The transmitter does not append any CRC to transmitted frames, as it is expecting a frame with CRC from the application.

Table continues on the next page...

ENETx_TCR field descriptions (continued)

Field	Description
8 ADDINS	Set MAC Address On Transmit 0 The source MAC address is not modified by the MAC. 1 The MAC overwrites the source MAC address with the programmed MAC address according to ADDSEL.
7–5 ADDSEL	Source MAC Address Select On Transmit If ADDINS is set, indicates the MAC address that overwrites the source MAC address. 000 Node MAC address programmed on PADDR1/2 registers. 100 Reserved. 101 Reserved. 110 Reserved.
4 RFC_PAUSE	Receive Frame Control Pause This status field is set when a full-duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This field automatically clears when the pause duration is complete.
3 TFC_PAUSE	Transmit Frame Control Pause Pauses frame transmission. When this field is set, EIR[GRA] is set. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC control PAUSE frame. 0 No PAUSE frame transmitted. 1 The MAC stops transmission of data frames after the current transmission is complete.
2 FDEN	Full-Duplex Enable If this field is set, frames transmit independent of carrier sense and collision inputs. Only modify this bit when ECR[ETHEREN] is cleared.
1 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
0 GTS	Graceful Transmit Stop When this field is set, MAC stops transmission after any frame currently transmitted is complete and EIR[GRA] is set. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHEREN] following the GRA interrupt.

22.5.11 Physical Address Lower Register (ENETx_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the six-byte source address field when transmitting PAUSE frames.

Address: Base address + E4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_PALR field descriptions

Field	Description
PADDR1	Pause Address Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

22.5.12 Physical Address Upper Register (ENETx_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the six-byte source address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) for transmission of PAUSE frames.

Address: Base address + E8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PADDR2																TYPE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

ENETx_PAUR field descriptions

Field	Description
31–16 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
TYPE	Type Field In PAUSE Frames These fields have a constant value of 0x8808.

22.5.13 Opcode/Pause Duration Register (ENETx_OPD)

OPD is read/write accessible. This register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. The lower 16 bits of this register are not reset and you must initialize it.

Address: Base address + ECh offset

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

ENETx_OPD field descriptions

Field	Description
31–16 OPCODE	Opcode Field In PAUSE Frames These fields have a constant value of 0x0001.
PAUSE_DUR	Pause Duration Pause duration field used in PAUSE frames.

22.5.14 Transmit Interrupt Coalescing Register (ENETx_TXIC)

See Interrupt coalescence for more information.

Address: Base address + F0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICEN	ICCS	Reserved		ICFT								Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_TXIC field descriptions

Field	Description
31 ICEN	Interrupt Coalescing Enable

Table continues on the next page...

ENETx_TXIC field descriptions (continued)

Field	Description
	0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.
30 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
29–28 Reserved	This field must be set to 0. This field is reserved.
27–20 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be transmitted for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
19–16 Reserved	This field must be set to 0. This field is reserved.
ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after transmitting a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame transmission defined by ICFT and starts again upon transmission of the next first frame. Must be greater than zero to avoid unpredictable behavior.

22.5.15 Receive Interrupt Coalescing Register (ENETx_RXIC)

See Interrupt coalescence for more information.

Address: Base address + 100h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICEN	ICCS	Reserved						ICFT				Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_RXIC field descriptions

Field	Description
31 ICEN	Interrupt Coalescing Enable 0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.

Table continues on the next page...

ENETx_RXIC field descriptions (continued)

Field	Description
30 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
29–28 Reserved	This field must be set to 0. This field is reserved.
27–20 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be received for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
19–16 Reserved	This field must be set to 0. This field is reserved.
ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after receiving a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame reception defined by ICFT and starts again upon reception of the next first frame. Must be greater than zero to avoid unpredictable behavior.

22.5.16 Descriptor Individual Upper Address Register (ENETx_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: Base address + 118h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_IAUR field descriptions

Field	Description
IADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

22.5.17 Descriptor Individual Lower Address Register (ENETx_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: Base address + 11Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_IALR field descriptions

Field	Description
IADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

22.5.18 Descriptor Group Upper Address Register (ENETx_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: Base address + 120h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_GAUR field descriptions

Field	Description
GADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

22.5.19 Descriptor Group Lower Address Register (ENETx_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: Base address + 124h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_GALR field descriptions

Field	Description
GADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

22.5.20 Transmit FIFO Watermark Register (ENETx_TFWR)

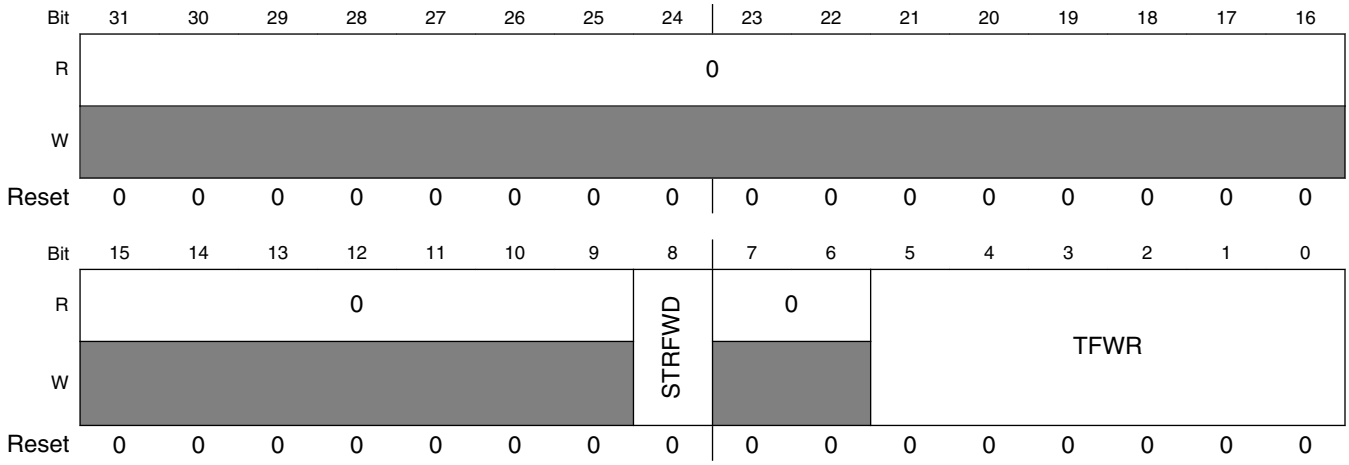
If TFWR[STRFWD] is cleared, TFWR[TFWR] controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement, for example, worst-case bus access latency by the transmit data uDMA channel.

When the FIFO level reaches the value the TFWR field and when the STR_FWD is set to ‘0’, the MAC transmit control logic starts frame transmission even before the end-of-frame is available in the FIFO (cut-through operation).

If a complete frame has a size smaller than the threshold programmed with TFWR, the MAC also transmits the Frame to the line.

To enable store and forward on the Transmit path, set STR_FWD to ‘1’. In this case, the MAC starts to transmit data only when a complete frame is stored in the Transmit FIFO.

Address: Base address + 144h offset



ENETx_TFWR field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 STRFWD	Store And Forward Enable 0 Reset. The transmission start threshold is programmed in TFWR[TFWR]. 1 Enabled.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TFWR	Transmit FIFO Write If TFWR[STRFWD] is cleared, this field indicates the number of bytes, in steps of 64 bytes, written to the transmit FIFO before transmission of a frame begins. **NOTE:** If a frame with less than the threshold is written, it is still sent independently of this threshold setting. The threshold is relevant only if the frame is larger than the threshold given. 000000 64 bytes written. 000001 64 bytes written. 000010 128 bytes written. 000011 192 bytes written. ... ... 011111 1984 bytes written.

22.5.21 Receive Descriptor Ring Start Register (ENETx_RDSR)

RDSR points to the beginning of the circular receive buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 2–0 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: Base address + 180h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_RDSR field descriptions

Field	Description
31–3 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue.
2 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.5.22 Transmit Buffer Descriptor Ring Start Register (ENETx_TDSR)

TDSR provides a pointer to the beginning of the circular transmit buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 2–0 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: Base address + 184h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_TDSR field descriptions

Field	Description
31–3 X_DES_START	Pointer to the beginning of the transmit buffer descriptor queue.
2 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.5.23 Maximum Receive Buffer Size Register (ENETx_MRBR)

The MRBR is a user-programmable register that dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To allow one maximum size frame per buffer, MRBR must be set to RCR[MAX_FL] or larger.
- To properly align the buffer, MRBR must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_SIZE to zero. The lower four bits of this register are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

Address: Base address + 188h offset

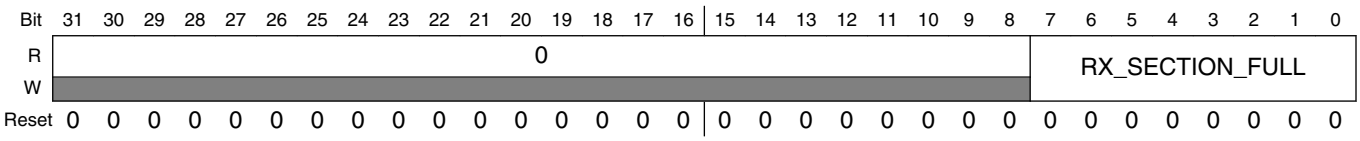
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																R_BUF_SIZE												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_MRBR field descriptions

Field	Description
31–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–4 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

22.5.24 Receive FIFO Section Full Threshold (ENETx_RSFL)

Address: Base address + 190h offset

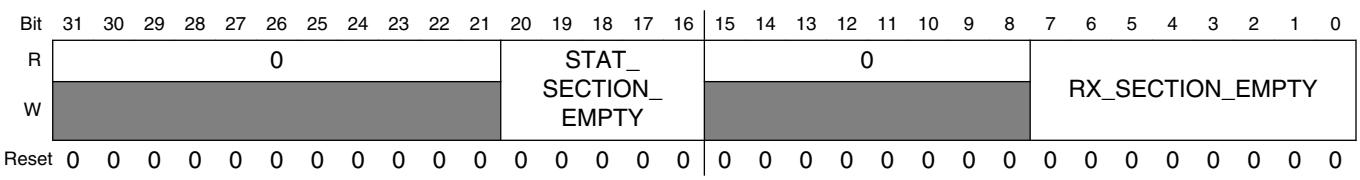


ENETx_RSFL field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RX_SECTION_ FULL	Value Of Receive FIFO Section Full Threshold Value, in 64-bit words, of the receive FIFO section full threshold. Clear this field to enable store and forward on the RX FIFO. When programming a value greater than 0 (cut-through operation), it must be greater than RAEM[RX_ALMOST_EMPTY]. When the FIFO level reaches the value in this field, data is available in the Receive FIFO (cut-through operation).

22.5.25 Receive FIFO Section Empty Threshold (ENETx_RSEM)

Address: Base address + 194h offset



ENETx_RSEM field descriptions

Field	Description
31–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 STAT_ SECTION_ EMPTY	RX Status FIFO Section Empty Threshold Defines number of frames in the receive FIFO, independent of its size, that can be accepted. If the limit is reached, reception will continue normally, however a pause frame will be triggered to indicate a possible congestion to the remote device to avoid FIFO overflow. A value of 0 disables automatic pause frame generation
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RX_SECTION_ EMPTY	Value Of The Receive FIFO Section Empty Threshold Value, in 64-bit words, of the receive FIFO section empty threshold. When the FIFO has reached this level, a pause frame will be issued.

Table continues on the next page...

ENETx_RSEM field descriptions (continued)

Field	Description
	A value of 0 disables automatic pause frame generation. When the FIFO level goes below the value programmed in this field, an XON pause frame is issued to indicate the FIFO congestion is cleared to the remote Ethernet client. **NOTE:** The section-empty threshold indications from both FIFOs are OR'ed to cause XOFF pause frame generation.

22.5.26 Receive FIFO Almost Empty Threshold (ENETx_RAEM)

Address: Base address + 198h offset

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

ENETx_RAEM field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RX_ALMOST_EMPTY	Value Of The Receive FIFO Almost Empty Threshold Value, in 64-bit words, of the receive FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field and the end-of-frame has not been received for the frame yet, the core receive read control stops FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. A minimum value of 4 should be set.

22.5.27 Receive FIFO Almost Full Threshold (ENETx_RAFL)

Address: Base address + 19Ch offset

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

ENETx_RAFL field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RX_ALMOST_FULL	Value Of The Receive FIFO Almost Full Threshold

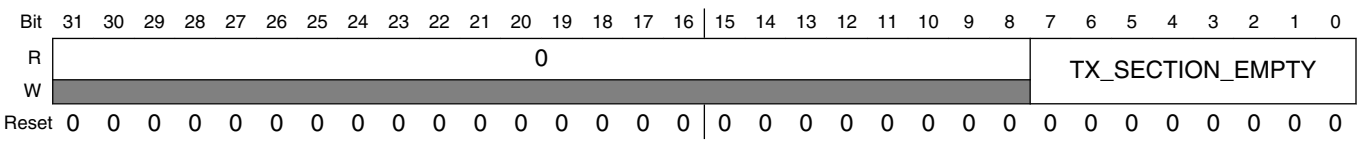
Table continues on the next page...

ENETx_RAFL field descriptions (continued)

Field	Description
	Value, in 64-bit words, of the receive FIFO almost full threshold. When the FIFO level comes close to the maximum, so that there is no more space for at least RX_ALMOST_FULL number of words, the MAC stops writing data in the FIFO and truncates the received frame to avoid FIFO overflow. The corresponding error status will be set when the frame is delivered to the application. A minimum value of 4 should be set.

22.5.28 Transmit FIFO Section Empty Threshold (ENETx_TSEM)

Address: Base address + 1A0h offset

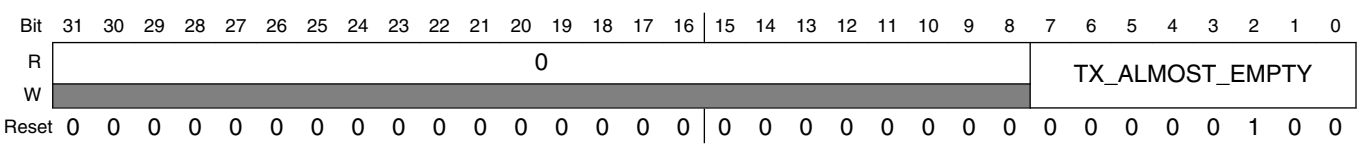


ENETx_TSEM field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TX_SECTION_EMPTY	Value Of The Transmit FIFO Section Empty Threshold Value, in 64-bit words, of the transmit FIFO section empty threshold. See Transmit FIFO for more information.

22.5.29 Transmit FIFO Almost Empty Threshold (ENETx_TAEM)

Address: Base address + 1A4h offset



ENETx_TAEM field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TX_ALMOST_EMPTY	Value of Transmit FIFO Almost Empty Threshold Value, in 64-bit words, of the transmit FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field, and no end-of-frame is available for the frame, the MAC transmit logic, to avoid FIFO underflow, stops reading the FIFO and transmits a frame with an MII error indication. See Transmit FIFO for more information. A minimum value of 4 should be set.

22.5.30 Transmit FIFO Almost Full Threshold (ENETx_TAFL)

Address: Base address + 1A8h offset

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

ENETx_TAFL field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TX_ALMOST_FULL	Value Of The Transmit FIFO Almost Full Threshold Value, in 64-bit words, of the transmit FIFO almost full threshold. A minimum value of six is required . A recommended value of at least 8 should be set allowing a latency of two clock cycles to the application. If more latency is required the value can be increased as necessary (latency = TAFL - 5). When the FIFO level comes close to the maximum, so that there is no more space for at least TX_ALMOST_FULL number of words, the pin ff_tx_rdy is deasserted. If the application does not react on this signal, the FIFO write control logic, to avoid FIFO overflow, truncates the current frame and sets the error status. As a result, the frame will be transmitted with an GMII/MII error indication. See Transmit FIFO for more information. **NOTE:** A FIFO overflow is a fatal error and requires a global reset on the transmit datapath or at least deassertion of ETHEREN.

22.5.31 Transmit Inter-Packet Gap (ENETx_TIPG)

Address: Base address + 1ACh offset

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

ENETx_TIPG field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
IPG	Transmit Inter-Packet Gap Indicates the IPG, in bytes, between transmitted frames. Valid values range from 8 to 26. If the written value is less than 8 or greater than 26, the internal (effective) IPG is 12. **NOTE:** The IPG value read will be the value that was written, even if it is out of range.

22.5.32 Frame Truncation Length (ENETx_FTRL)

Address: Base address + 1B0h offset

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

ENETx_FTRL field descriptions

Field	Description
31–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TRUNC_FL	Frame Truncation Length Indicates the value a receive frame is truncated, if it is greater than this value. Must be greater than or equal to RCR[MAX_FL]. **NOTE:** Truncation happens at TRUNC_FL. However, when truncation occurs, the application (FIFO) may receive less data, guaranteeing that it never receives more than the set limit.

22.5.33 Transmit Accelerator Function Configuration (ENETx_TACC)

TACC controls accelerator actions when sending frames. The register can be changed before or after each frame, but it must remain unmodified during frame writes into the transmit FIFO.

The TFWR[STRFWD] field must be set to use the checksum feature.

Address: Base address + 1C0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R												PROCHK	IPCHK			SHIFT16
W	0													0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_TACC field descriptions

Field	Description
31–5 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
4 PROCHK	Enables insertion of protocol checksum. 0 Checksum not inserted. 1 If an IP frame with a known protocol is transmitted, the checksum is inserted automatically into the frame. The checksum field must be cleared. The other frames are not modified.
3 IPCHK	Enables insertion of IP header checksum. 0 Checksum is not inserted. 1 If an IP frame is transmitted, the checksum is inserted automatically. The IP header checksum field must be cleared. If a non-IP frame is transmitted the frame is not modified.
2–1 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
0 SHIFT16	TX FIFO Shift-16 0 Disabled. 1 Indicates to the transmit data FIFO that the written frames contain two additional octets before the frame data. This means the actual frame begins at bit 16 of the first word written into the FIFO. This function allows putting the frame payload on a 32-bit boundary in memory, as the 14-byte Ethernet header is extended to a 16-byte header.

22.5.34 Receive Accelerator Function Configuration (ENETx_RACC)

Address: Base address + 1C4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									SHIFT16	LINEDIS				PRODIS	IPDIS	PADREM
W	0										0					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_RACC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

Table continues on the next page...

ENETx_RACC field descriptions (continued)

Field	Description
7 SHIFT16	RX FIFO Shift-16 When this field is set, the actual frame data starts at bit 16 of the first word read from the RX FIFO aligning the Ethernet payload on a 32-bit boundary. **NOTE:** This function only affects the FIFO storage and has no influence on the statistics, which use the actual length of the frame received. 0 Disabled. 1 Instructs the MAC to write two additional bytes in front of each frame received into the RX FIFO.
6 LINEDIS	Enable Discard Of Frames With MAC Layer Errors 0 Frames with errors are not discarded. 1 Any frame received with a CRC, length, or PHY error is automatically discarded and not forwarded to the user application interface.
5–3 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
2 PRODIS	Enable Discard Of Frames With Wrong Protocol Checksum 0 Frames with wrong checksum are not discarded. 1 If a TCP/IP, UDP/IP, or ICMP/IP frame is received that has a wrong TCP, UDP, or ICMP checksum, the frame is discarded. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
1 IPDIS	Enable Discard Of Frames With Wrong IPv4 Header Checksum 0 Frames with wrong IPv4 header checksum are not discarded. 1 If an IPv4 frame is received with a mismatching header checksum, the frame is discarded. IPv6 has no header checksum and is not affected by this setting. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
0 PADREM	Enable Padding Removal For Short IP Frames 0 Padding not removed. 1 Any bytes following the IP payload section of the frame are removed from the frame.

22.5.35 Reserved Statistic Register (ENETx_RMON_T_DROP)

Address: Base address + 200h offset

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

ENETx_RMON_T_DROP field descriptions

Field	Description
Reserved	This read-only field always has the value 0. This field is reserved.

22.5.36 Tx Packet Count Statistic Register (ENETx_RMON_T_PACKETS)

Address: Base address + 204h offset

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

ENETx_RMON_T_PACKETS field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Packet count Transmit packet count

22.5.37 Tx Broadcast Packets Statistic Register (ENETx_RMON_T_BC_PKT)

RMON Tx Broadcast Packets

Address: Base address + 208h offset

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

ENETx_RMON_T_BC_PKT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Broadcast packets

22.5.38 Tx Multicast Packets Statistic Register (ENETx_RMON_T_MC_PKT)

Address: Base address + 20Ch offset

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

ENETx_RMON_T_MC_PKT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Multicast packets

22.5.39 Tx Packets with CRC/Align Error Statistic Register (ENETx_RMON_T_CRC_ALIGN)

Address: Base address + 210h offset

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

ENETx_RMON_T_CRC_ALIGN field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Packets with CRC/align error

22.5.40 Tx Packets Less Than Bytes and Good CRC Statistic Register (ENETx_RMON_T_UNDERSIZE)

Address: Base address + 214h offset

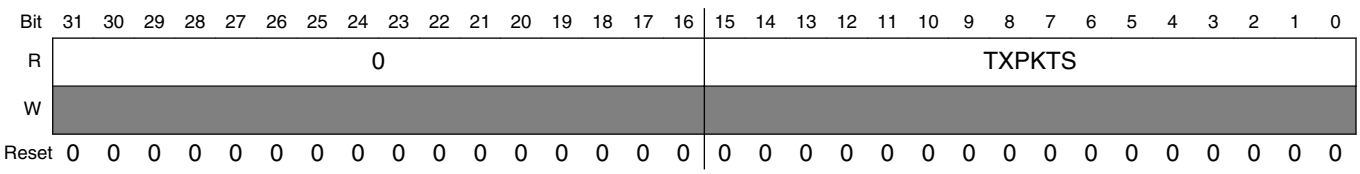
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_RMON_T_UNDERSIZE field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of transmit packets less than 64 bytes with good CRC

22.5.41 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENETx_RMON_T_OVERSIZE)

Address: Base address + 218h offset



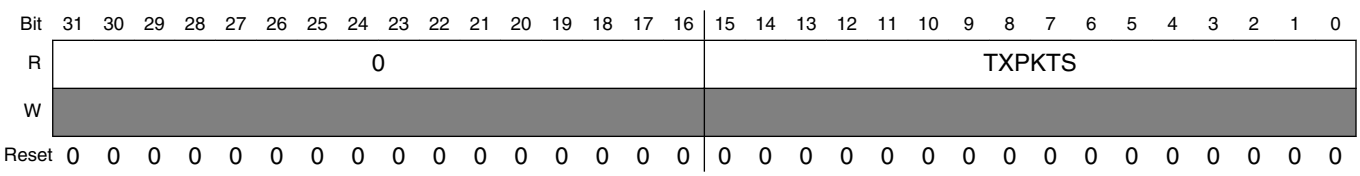
ENETx_RMON_T_OVERSIZE field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of transmit packets greater than MAX_FL bytes with good CRC

22.5.42 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENETx_RMON_T_FRAG)

.

Address: Base address + 21Ch offset



ENETx_RMON_T_FRAG field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of packets less than 64 bytes with bad CRC

22.5.43 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENETx_RMON_T_JAB)

Address: Base address + 220h offset

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

ENETx_RMON_T_JAB field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of transmit packets greater than MAX_FL bytes and bad CRC

22.5.44 Tx Collision Count Statistic Register (ENETx_RMON_T_COL)

Address: Base address + 224h offset

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

ENETx_RMON_T_COL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of transmit collisions

22.5.45 Tx 64-Byte Packets Statistic Register (ENETx_RMON_T_P64)

.

Address: Base address + 228h offset

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

ENETx_RMON_T_P64 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of 64-byte transmit packets

22.5.46 Tx 65- to 127-byte Packets Statistic Register (ENETx_RMON_T_P65TO127)

Address: Base address + 22Ch offset

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

ENETx_RMON_T_P65TO127 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of 65- to 127-byte transmit packets

22.5.47 Tx 128- to 255-byte Packets Statistic Register (ENETx_RMON_T_P128TO255)

Address: Base address + 230h offset

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

ENETx_RMON_T_P128TO255 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of 128- to 255-byte transmit packets

22.5.48 Tx 256- to 511-byte Packets Statistic Register (ENETx_RMON_T_P256TO511)

Address: Base address + 234h offset

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

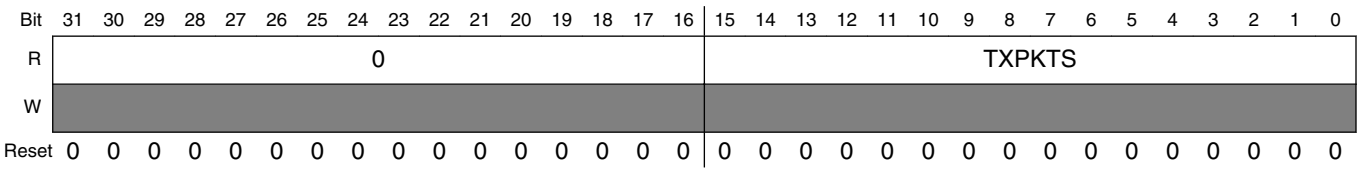
ENETx_RMON_T_P256TO511 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of 256- to 511-byte transmit packets

22.5.49 Tx 512- to 1023-byte Packets Statistic Register (ENETx_RMON_T_P512TO1023)

.

Address: Base address + 238h offset

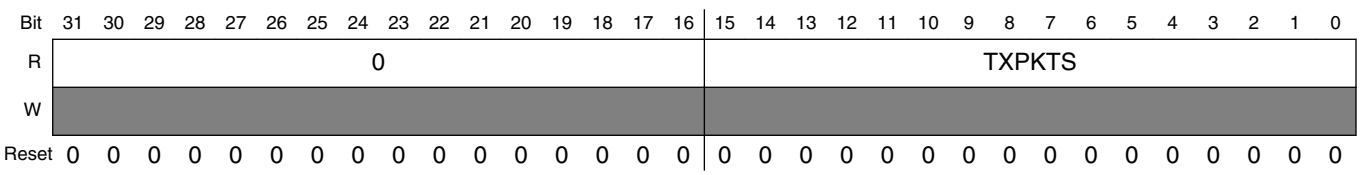


ENETx_RMON_T_P512TO1023 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of 512- to 1023-byte transmit packets

22.5.50 Tx 1024- to 2047-byte Packets Statistic Register (ENETx_RMON_T_P1024TO2047)

Address: Base address + 23Ch offset

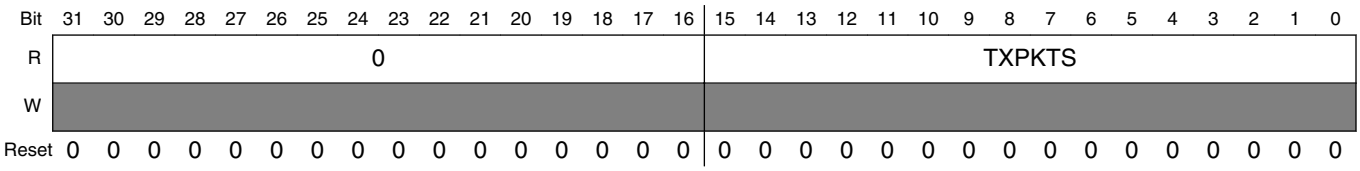


ENETx_RMON_T_P1024TO2047 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of 1024- to 2047-byte transmit packets

22.5.51 Tx Packets Greater Than 2048 Bytes Statistic Register (ENETx_RMON_T_P_GTE2048)

Address: Base address + 240h offset

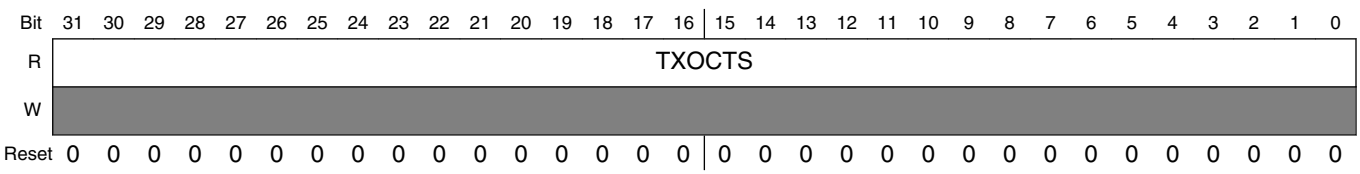


ENETx_RMON_T_P_GTE2048 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXPKTS	Number of transmit packets greater than 2048 bytes

22.5.52 Tx Octets Statistic Register (ENETx_RMON_T_OCTETS)

Address: Base address + 244h offset

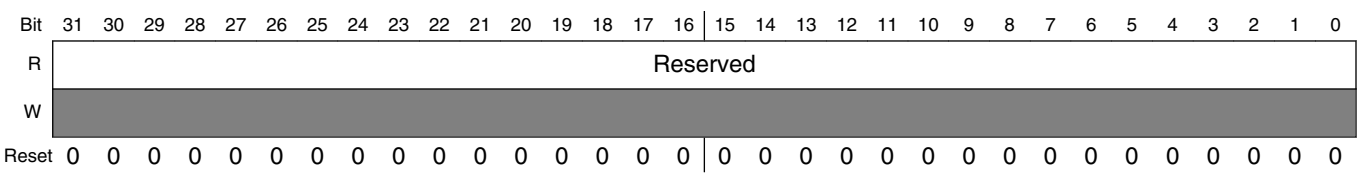


ENETx_RMON_T_OCTETS field descriptions

Field	Description
TXOCTS	Number of transmit octets

22.5.53 Reserved Statistic Register (ENETx_IEEE_T_DROP)

Address: Base address + 248h offset

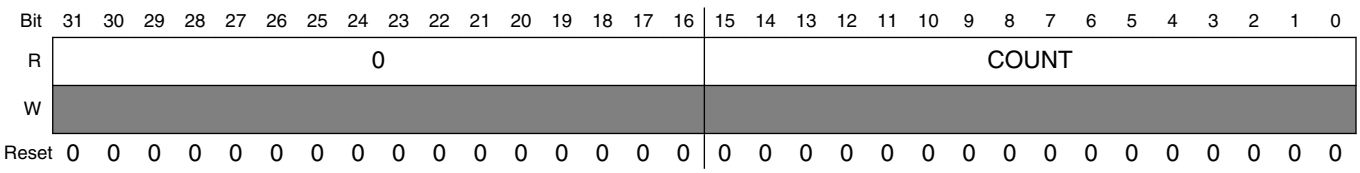


ENETx_IEEE_T_DROP field descriptions

Field	Description
Reserved	This read-only field always has the value 0. This field is reserved.

22.5.54 Frames Transmitted OK Statistic Register (ENETx_IEEE_T_FRAME_OK)

Address: Base address + 24Ch offset

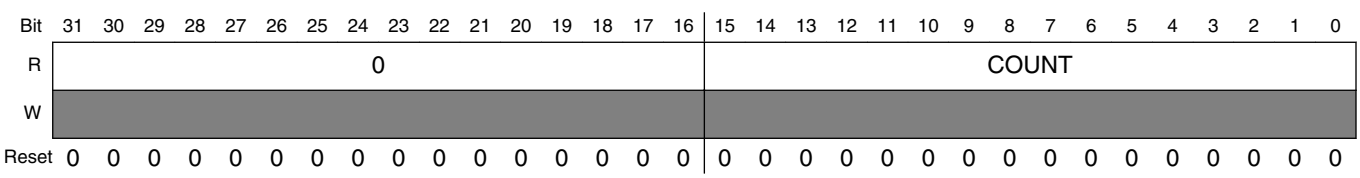


ENETx_IEEE_T_FRAME_OK field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted OK **NOTE:** Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

22.5.55 Frames Transmitted with Single Collision Statistic Register (ENETx_IEEE_T_1COL)

Address: Base address + 250h offset

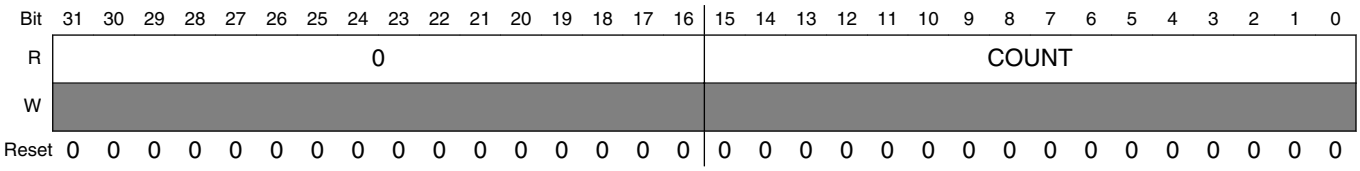


ENETx_IEEE_T_1COL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted with one collision

22.5.56 Frames Transmitted with Multiple Collisions Statistic Register (ENETx_IEEE_T_MCOL)

Address: Base address + 254h offset

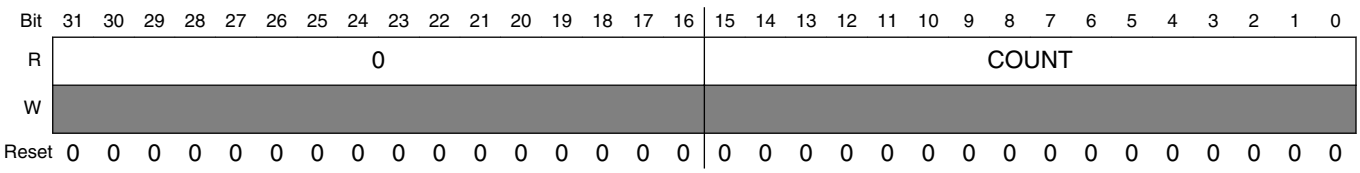


ENETx_IEEE_T_MCOL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted with multiple collisions

22.5.57 Frames Transmitted after Deferral Delay Statistic Register (ENETx_IEEE_T_DEF)

Address: Base address + 258h offset

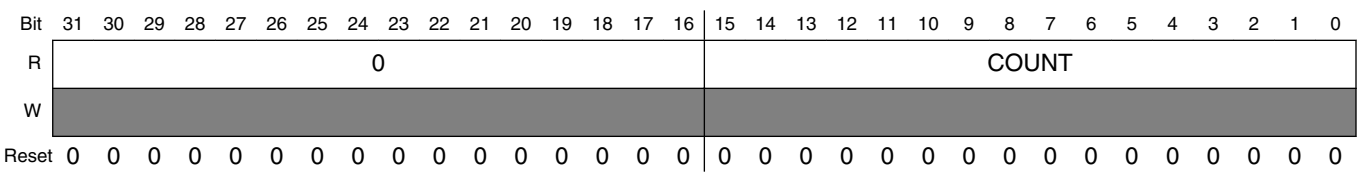


ENETx_IEEE_T_DEF field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted with deferral delay

22.5.58 Frames Transmitted with Late Collision Statistic Register (ENETx_IEEE_T_LCOL)

Address: Base address + 25Ch offset

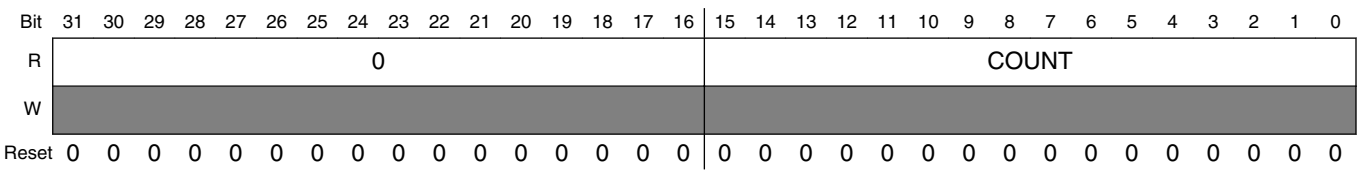


ENETx_IEEE_T_LCOL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted with late collision

22.5.59 Frames Transmitted with Excessive Collisions Statistic Register (ENETx_IEEE_T_EXCOL)

Address: Base address + 260h offset

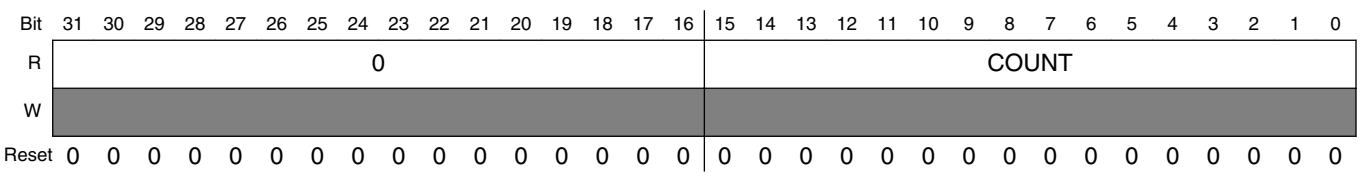


ENETx_IEEE_T_EXCOL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted with excessive collisions

22.5.60 Frames Transmitted with Tx FIFO Underrun Statistic Register (ENETx_IEEE_T_MACERR)

Address: Base address + 264h offset



ENETx_IEEE_T_MACERR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted with transmit FIFO underrun

22.5.61 Frames Transmitted with Carrier Sense Error Statistic Register (ENETx_IEEE_T_CSERR)

Address: Base address + 268h offset

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

ENETx_IEEE_T_CSERR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames transmitted with carrier sense error

22.5.62 Reserved Statistic Register (ENETx_IEEE_T_SQE)

Address: Base address + 26Ch offset

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

ENETx_IEEE_T_SQE field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	This read-only field is reserved and always has the value 0. **NOTE:** Counter not implemented as no SQE information is available.

22.5.63 Flow Control Pause Frames Transmitted Statistic Register (ENETx_IEEE_T_FDXFC)

Address: Base address + 270h offset

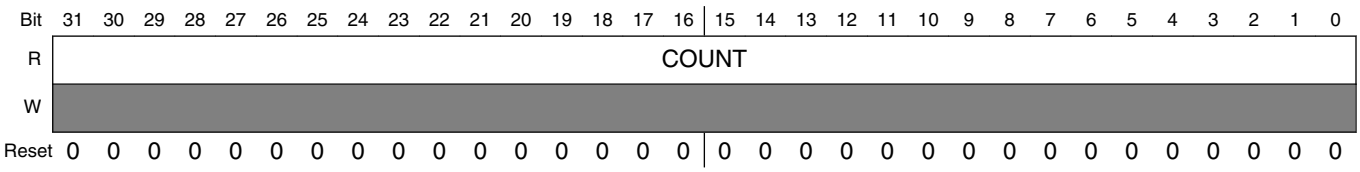
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_IEEE_T_FDXFC field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of flow-control pause frames transmitted

22.5.64 Octet Count for Frames Transmitted w/o Error Statistic Register (ENETx_IEEE_T_OCTETS_OK)

Address: Base address + 274h offset

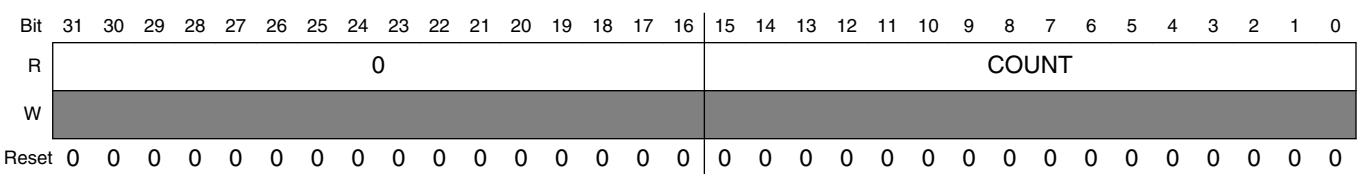


ENETx_IEEE_T_OCTETS_OK field descriptions

Field	Description
COUNT	Octet count for frames transmitted without error **NOTE** Counts total octets (includes header and FCS fields).

22.5.65 Rx Packet Count Statistic Register (ENETx_RMON_R_PACKETS)

Address: Base address + 284h offset

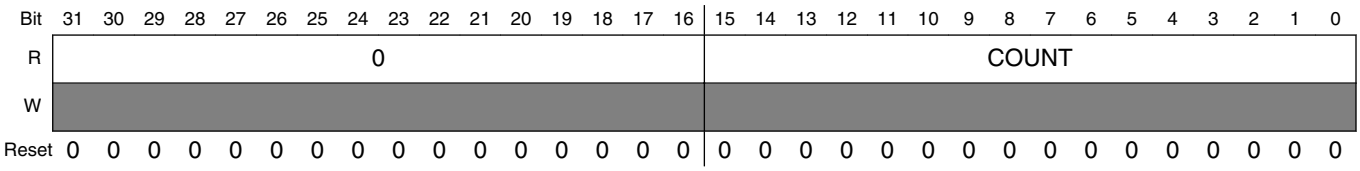


ENETx_RMON_R_PACKETS field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of packets received

22.5.66 Rx Broadcast Packets Statistic Register (ENETx_RMON_R_BC_PKT)

Address: Base address + 288h offset

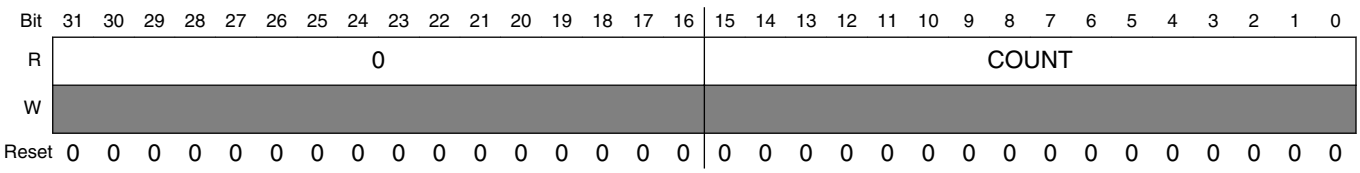


ENETx_RMON_R_BC_PKT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of receive broadcast packets

22.5.67 Rx Multicast Packets Statistic Register (ENETx_RMON_R_MC_PKT)

Address: Base address + 28Ch offset

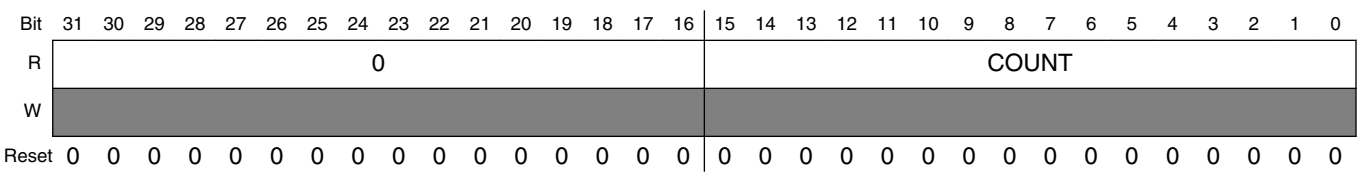


ENETx_RMON_R_MC_PKT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of receive multicast packets

22.5.68 Rx Packets with CRC/Align Error Statistic Register (ENETx_RMON_R_CRC_ALIGN)

Address: Base address + 290h offset



ENETx_RMON_R_CRC_ALIGN field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of receive packets with CRC or align error

22.5.69 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENETx_RMON_R_UNDERSIZE)

Address: Base address + 294h offset

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

ENETx_RMON_R_UNDERSIZE field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of receive packets with less than 64 bytes and good CRC

22.5.70 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENETx_RMON_R_OVERSIZE)

Address: Base address + 298h offset

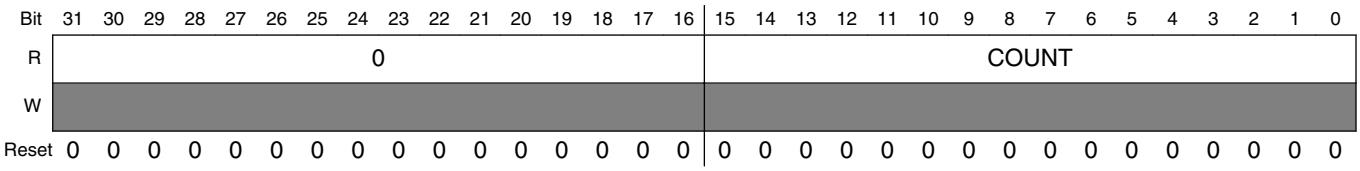
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_RMON_R_OVERSIZE field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of receive packets greater than MAX_FL and good CRC

22.5.71 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENETx_RMON_R_FRAG)

Address: Base address + 29Ch offset

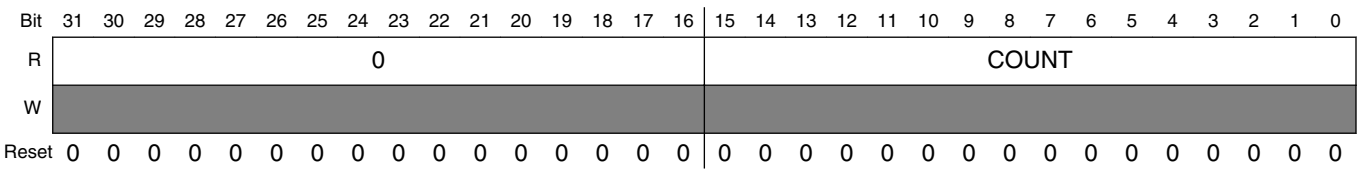


ENETx_RMON_R_FRAG field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of receive packets with less than 64 bytes and bad CRC

22.5.72 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENETx_RMON_R_JAB)

Address: Base address + 2A0h offset

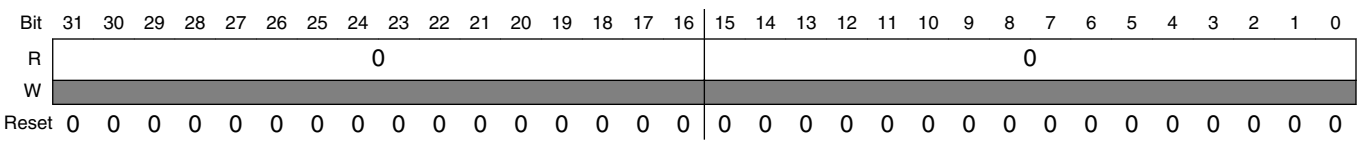


ENETx_RMON_R_JAB field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of receive packets greater than MAX_FL and bad CRC

22.5.73 Reserved Statistic Register (ENETx_RMON_R_RESVD_0)

Address: Base address + 2A4h offset

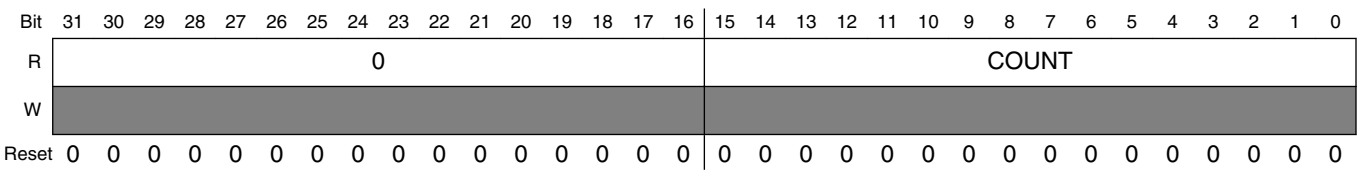


ENETx_RMON_R_RESVD_0 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.5.74 Rx 64-Byte Packets Statistic Register (ENETx_RMON_R_P64)

Address: Base address + 2A8h offset

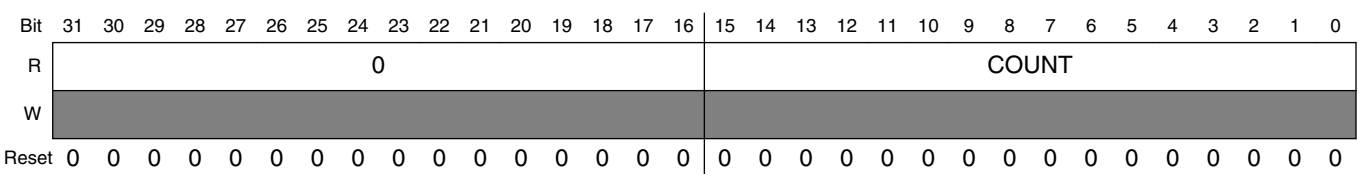


ENETx_RMON_R_P64 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of 64-byte receive packets

22.5.75 Rx 65- to 127-Byte Packets Statistic Register (ENETx_RMON_R_P65TO127)

Address: Base address + 2ACh offset



ENETx_RMON_R_P65TO127 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of 65- to 127-byte recieve packets

22.5.76 Rx 128- to 255-Byte Packets Statistic Register (ENETx_RMON_R_P128TO255)

Address: Base address + 2B0h offset

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

ENETx_RMON_R_P128TO255 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of 128- to 255-byte recieve packets

22.5.77 Rx 256- to 511-Byte Packets Statistic Register (ENETx_RMON_R_P256TO511)

Address: Base address + 2B4h offset

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

ENETx_RMON_R_P256TO511 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of 256- to 511-byte recieve packets

22.5.78 Rx 512- to 1023-Byte Packets Statistic Register (ENETx_RMON_R_P512TO1023)

Address: Base address + 2B8h offset

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

ENETx_RMON_R_P512TO1023 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of 512- to 1023-byte recieve packets

22.5.79 Rx 1024- to 2047-Byte Packets Statistic Register (ENETx_RMON_R_P1024TO2047)

Address: Base address + 2BCh offset

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

ENETx_RMON_R_P1024TO2047 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of 1024- to 2047-byte recieve packets

22.5.80 Rx Packets Greater than 2048 Bytes Statistic Register (ENETx_RMON_R_P_GTE2048)

Address: Base address + 2C0h offset

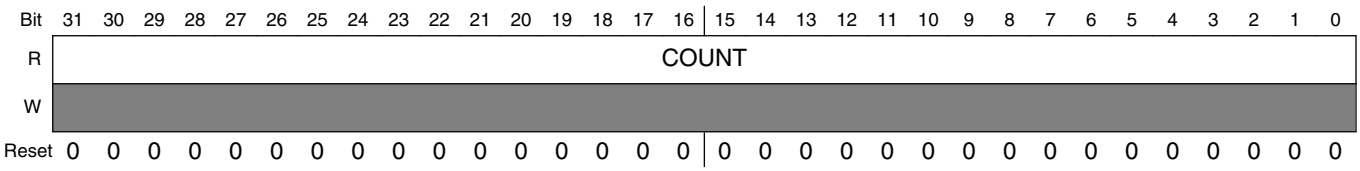
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_RMON_R_P_GTE2048 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of greater-than-2048-byte recieve packets

22.5.81 Rx Octets Statistic Register (ENETx_RMON_R_OCTETS)

Address: Base address + 2C4h offset



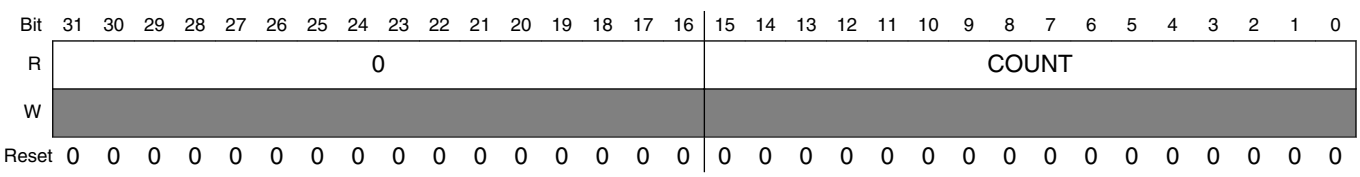
ENETx_RMON_R_OCTETS field descriptions

Field	Description
COUNT	Number of receive octets

22.5.82 Frames not Counted Correctly Statistic Register (ENETx_IEEE_R_DROP)

Counter increments if a frame with invalid or missing SFD character is detected and has been dropped. None of the other counters increments if this counter increments.

Address: Base address + 2C8h offset

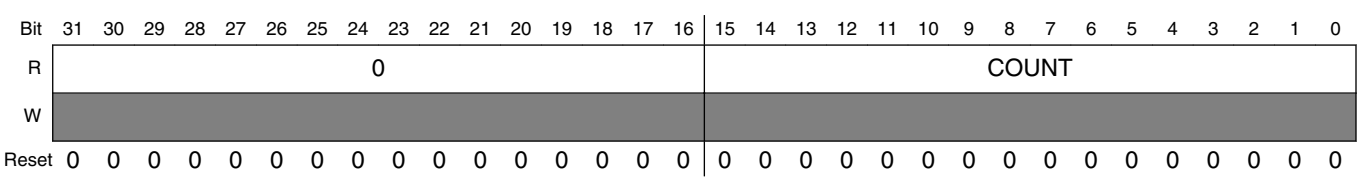


ENETx_IEEE_R_DROP field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Frame count

22.5.83 Frames Received OK Statistic Register (ENETx_IEEE_R_FRAME_OK)

Address: Base address + 2CCh offset



ENETx_IEEE_R_FRAME_OK field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames received OK

22.5.84 Frames Received with CRC Error Statistic Register (ENETx_IEEE_R_CRC)

Address: Base address + 2D0h offset

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

ENETx_IEEE_R_CRC field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames received with CRC error

22.5.85 Frames Received with Alignment Error Statistic Register (ENETx_IEEE_R_ALIGN)

Address: Base address + 2D4h offset

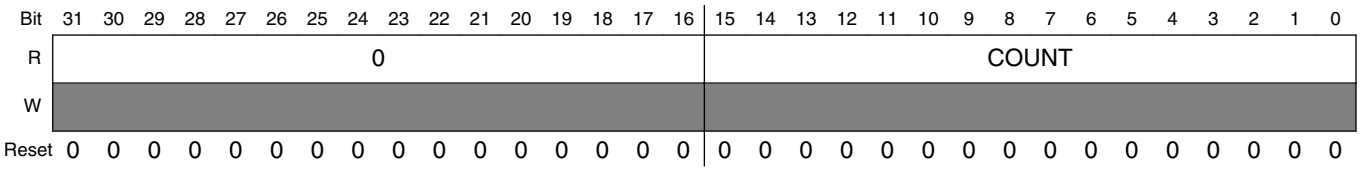
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_IEEE_R_ALIGN field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of frames received with alignment error

22.5.86 Receive FIFO Overflow Count Statistic Register (ENETx_IEEE_R_MACERR)

Address: Base address + 2D8h offset

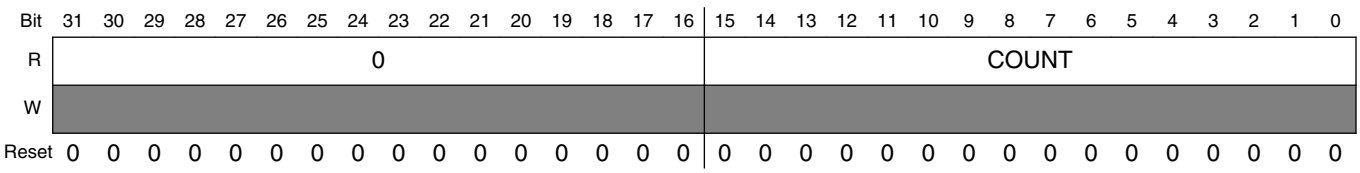


ENETx_IEEE_R_MACERR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Receive FIFO overflow count

22.5.87 Flow Control Pause Frames Received Statistic Register (ENETx_IEEE_R_FDXFC)

Address: Base address + 2DCh offset

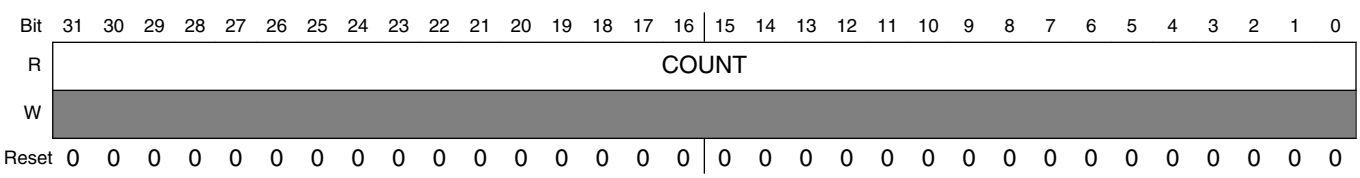


ENETx_IEEE_R_FDXFC field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Number of flow-control pause frames received

22.5.88 Octet Count for Frames Received without Error Statistic Register (ENETx_IEEE_R_OCTETS_OK)

Address: Base address + 2E0h offset



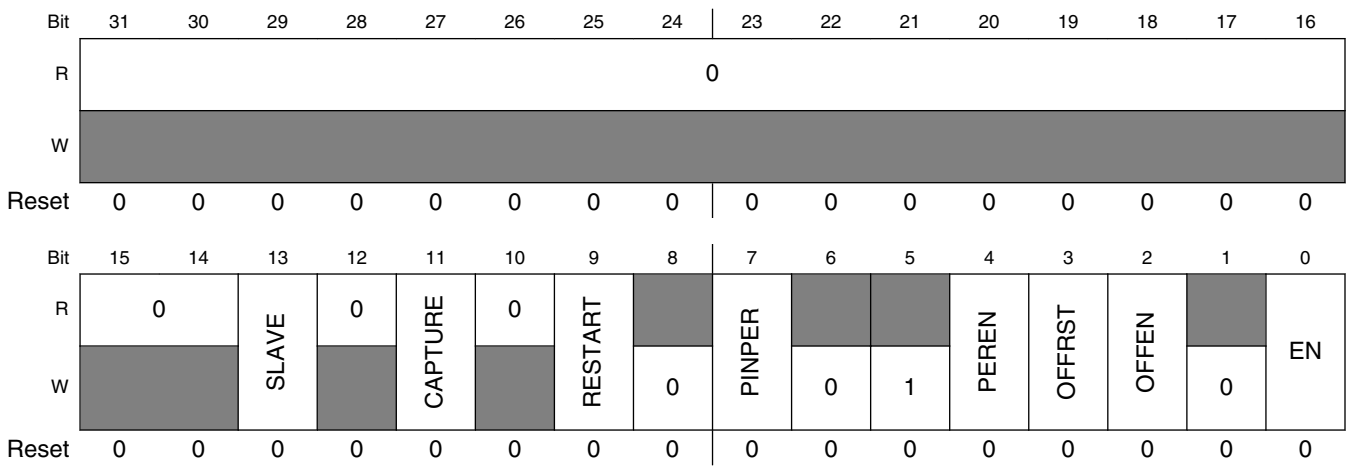
ENETx_IEEE_R_OCTETS_OK field descriptions

Field	Description
COUNT	Number of octets for frames received without error **NOTE:** Counts total octets (includes header and FCS fields). Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

22.5.89 Adjustable Timer Control Register (ENETx_ATCR)

ATCR command fields can trigger the corresponding events directly. It is not necessary to preserve any of the configuration fields when a command field is set in the register, that is, no read-modify-write is required.

Address: Base address + 400h offset



ENETx_ATCR field descriptions

Field	Description
31–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SLAVE	Enable Timer Slave Mode 0 The timer is active and all configuration fields in this register are relevant. 1 The internal timer is disabled and the externally provided timer value is used. All other fields, except CAPTURE, in this register have no effect. CAPTURE can still be used to capture the current timer value.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 CAPTURE	Capture Timer Value

Table continues on the next page...

ENETx_ATCR field descriptions (continued)

Field	Description
	When this field is set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. 0 No effect. 1 The current time is captured and can be read from the ATVR register.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 RESTART	Reset Timer Resets the timer to zero. This has no effect on the counter enable. If the counter is enabled when this field is set, the timer is reset to zero and starts counting from there. When set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes.
8 Reserved	This field is reserved.
7 PINPER	Enables event signal output assertion on period event. **NOTE:** Not all devices contain the event signal output. See the chip configuration details. 0 Disable. 1 Enable.
6 Reserved	This field is reserved.
5 Reserved	This field is reserved. **NOTE:** This field must be written always with one.
4 PEREN	Enable Periodical Event 0 Disable. 1 A period event interrupt can be generated (EIR[TS_TIMER]) and the event signal output is asserted when the timer wraps around according to the periodic setting ATPER. The timer period value must be set before setting this bit. **NOTE:** Not all devices contain the event signal output. See the chip configuration details.
3 OFFRST	Reset Timer On Offset Event 0 The timer is not affected and no action occurs, besides clearing OFFEN, when the offset is reached. 1 If OFFEN is set, the timer resets to zero when the offset setting is reached. The offset event does not cause a timer interrupt.
2 OFFEN	Enable One-Shot Offset Event 0 Disable. 1 The timer can be reset to zero when the given offset time is reached (offset event). The field is cleared when the offset event is reached, so no further event occurs until the field is set again. The timer offset value must be set before setting this field.
1 Reserved	This field is reserved.
0 EN	Enable Timer 0 The timer stops at the current value. 1 The timer starts incrementing.

22.5.90 Timer Value Register (ENETx_ATVR)

Address: Base address + 404h offset

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

ENETx_ATVR field descriptions

Field	Description
ATIME	A write sets the timer. A read returns the last captured value. To read the current value, issue a capture command (i.e., set ATCR[CAPTURE]) prior to reading this register.

22.5.91 Timer Offset Register (ENETx_ATOFF)

Address: Base address + 408h offset

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

ENETx_ATOFF field descriptions

Field	Description
OFFSET	Offset value for one-shot event generation. When the timer reaches the value, an event can be generated to reset the counter. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds.

22.5.92 Timer Period Register (ENETx_ATPER)

Address: Base address + 40Ch offset

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

ENETx_ATPER field descriptions

Field	Description
PERIOD	Value for generating periodic events. Each instance the timer reaches this value, the period event occurs and the timer restarts. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds. The value should be initialized to 1,000,000,000 (1×10^9) to represent a timer wrap around of one second. The increment value set in ATINC should be set to the true nanoseconds of the period of clock ts_clk, hence implementing a true 1 second counter. **NOTE:** The value of PERIOD has the following constraint:

ENETx_ATPER field descriptions (continued)

Field	Description
	2^{32} – ENET_ATINC[INC_COR] – 3×ENET_ATINC[INC] ≥ PERIOD > 0.

22.5.93 Timer Correction Register (ENETx_ATCOR)

Address: Base address + 410h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	COR														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_ATCOR field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COR	Correction Counter Wrap-Around Value Defines after how many timer clock cycles (ts_clk) the correction counter should be reset and trigger a correction increment on the timer. The amount of correction is defined in ATINC[INC_CORR]. A value of 0 disables the correction counter and no corrections occur. **NOTE:** This value is given in clock cycles, not in nanoseconds as all other values.

22.5.94 Time-Stamping Clock Period Register (ENETx_ATINC)

Address: Base address + 414h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ENETx_ATINC field descriptions

Field	Description
31–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 INC_CORR	Correction Increment Value This value is added every time the correction timer expires (every clock cycle given in ATCOR). A value less than INC slows down the timer. A value greater than INC speeds up the timer.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
INC	Clock Period Of The Timestamping Clock (ts_clk) In Nanoseconds The timer increments by this amount each clock cycle. For example, set to 10 for 100 MHz, 8 for 125 MHz, 5 for 200 MHz. **NOTE:** For highest precision, use a value that is an integer fraction of the period set in ATPER.

22.5.95 Timestamp of Last Transmitted Frame (ENETx_ATSTMP)

Address: Base address + 418h offset

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

ENETx_ATSTMP field descriptions

Field	Description
TIMESTAMP	Timestamp of the last frame transmitted by the core that had TxBD[TS] set . This register is only valid when EIR[TS_AVAIL] is set.

22.5.96 Timer Global Status Register (ENETx_TGSR)

Address: Base address + 604h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												TF3	TF2	TF1	TF0
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_TGSR field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 TF3	Copy Of Timer Flag For Channel 3 0 Timer Flag for Channel 3 is clear 1 Timer Flag for Channel 3 is set
2 TF2	Copy Of Timer Flag For Channel 2 0 Timer Flag for Channel 2 is clear 1 Timer Flag for Channel 2 is set
1 TF1	Copy Of Timer Flag For Channel 1 0 Timer Flag for Channel 1 is clear 1 Timer Flag for Channel 1 is set
0 TF0	Copy Of Timer Flag For Channel 0 0 Timer Flag for Channel 0 is clear 1 Timer Flag for Channel 0 is set

22.5.97 Timer Control Status Register (ENETx_TCSR*n*)

Address: Base address + 608h offset + (8d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TPWC					0			TF	TIE	TMODE				0	TDRE
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENETx_TCSR*n* field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–11 TPWC	Timer PulseWidth Control Specifies the pulse width associated with TMODE values of 1110 or 11X1. Updating this field takes a few cycles to register because it is synchronized to the 1588 clock. When changing this field: 1. Always disable the channel and read the TMODE field to verify that the channel is disabled. 2. Set TPWC to the desired value. 3. Reenable the channel.

Table continues on the next page...

ENETx_TCSR*n* field descriptions (continued)

Field	Description
	00000 Pulse width is one 1588-clock cycle. 00001 Pulse width is two 1588-clock cycles. 00010 Pulse width is three 1588-clock cycles. 00011 Pulse width is four 1588-clock cycles. ... ... 11111 Pulse width is 32 1588-clock cycles.
10–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TF	Timer Flag Sets when input capture or output compare occurs. This flag is double buffered between the module clock and 1588 clock domains. When this field is 1, it can be cleared to 0 by writing 1 to it. 0 Input Capture or Output Compare has not occurred. 1 Input Capture or Output Compare has occurred.
6 TIE	Timer Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
5–2 TMODE	Timer Mode Updating the Timer Mode field takes a few cycles to register because it is synchronized to the 1588 clock. The version of Timer Mode returned on a read is from the 1588 clock domain. When changing Timer Mode, always disable the channel and read this register to verify the channel is disabled first. 0000 Timer Channel is disabled. 0001 Timer Channel is configured for Input Capture on rising edge. 0010 Timer Channel is configured for Input Capture on falling edge. 0011 Timer Channel is configured for Input Capture on both edges. 0100 Timer Channel is configured for Output Compare - software only. 0101 Timer Channel is configured for Output Compare - toggle output on compare. 0110 Timer Channel is configured for Output Compare - clear output on compare. 0111 Timer Channel is configured for Output Compare - set output on compare. 1000 Reserved 1010 Timer Channel is configured for Output Compare - clear output on compare, set output on overflow. 10X1 Timer Channel is configured for Output Compare - set output on compare, clear output on overflow. 110X Reserved 1110 Timer Channel is configured for Output Compare - pulse output low on compare for 1 to 32 1588-clock cycles as specified by TPWC. 1111 Timer Channel is configured for Output Compare - pulse output high on compare for 1 to 32 1588-clock cycles as specified by TPWC.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 TDRE	Timer DMA Request Enable 0 DMA request is disabled 1 DMA request is enabled

22.5.98 Timer Compare Capture Register (ENETx_TCCR*n*)

Address: Base address + 60Ch offset + (8d × i), where i=0d to 3d

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

ENETx_TCCR*n* field descriptions

Field	Description
TCC	Timer Capture Compare This register is double buffered between the module clock and 1588 clock domains. When configured for compare, the 1588 clock domain updates with the value in the module clock domain whenever the Timer Channel is first enabled and on each subsequent compare. Write to this register with the first compare value before enabling the Timer Channel. When the Timer Channel is enabled, write the second compare value either immediately, or at least before the first compare occurs. After each compare, write the next compare value before the previous compare occurs and before clearing the Timer Flag. The compare occurs one 1588 clock cycle after the IEEE 1588 Counter increments past the compare value in the 1588 clock domain. If the compare value is less than the value of the 1588 Counter when the Timer Channel is first enabled, then the compare does not occur until following the next overflow of the 1588 Counter. If the compare value is greater than the IEEE 1588 Counter when the 1588 Counter overflows, or the compare value is less than the value of the IEEE 1588 Counter after the overflow, then the compare occurs one 1588 clock cycle following the overflow. When configured for capture, the value of the IEEE 1588 Counter is captured into the 1588 clock domain and then updated into the module clock domain, provided the Timer Flag is clear. Always read the capture value before clearing the Timer Flag.

22.6 Functional description

This section provides a complete functional description of the MAC-NET core.

22.6.1 Ethernet MAC frame formats

The IEEE 802.3 standard defines the Ethernet frame format as follows:

- Minimum length of 64 bytes
- Maximum length of 1518 bytes excluding the preamble and the start frame delimiter (SFD) bytes

An Ethernet frame consists of the following fields:

- Seven bytes preamble
- Start frame delimiter (SFD)
- Two address fields
- Length or type field
- Data field
- Frame check sequence (CRC value)

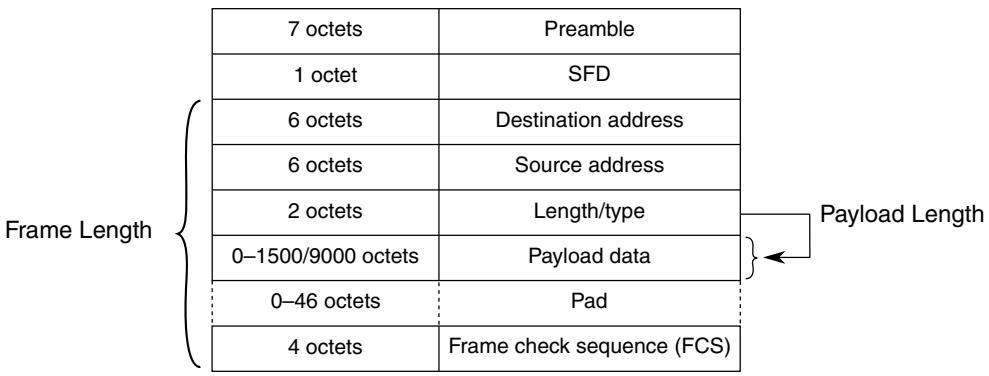


Figure 22-2. MAC frame format overview

Optionally, MAC frames can be VLAN-tagged with an additional four-byte field inserted between the MAC source address and the type/length field. VLAN tagging is defined by the IEEE P802.1q specification. VLAN-tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes.

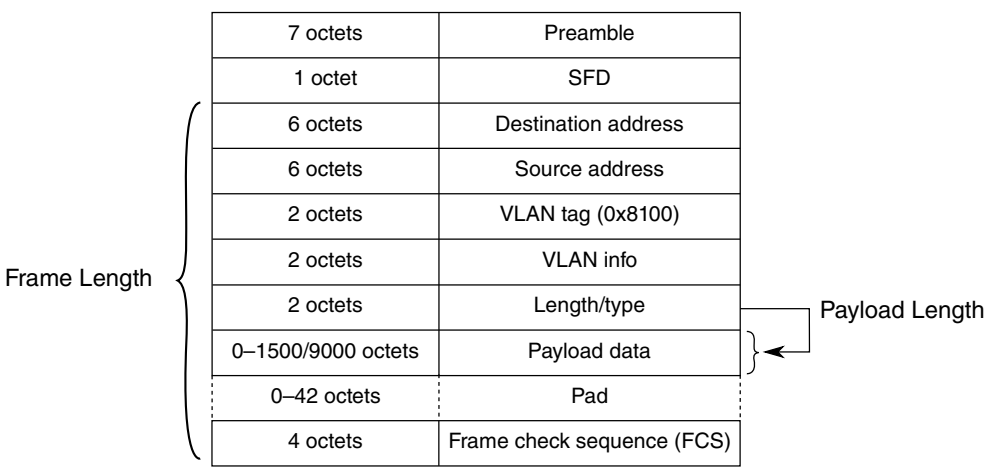


Figure 22-3. VLAN-tagged MAC frame format overview

Table 22-5. MAC frame definition

Term	Description
Frame length	Defines the length, in octets, of the complete frame without preamble and SFD. A frame has a valid length if it contains at least 64 octets and does not exceed the programmed maximum length.
Payload length	The length/type field indicates the length of the frame's payload section. The most significant byte is sent/received first. • If the length/type field is set to a value less than 46, the payload is padded so that the minimum frame length requirement (64 bytes) is met. For VLAN-tagged frames, a value less than 42 indicates a padded frame. • If the length/type field is set to a value larger than the programmed frame maximum length (e.g. 1518) it is interpreted as a type field.
Destination and source address	48-bit MAC addresses. The least significant byte is sent/received first and the first two least significant bits of the MAC address distinguish MAC frames, as detailed in MAC address check.

Note

Although the IEEE specification defines a maximum frame length, the MAC core provides the flexibility to program any value for the frame maximum length.

22.6.1.1 Pause Frames

The receiving device generates a pause frame to indicate a congestion to the emitting device, which should stop sending data.

Pause frames are indicated by the length/type set to 0x8808. The two first bytes of a pause frame following the type, defines a 16-bit opcode field set to 0x0001 always. A 16-bit pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2) as defined in the following table. The P1 pause quanta byte is the most significant.

Table 22-6. Pause Frame Format (Values in Hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
						Preamble	SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27 –68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
				Source Address		Type		Opcode		P1	P2	pad (42)	
69	70	71	72										
26	6B	AE	0A										
			CRC-32										

There is no payload length field found within a pause frame and a pause frame is always padded with 42 bytes (0x00).

If a pause frame with a pause value greater than zero (XOFF condition) is received, the MAC stops transmitting data as soon the current frame transfer is completed. The MAC stops transmitting data for the value defined in pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON condition) is received, the transmitter is allowed to send data immediately (see Full-duplex flow control operation for details).

22.6.1.2 Magic packets

A magic packet is a unicast, multicast, or broadcast packet, which carries a defined sequence in the payload section.

Magic packets are received and inspected only under specific conditions as described in Magic packet detection.

The defined sequence to decode a magic packet is formed with a synchronization stream which consists of six consecutive 0xFF bytes, and is followed by sequence of sixteen consecutive unicast MAC addresses of the node to be awakened.

This sequence can be located anywhere in the magic packet payload. The magic packet is formed with a standard Ethernet header, optional padding, and CRC.

22.6.2 IP and higher layers frame format

The following sections use the term datagram to describe the protocol specific data unit that is found within the payload section of its container entity.

For example, an IP datagram specifies the payload section of an Ethernet frame. A TCP datagram specifies the payload section within an IP datagram.

22.6.2.1 Ethernet types

IP datagrams are carried in the payload section of an Ethernet frame. The Ethernet frame type/length field discriminates several datagram types.

The following table lists the types of interest:

Table 22-7. Ethernet type value examples

Type	Description
0x8100	VLAN-tagged frame. The actual type is found 4 octets later in the frame.
0x0800	IPv4
0x0806	ARP
0x86DD	IPv6

22.6.2.2 IPv4 datagram format

The following figure shows the IP Version 4 (IPv4) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words. The first byte sent/received is the leftmost byte of the first word (in other words, version/IHL field).

The IP header can contain further options, which are always padded if necessary to guarantee the payload following the header is aligned to a 32-bit boundary.

The IP header is immediately followed by the payload, which can contain further protocol headers (for example, TCP or UDP, as indicated by the protocol field value). The complete IP datagram is transported in the payload section of an Ethernet frame.

Table 22-8. IPv4 header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Version	IHL	TOS		Length			
Fragment ID				Flags	Fragment offset		
TTL		Protocol		Header checksum			
Source address							
Destination address							
Options							

Table 22-9. IPv4 header fields

Field name	Description
Version	4-bit IP version information. 0x4 for IPv4 frames.
IHL	4-bit Internet header length information. Determines number of 32-bit words found within the IP header. If no options are present, the default value is 0x5.
TOS	Type of service/DiffServ field.
Length	Total length of the datagram in bytes, including all octets of header and payload.
Fragment ID, flags, fragment offset	Fields used for IP fragmentation.

Table continues on the next page...

Table 22-9. IPv4 header fields (continued)

Field name	Description
TTL	Time-to-live. In effect, is decremented at each router arrival. If zero, datagram must be discarded.
Protocol	Identifier of protocol that follows in the datagram.
Header checksum	Checksum of IP header. For computational purposes, this field's value is zero.
Source address	Source IP address.
Destination address	Destination IP address.

22.6.2.3 IPv6 datagram format

The following figure shows the IP version 6 (IPv6) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words and has a fixed length of ten words (40 bytes). The next header field identifies the type of the header that follows the IPv6 header. It is defined similar to the protocol identifier within IPv4, with new definitions for identifying extension headers. These headers can be inserted between the IPv6 header and the protocol header, which will shift the protocol header accordingly.The accelerator currently only supports IPv6 without extension headers (in other words, the next header specifies TCP, UDP, or IMCP).

The first byte sent/received is the leftmost byte of the first word (in other words, version/ traffic class fields).

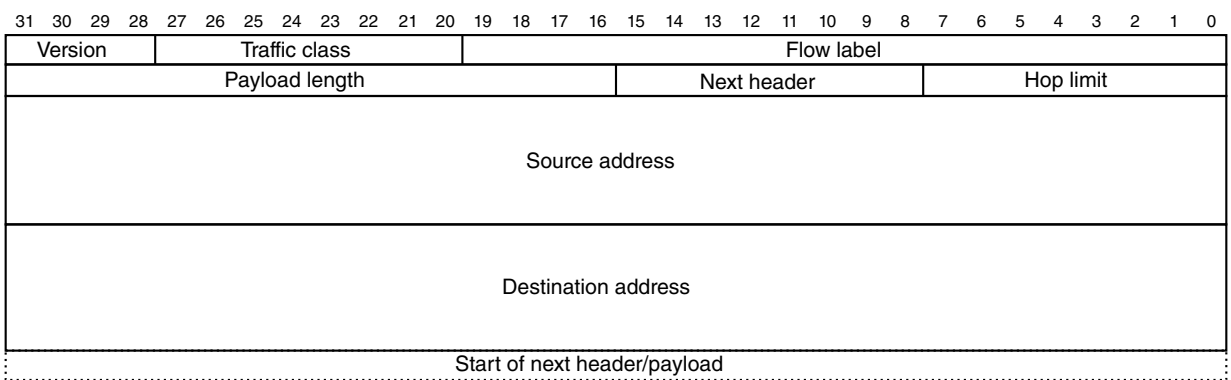


Figure 22-4. IPv6 header format

Table 22-10. IPv6 header fields

Field name	Description
Version	4-bit IP version information. 0x6 for all IPv6 frames.
Traffic class	8-bit field defining the traffic class.
Flow label	20-bit flow label identifying frames of the same flow.
Payload length	16-bit length of the datagram payload in bytes. It includes all octets following the IPv6 header.

Table continues on the next page...

Table 22-10. IPv6 header fields (continued)

Field name	Description
Next header	Identifies the header that follows the IPv6 header. This can be the protocol header or any IPv6 defined extension header.
Hop limit	Hop counter, decremented by one by each station that forwards the frame. If hop limit is 0 the frame must be discarded.
Source address	128-bit IPv6 source address.
Destination address	128-bit IPv6 destination address.

22.6.2.4 Internet Control Message Protocol (ICMP) datagram format

An internet control message protocol (ICMP) is found following the IP header, if the protocol identifier is 1. The ICMP datagram has a four-octet header followed by additional message data.

Table 22-11. ICMP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Type		Code		Checksum			
ICMP message data							

Table 22-12. IP header fields

Field name	Description
Type	8-bit type information
Code	8-bit code that is related to the message type
Checksum	16-bit one's complement checksum over the complete ICMP datagram

22.6.2.5 User Datagram Protocol (UDP) datagram format

A user datagram protocol header is found after the IP header, when the protocol identifier is 17.

The payload of the datagram is after the UDP header. The header byte order follows the conventions given for the IP header above.

Table 22-13. UDP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Length				Checksum			

Table 22-14. UDP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Length	Length of user data which immediately follows the header, including the UDP header (that is, minimum value is 8)
Checksum	Checksum over the complete datagram and some IP header information

22.6.2.6 TCP datagram format

A TCP header is found following the IP header, when the protocol identifier has a value of 6.

The TCP payload immediately follows the TCP header.

Table 22-15. TCP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Sequence number							
Acknowledgement number							
Offset	Reserved		Flags	Window			
Checksum				Urgent pointer			
Options							

Table 22-16. TCP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Sequence number	Transmit sequence number
Ack. number	Receive sequence number
Offset	Data offset, which is number of 32-bit words within TCP header — if no options selected, defaults to value of 5
Flags	URG, ACK, PSH, RST, SYN, FIN flags
Window	TCP receive window size information
Checksum	Checksum over the complete datagram (TCP header and data) and IP header information
Options	Additional 32-bit words for protocol options

22.6.3 IEEE 1588 message formats

The following sections describe the IEEE 1588 message formats.

22.6.3.1 Transport encapsulation

The precision time protocol (PTP) datagrams are encapsulated in Ethernet frames using the UDP/IP transport mechanism, or optionally, with the newer 1588v2 directly in Ethernet frames (layer 2).

Typically, multicast addresses are used to allow efficient distribution of the synchronization messages.

22.6.3.1.1 UDP/IP

The 1588 messages (v1 and v2) can be transported using UDP/IP multicast messages.

Table 22-17 shows IP multicast groups defined for PTP. The table also shows their respective MAC layer multicast address mapping according to RFC 1112 (last three octets of IP follow the fixed value of 01-00-5E).

Table 22-17. UDP/IP multicast domains

Name	IP Address	MAC Address mapping
DefaultPTPdomain	224.0.1.129	01-00-5E-00-01-81
AlternatePTPdomain1	224.0.1.130	01-00-5E-00-01-82
AlternatePTPdomain2	224.0.1.131	01-00-5E-00-01-83
AlternatePTPdomain3	224.0.1.132	01-00-5E-00-01-84

Table 22-18. UDP port numbers

Message type	UDP port	Note
Event	319	Used for SYNC and DELAY_REQUEST messages
General	320	All other messages (for example, follow-up, delay-response)

22.6.3.1.2 Native Ethernet (PTPv2)

In addition to using UDP/IP frames, IEEE 1588v2 defines a native Ethernet frame format that uses ethertype = 0x88F7. The payload of the Ethernet frame immediately contains the PTP datagram, starting with the PTPv2 header.

Besides others, version 2 adds a peer delay mechanism to allow delay measurements between individual point-to-point links along a path over multiple nodes. The following multicast domains are also defined in PTPv2.

Table 22-19. PTPv2 multicast domains

Name	MAC address
Normal messages	01-1B-19-00-00-00
Peer delay messages	01-80-C2-00-00-0E

22.6.3.2 PTP header

All PTP frames contain a common header that determines the protocol version and the type of message, which defines the remaining content of the message.

All multi-octet fields are transmitted in big-endian order (the most significant byte is transmitted/received first).

The last four bits of versionPTP are at the same position (second byte) for PTPv1 and PTPv2 headers. This allows accurate identification by inspecting the first two bytes of the message.

22.6.3.2.1 PTPv1 header

Table 22-20. Common PTPv1 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	2	versionPTP = 0x0001							
2	2	versionNetwork							
4	16	subdomain							
20	1	messageType							
21	1	sourceCommunicationTechnology							
22	6	sourceUuid							
28	2	sourcePortId							
30	2	sequenceId							
32	1	control							
33	1	0x00							
34	2	flags							
36	4	reserved							

The type of message is encoded in the messageType and control fields as shown in Table 22-21 :

Table 22-21. PTPv1 message type identification

messageType	control	Message Name	Message
0x01	0x0	SYNC	Event message
0x01	0x1	DELAY_REQ	Event message
0x02	0x2	FOLLOW_UP	General message
0x02	0x3	DELAY_RESP	General message
0x02	0x4	MANAGEMENT	General message
other	other	—	Reserved

The field sequenceId is used to non-ambiguously identify a message.

22.6.3.2.2 PTPv2 header

Table 22-22. Common PTPv2 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	1	transportSpecific				messageId			
1	1	reserved				versionPTP = 0x2			
2	2	messageLength							
4	1	domainNumber							
5	1	reserved							
6	2	flags							
8	8	correctionField							
16	4	reserved							
20	10	sourcePortIdentity							
30	2	sequenceId							
32	1	control							
33	1	logMeanMessageInterval							

The type of message is encoded in the field messageId as follows:

Table 22-23. PTPv2 message type identification

messageId	Message name	Message
0x0	SYNC	Event message
0x1	DELAY_REQ	Event message
0x2	PATH_DELAY_REQ	Event message
0x3	PATH_DELAY_RESP	Event message
0x4–0x7	—	Reserved
0x8	FOLLOW_UP	General message
0x9	DELAY_RESP	General message

Table continues on the next page...

Table 22-23. PTPv2 message type identification (continued)

messageId	Message name	Message
0xa	PATH_DELAY_FOLLOW_UP	General message
0xb	ANNOUNCE	General message
0xc	SIGNALING	General message
0xd	MANAGEMENT	General message

The PTPv2 flags field contains further details on the type of message, especially if one-step or two-step implementations are used. The one- or two-step implementation is controlled by the TWO_STEP bit in the first octet of the flags field as shown below. Reserved bits are cleared.

Table 22-24. PTPv2 message flags field definitions

Bit	Name	Description
0	ALTERNATE_MASTER	See IEEE 1588 Clause 17.4
1	TWO_STEP	1 Two-step clock 0 One-step clock
2	UNICAST	1 Transport layer address uses a unicast destination address 0 Multicast is used
3	—	Reserved
4	—	Reserved
5	Profile specific	
6	Profile specific	
7	—	Reserved

22.6.4 MAC receive

The MAC receive engine performs the following tasks:

- Check frame framing
- Remove frame preamble and frame SFD field
- Discard frame based on frame destination address field
- Terminate pause frames
- Check frame length
- Remove payload padding if it exists

- Calculate and verify CRC-32
- Write received frames in the core receive FIFO

If the MAC is programmed to operate in half-duplex mode, it will also check if the frame is received with a collision.

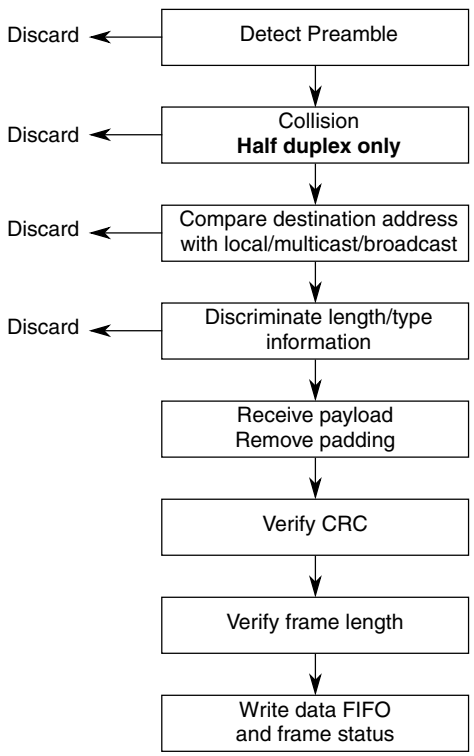


Figure 22-5. MAC receive flow

22.6.4.1 Collision detection in half-duplex mode

If the packet is received with a collision detected during reception of the first 64 bytes, the packet is discarded (if frame size was less than ~14 octets) or transmitted to the user application with an error and RxBD[CE] set.

22.6.4.2 Preamble processing

The IEEE 802.3 standard allows a maximum size of 56 bits (seven bytes) for the preamble, while the MAC core allows any preamble length, including zero length preamble.

The MAC core checks for the start frame delimiter (SFD) byte. If the next byte of the preamble, which is different from 0x55, is not 0xD5, the frame is discarded.

Although the IEEE specification dictates that the inner-packet gap should be at least 96 bits, the MAC core is designed to accept frames separated by only 64 10/100-Mbit/s operation (MII) bits.

The MAC core removes the preamble and SFD bytes.

22.6.4.3 MAC address check

The destination address bit 0 differentiates between multicast and unicast addresses.

- If bit 0 is 0, the MAC address is an individual (unicast) address.
- If bit 0 is 1, the MAC address defines a group (multicast) address.
- If all 48 bits of the MAC address are set, it indicates a broadcast address.

22.6.4.3.1 Unicast address check

If a unicast address is received, the destination MAC address is compared to the node MAC address programmed by the host in the PADDR1/2 registers.

If the destination address matches any of the programmed MAC addresses, the frame is accepted.

If Promiscuous mode is enabled (RCR[PROM] = 1) no address checking is performed and all unicast frames are accepted.

22.6.4.3.2 Multicast and unicast address resolution

The hash table algorithm used in the group and individual hash filtering operates as follows.

- The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits in ENET*n*_GAUR/GALR (group address hash match) or ENET*n*_IAUR/IALR (individual address hash match).
- This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63.
- The msb of the CRC result selects ENET*n*_GAUR (msb = 1) or ENET*n*_GALR (msb = 0).
- The five lsbs of the hash result select the bit within the selected register.
- If the CRC generator selects a bit set in the hash table, the frame is accepted; else, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

If Promiscuous mode is enabled (ENET*n*_RCR[PROM] = 1) all unicast and multicast frames are accepted regardless of ENET*n*_GAUR/GALR and ENET*n*_IAUR/IALR settings.

22.6.4.3.3 Broadcast address reject

All broadcast frames are accepted if BC_REJ is cleared or ENET*n*_RCR[PROM] is set. If PROM is cleared when ENET*n*_RCR[BC_REJ] is set, all broadcast frames are rejected.

Table 22-25. Broadcast address reject programming

PROM	BC_REJ	Broadcast frames
0	0	Accepted
0	1	Rejected
1	0	Accepted
1	1	Accepted

22.6.4.3.4 Miss-bit implementation

For higher layer filtering purposes, RxBD[M] indicates an address miss when the MAC operates in promiscuous mode and accepts a frame that would otherwise be rejected.

If a group/individual hash or exact match does not occur and Promiscuous mode is enabled (RCR[PROM] = 1), the frame is accepted and the M bit is set in the buffer descriptor; otherwise, the frame is rejected.

This means the status bit is set in any of the following conditions during Promiscuous mode:

- A broadcast frame is received when BC_REJ is set

- A unicast is received that does not match either:
  - Node address (PALR[PADDR1] and PAUR[PADDR2])
  - Hash table for unicast (IAUR[IADDR1] and IALR[IADDR2])
- A multicast is received that does not match the GAUR[GADDR1] and GALR[GADDR2] hash table entries

22.6.4.4 Frame length/type verification: payload length check

If the length/type is less than 0x600 and NLC is set, the MAC checks the payload length and reports any error in the frame status word and interrupt bit PLR.

If the length/type is greater than or equal to 0x600, the MAC interprets the field as a type and no payload length check is performed.

The length check is performed on VLAN and stacked VLAN frames. If a padded frame is received, no length check can be performed due to the extended frame payload because padded frames can never have a payload length error.

22.6.4.5 Frame length/type verification: frame length check

When the receive frame length exceeds MAX_FL bytes, the BABR interrupt is generated and the RxBD[LG] bit is set.

The frame is not truncated unless the frame length exceeds the value programmed in ENET*n*_FTRL[TRUNC_FL]. If the frame is truncated, RxBD[TR] is set. In addition, a truncated frame always has the CRC error indication set (RxBD[CR]).

22.6.4.6 VLAN frames processing

VLAN frames have a length/type field set to 0x8100 immediately followed by a 16-Bit VLAN control information field.

VLAN-tagged frames are received as normal frames because the VLAN tag is not interpreted by the MAC function, and are pushed complete with the VLAN tag to the user application. If the length/type field of the VLAN-tagged frame, which is found four octets later in the frame, is less than 42, the padding is removed. In addition, the frame status word (RxBD[NO]) indicates that the current frame is VLAN tagged.

22.6.4.7 Pause frame termination

The receive engine terminates pause frames and does not transfer them to the receive FIFO. The quanta is extracted and sent to the MAC transmit path via a small internal clock rate decoupling asynchronous FIFO.

The quanta is written only if a correct CRC and frame length are detected by the control state machine. If not, the quanta is discarded and the MAC transmit path is not paused.

Good pause frames are ignored if ENET*n*_RCR[FCE] is cleared and are forwarded to the client interface when ENET*n*_RCR[PAUFWD] is set.

22.6.4.8 CRC check

The CRC-32 field is checked and forwarded to the core FIFO interface if ENET*n*_RCR[CRCFWD] is cleared and ENET*n*_RCR[PADEN] is set.

When CRCFWD is set (regardless of PADEN), the CRC-32 field is checked and terminated (not transmitted to the FIFO).

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the frame check sequence (FCS) field with the x^{31} term as right-most bit of the first octet. The CRC bits are thus received in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

If a CRC error is detected, the frame is marked invalid and RxBD[CR] is set.

22.6.4.9 Frame padding removal

When a frame is received with a payload length field set to less than 46 (42 for VLAN-tagged frames and 38 for frames with stacked VLANs), the zero padding can be removed before the frame is written into the data FIFO depending on the setting of ENET*n*_RCR[PADEN].

Note

If a frame is received with excess padding (in other words, the length field is set as mentioned above, but the frame has more than 64 octets) and padding removal is enabled, then the

padding is removed as normal and no error is reported if the frame is otherwise correct (for example: good CRC, less than maximum length, and no other error).

22.6.5 MAC transmit

Frame transmission starts when the transmit FIFO holds enough data.

After a transfer starts, the MAC transmit function performs the following tasks:

- Generates preamble and SFD field before frame transmission
- Generates XOFF pause frames if the receive FIFO reports a congestion or if ENET*n*_TCR[TFC_PAUSE] is set with ENET*n*_OPD[PAUSE_DUR] set to a non-zero value
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if TFC_PAUSE is set with PAUSE_DUR cleared
- Suspends Ethernet frame transfer (XOFF) if a non-zero pause quanta is received from the MAC receive path
- Adds padding to the frame if required
- Calculates and appends CRC-32 to the transmitted frame
- Sends the frame with correct inter-packet gap (IPG) (deferring)

When the MAC is configured to operate in half-duplex mode, the following additional tasks are performed:

- Collision detection
- Frame retransmit after back-off timer expires

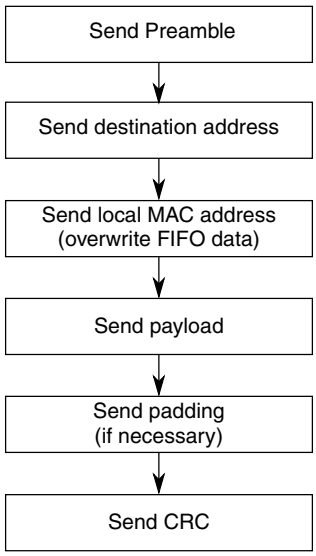


Figure 22-6. Frame transmit overview

22.6.5.1 Frame payload padding

The IEEE specification defines a minimum frame length of 64 bytes.

If the frame sent to the MAC from the user application has a size smaller than 60 bytes, the MAC automatically adds padding bytes (0x00) to comply with the Ethernet minimum frame length specification. Transmit padding is always performed and cannot be disabled.

If the MAC is not allowed to append a CRC (TxBD[TC] = 1), the user application is responsible for providing frames with a minimum length of 64 octets.

22.6.5.2 MAC address insertion

On each frame received from the core transmit FIFO interface, the source MAC address is either:

- Replaced by the address programmed in the PADDR1/2 fields (ENET*n*_TCR[ADDINS] = 1)
- Transparently forwarded to the Ethernet line (ENET*n*_TCR[ADDINS] = 0)

22.6.5.3 CRC-32 generation

The CRC-32 field is optionally generated and appended at the end of a frame.

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the FCS field so that the x^{31} term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

22.6.5.4 Inter-packet gap (IPG)

In full-duplex mode, after frame transmission and before transmission of a new frame, an IPG (programmed in ENET*n*_TIPG) is maintained. The minimum IPG can be programmed between 8 and 26 byte-times (64 and 208 bit-times).

In half-duplex mode, the core constantly monitors the line. Actual transmission of the data onto the network occurs only if it has been idle for a 96-bit time period, and any back-off time requirements have been satisfied. In accordance with the standard, the core begins to measure the IPG from CRS de-assertion.

22.6.5.5 Collision detection and handling — half-duplex operation only

A collision occurs on a half-duplex network when concurrent transmissions from two or more nodes take place. During transmission, the core monitors the line condition and detects a collision when the PHY device asserts COL.

When the core detects a collision while transmitting, it stops transmission of the data and transmits a 32-bit jam pattern. If the collision is detected during the preamble or the SFD transmission, the jam pattern is transmitted after completing the SFD, which results in a minimum 96-bit fragment. The jam pattern is a fixed pattern that is not compared to the actual frame CRC, and has a very low probability (0.532) of having a jam pattern identical to the CRC.

If a collision occurs before transmission of 64 bytes (including preamble and SFD), the MAC core waits for the backoff period and retransmits the packet data (stored in a 64-byte re-transmit buffer) that has already been sent on the line. The backoff period is generated from a pseudo-random process (truncated binary exponential backoff).

If a collision occurs after transmission of 64 bytes (including preamble and SFD), the MAC discards the remainder of the frame, optionally sets the LC interrupt bit, and sets TxBD[LCE].

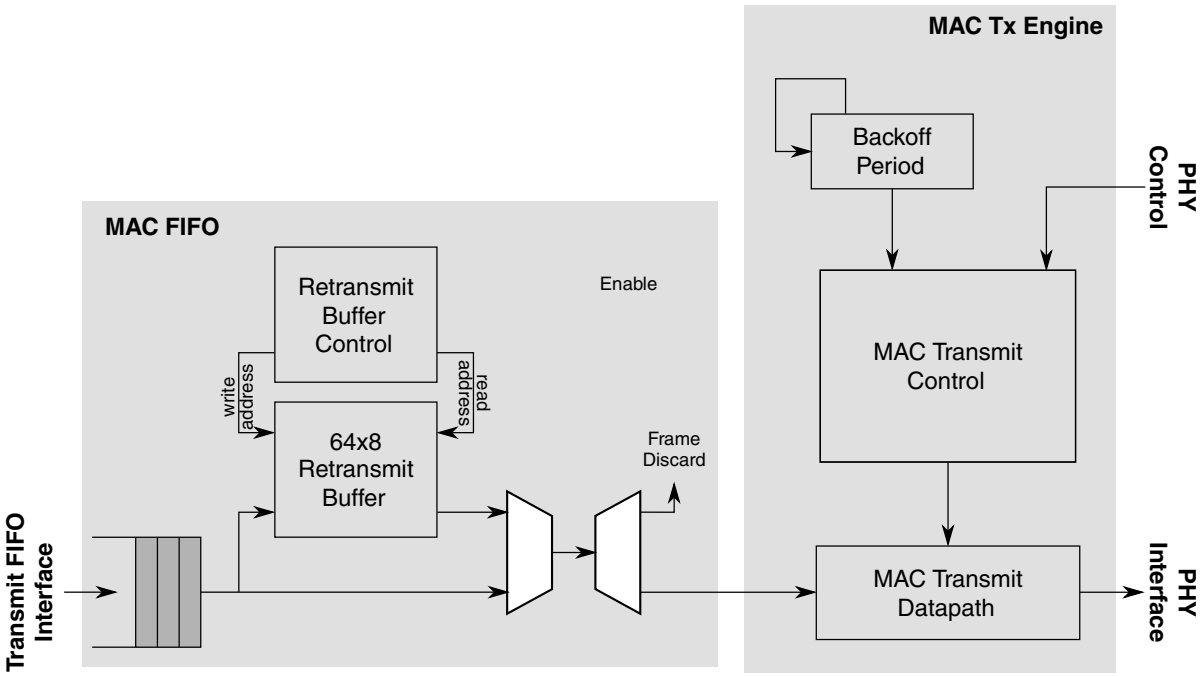


Figure 22-7. Packet re-transmit overview

The backoff time is represented by an integer multiple of slot times. One slot is equal to a 512-bit time period. The number of the delay slot times, before the n^{th} re-transmission attempt, is chosen as a uniformly-distributed random integer in the range:

- $0 < r < 2^k$
- $k = \min(n, N)$; where n is the number of retransmissions and $N = 10$

For example, after the first collision, the backoff period is 0 or 1 slot time. If a collision occurs on the first retransmission, the backoff period is 0, 1, 2, or 3, and so on.

The maximum backoff time (in 512-bit time slots) is limited by N = 10 as specified in the IEEE 802.3 standard.

If a collision occurs after 16 consecutive retransmissions, the core reports an excessive collision condition (ENET*n*_EIR[RL] interrupt field and TxBD[EE]) and discards the current packet from the FIFO.

In networks violating the standard requirements, a collision may occur after transmission of the first 64 bytes. In this case, the core stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The core resumes transmission with the next packet available in the core transmit FIFO.

warning

Ethernet PHYs that support the SQE Test, or "heartbeat,"
feature must disable this feature. When this feature is enabled,

the PHY asserts the collision signal after a frame is transmitted to indicate to the ENET that the PHY's collision logic is working. This may cause data corruption in the next frame from the ENET. This corrupted frame contains up to 21 zero bytes which start somewhere within the MAC destination address field. The ENET, however, will still generate a good FCS (CRC-32) but with corrupted data.

22.6.6 Full-duplex flow control operation

Three conditions are handled by the core's flow control engine:

- Remote device congestion — The remote device connected to the same Ethernet segment as the core reports congestion and requests that the core stop sending data.
- Core FIFO congestion — When the core's receive FIFO reaches a user-programmable threshold (RX section empty), the core sends a pause frame back to the remote device requesting the data transfer to stop.
- Local device congestion — Any device connected to the core can request (typically, via the host processor) the remote device to stop transmitting data.

22.6.6.1 Remote device congestion

When the MAC transmit control gets a valid pause quanta from the receive path and if ENET*n*_RCR[FCE] is set, the MAC transmit logic:

- Completes the transfer of the current frame.
- Stops sending data for the amount of time specified by the pause quanta in 512 bit time increments.
- Sets ENET*n*_TCR[RFC_PAUSE].

Frame transfer resumes when the time specified by the quanta expires and if no new quanta value is received, or if a new pause frame with a quanta value set to 0x0000 is received. The MAC also resets RFC_PAUSE to zero.

If ENET*n*_RCR[FCE] cleared, the MAC ignores received pause frames.

Optionally and independent of ENET*n*_RCR[FCE], pause frames are forwarded to the client interface if PAUFWD is set.

22.6.6.2 Local device/FIFO congestion

The MAC transmit engine generates pause frames when the local receive FIFO is not able to receive more than a pre-defined number of words (FIFO programmable threshold) or when pause frame generation is requested by the local host processor.

- To generate a pause frame, the host processor sets ENET*n*_TCR[TFC_PAUSE]. A single pause frame is generated when the current frame transfer is completed and TFC_PAUSE is automatically cleared. Optionally, an interrupt is generated.
- An XOFF pause frame is generated when the receive FIFO asserts its section empty flag (internal). An XOFF pause frame is generated automatically, when the current frame transfer completes.
- An XON pause frame is generated when the receive FIFO deasserts its section empty flag (internal). An XON pause frame is generated automatically, when the current frame transfer completes.

When an XOFF pause frame is generated, the pause quanta (payload byte P1 and P2) is filled with the value programmed in ENET*n*_OPD[PAUSE_DUR].

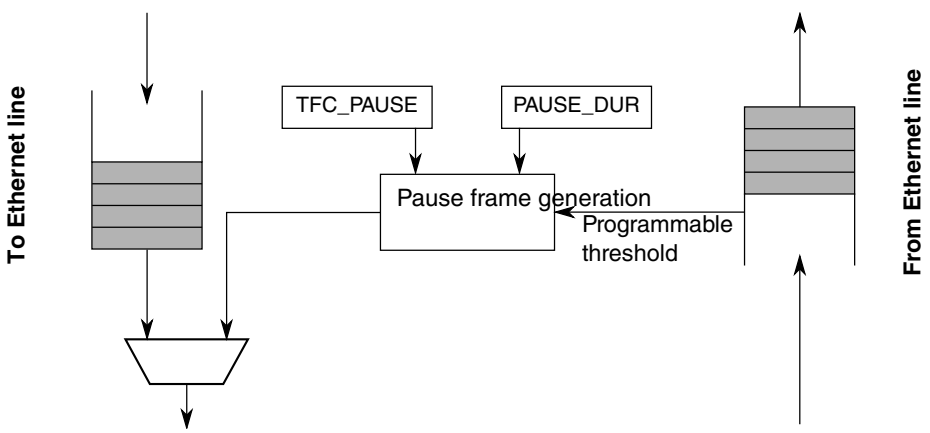


Figure 22-8. Pause frame generation overview

Note

Although the flow control mechanism should prevent any FIFO overflow on the MAC core receive path, the core receive FIFO is protected. When an overflow is detected on the receive FIFO, the current frame is truncated with an error indication set in the frame status word. The frame should subsequently be discarded by the user application.

22.6.7 Magic packet detection

Magic packet detection wakes a node that is put in power-down mode by the node management agent. Magic packet detection is supported only if the MAC is configured in sleep mode.

22.6.7.1 Sleep mode

To put the MAC in Sleep mode, set ENET*n*_ECR[SLEEP]. At the same time ENET*n*_ECR[MAGICEN] should also be set to enable magic packet detection.

In addition, when the processor is in Stop mode, Sleep mode is entered, without affecting the ENET*n*_ECR register bits.

When the MAC is in Sleep mode:

- The transmit logic is disabled.
- The FIFO receive/transmit functions are disabled.
- The receive logic is kept in Normal mode, but it ignores all traffic from the line except magic packets. They are detected so that a remote agent can wake the node.

22.6.7.2 Magic packet detection

The core is designed to detect magic packets (see Magic packets) with the destination address set to:

- Any multicast address
- The broadcast address
- The unicast address programmed in PADDR1/2

When a magic packet is detected, EIR[WAKEUP] is set and none of the statistic registers are incremented.

22.6.7.3 Wakeup

When a magic packet is detected, indicated by ENET*n*_EIR[WAKEUP], ENET*n*_ECR[SLEEP] should be cleared to resume normal operation of the MAC. Clearing the SLEEP bit automatically masks ENET*n*_ECR[MAGICEN], disabling magic packet detection.

22.6.8 IP accelerator functions

The following sections describe the IP accelerator functions.

22.6.8.1 Checksum calculation

The IP and ICMP, TCP, UDP checksums are calculated with one's complement arithmetic summing up 16-bit values.

- For ICMP, the checksum is calculated over the complete ICMP datagram, in other words without IP header.
- For TCP and UDP, the checksums contain the header and data sections and values from the IP header, which can be seen as a pseudo-header that is not actually present in the data stream.

Table 22-26. IPv4 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
Zero		Protocol		TCP/UDP length			

Table 22-27. IPv6 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
TCP/UDP length							
Zero						Next header	

The TCP/UDP length value is the length of the TCP or UDP datagram, which is equal to the payload of an IP datagram. It is derived by subtracting the IP header length from the complete IP datagram length that is given in the IP header (IPv4), or directly taken from the IP header (IPv6). The protocol field is the corresponding value from the IP header. The Zero fields are all zeroes.

For IPv6, the complete 128-bit addresses are considered. The next header value identifies the upper layer protocol as either TCP or UDP. It may differ from the next header value of the IPv6 header if extension headers are inserted before the protocol header.

The checksum calculation uses 16-bit words in network byte order: The first byte sent/received is the MSB, and the second byte sent/received is the LSB of the 16-bit value to add to the checksum. If the frame ends on an odd number of bytes, a zero byte is appended for checksum calculation only, and is not actually transmitted.

22.6.8.2 Additional padding processing

According to IEEE 802.3, any Ethernet frame must have a minimum length of 64 octets.

The MAC usually removes padding on receive when a frame with length information is received. Because IP frames have a type value instead of length, the MAC does not remove padding for short IP frames, as it is not aware of the frame contents.

The IP accelerator function can be configured to remove the Ethernet padding bytes that might follow the IP datagram.

On transmit, the MAC automatically adds padding as necessary to fill any frame to a 64-byte length.

22.6.8.3 32-bit Ethernet payload alignment

The data FIFOs allow inserting two additional arbitrary bytes in front of a frame. This extends the 14-byte Ethernet header to a 16-byte header, which leads to alignment of the Ethernet payload, following the Ethernet header, on a 32-bit boundary.

This function can be enabled for transmit and receive independently with the corresponding SHIFT16 bits in the ENET*n*_TACC and ENET*n*_RACC registers.

When enabled, the valid frame data is arranged as shown in Table 22-28.

Table 22-28. 64-bit interface data structure with SHIFT16 enabled

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0	

Table continues on the next page...

Table 22-28. 64-bit interface data structure with SHIFT16 enabled (continued)

Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Any value	Any value
Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6
...							

22.6.8.3.1 Receive processing

When ENET*n*_RACC[SHIFT16] is set, each frame is received with two additional bytes in front of the frame.

The user application must ignore these first two bytes and find the first byte of the frame in bits 23–16 of the first word from the RX FIFO.

Note

SHIFT16 must be set during initialization and kept set during the complete operation, because it influences the FIFO write behavior.

22.6.8.3.2 Transmit processing

When ENET*n*_TACC[SHIFT16] is set, the first two bytes of the first word written (bits 15–0) are discarded immediately by the FIFO write logic.

The SHIFT16 bit can be enabled/disabled for each frame individually if required, but can be changed only between frames.

22.6.8.4 Received frame discard

Because the receive FIFO must be operated in store and forward mode (ENET*n*_RSFL cleared), received frames can be discarded based on the following errors:

- The MAC function receives the frame with an error:
  - The frame has an invalid payload length
  - Frame length is greater than MAX_FL
  - Frame received with a CRC-32 error
  - Frame truncated due to receive FIFO overflow
  - Frame is corrupted as PHY signaled an error (RX_ERR asserted during reception)

- An IP frame is detected and the IP header checksum is wrong
- An IP frame with a valid IP header and a valid IP header checksum is detected, the protocol is known but the protocol-specific checksum is wrong

If one of the errors occurs and the IP accelerator function is configured to discard frames (ENET*n*_RACC), the frame is automatically discarded. Statistics are maintained normally and are not affected by this discard function.

22.6.8.5 IPv4 fragments

When an IPv4 IP fragment frame is received, only the IP header is inspected and its checksum verified. 32-bit alignment operates the same way on fragments as it does on normal IP frames, as specified above.

The IP fragment frame payload is not inspected for any protocol headers. As such, a protocol header would only exist in the very first fragment. To assist in protocol-specific checksum verification, the one's-complement sum is calculated on the IP payload (all bytes following the IP header) and provided with the frame status word.

The frame fragment status field (RxBD[FRAG]) is set to indicate a fragment reception, and the one's-complement sum of the IP payload is available in RxBD[Payload checksum].

Note

After all fragments have been received and reassembled, the application software can take advantage of the payload checksum delivered with the frame's status word to calculate the protocol-specific checksum of the datagram.

For example, if a TCP payload is delivered by multiple IP fragments, the application software can calculate the pseudo-header checksum value from the first fragment, and add the payload checksums delivered with the status for all fragments to verify the TCP datagram checksum.

22.6.8.6 IPv6 support

The following sections describe the IPv6 support.

22.6.8.6.1 Receive processing

An Ethernet frame of type 0x86DD identifies an IP Version 6 frame (IPv6) frame. If an IPv6 frame is received, the first IP header is inspected (first ten words), which is available in every IPv6 frame.

If the receive SHIFT16 function is enabled, the IP header is aligned on a 32-bit boundary allowing more efficient processing (see 32-bit Ethernet payload alignment).

For TCP and UDP datagrams, the pseudo-header checksum calculation is performed and verified.

To assist in protocol-specific checksum verification, the one's-complement sum is always calculated on the IP payload (all bytes following the IP header) and provided with the frame status word. For example, if extension headers were present, their sums can be subtracted in software from the checksum to isolate the TCP/UDP datagram checksum, if required.

22.6.8.6.2 Transmit processing

For IPv6 transmission, the SHIFT16 function is supported to process 32-bit aligned datagrams.

IPv6 has no IP header checksum; therefore, the IP checksum insertion configuration is ignored.

The protocol checksum is inserted only if the next header of the IP header is a known protocol (TCP, UDP, or ICMP). If a known protocol is detected, the checksum over all bytes following the IP header is calculated and inserted in the correct position.

The pseudo-header checksum calculation is performed for TCP and UDP datagrams accordingly.

22.6.9 Resets and stop controls

The following sections describe the resets and stop controls.

22.6.9.1 Hardware reset

To reset the Ethernet module, set ENET*n*_ECR[RESET].

22.6.9.2 Soft reset

When ENET*n*_ECR[ETHER_EN] is cleared during operation, the following occurs:

- uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers.
- A currently ongoing transmit is terminated by asserting TXER to the PHY.
- A currently ongoing transmit FIFO write from the application is terminated by stopping the write to the FIFO, and all further data from the application is ignored. All subsequent writes are ignored until re-enabled.
- A currently ongoing receive FIFO read is terminated. The RxBD has arbitrary values in this case.

22.6.9.3 Hardware freeze

When the processor enters debug mode and ECR[DBGEN] is set, the MAC enters a freeze state where it stops all transmit and receive activities gracefully.

The following happens when the MAC enters hardware freeze:

- A currently ongoing receive transaction on the receive application interface is completed as normal. No further frames are read from the FIFO.
- A currently ongoing transmit transaction on the transmit application interface is completed as normal (in other words, until writing end-of-packet (EOP)).
- A currently ongoing frame receive is completed normally, after which no further frames are accepted from the MII.
- A currently ongoing frame transmit is completed normally, after which no further frames are transmitted.

22.6.9.4 Graceful stop

During a graceful stop, any currently ongoing transactions are completed normally and no further frames are accepted. The MAC can resume from a graceful stop without the need for a reset (for example, clearing ETHER_EN is not required).

The following conditions lead to a graceful stop of the MAC transmit or receive datapaths.

22.6.9.4.1 Graceful transmit stop (GTS)

When gracefully stopped, the MAC is no longer reading frame data from the transmit FIFO and has completed any ongoing transmission.

In any of the following conditions, the transmit datapath stops after an ongoing frame transmission has been completed normally.

- ENET*n*_TCR[GTS] is set by software.
- ENET*n*_TCR[TFC_PAUSE] is set by software requesting a pause frame transmission. The status (and register bit) is cleared after the pause frame has been sent.
- A pause frame was received stopping the transmitter. The stopped situation is terminated when the pause timer expires or a pause frame with zero quanta is received.
- MAC is placed in Sleep mode by software or the processor entering Stop mode (see Sleep mode).
- The MAC is in Hardware Freeze mode.

When the transmitter has reached its stopped state, the following events occur:

- The GRA interrupt is asserted, when transitioned into stopped.
- In Hardware Freeze mode, the GRA interrupt does not wait for the application write completion and asserts when the transmit state machine (in other words, line side of TX FIFO) reaches its stopped state.

22.6.9.4.2 Graceful receive stop (GRS)

When gracefully stopped, the MAC is no longer writing frames into the receive FIFO.

The receive datapath stops after any ongoing frame reception has been completed normally, if any of the following conditions occur:

- MAC is placed in Sleep mode either by the software or the processor is in Stop mode). The MAC continues to receive frames and search for magic packets if enabled (see Magic packet detection). However, no frames are written into the receive FIFO, and therefore are not forwarded to the application.
- The MAC is in Hardware Freeze mode. The MAC does not accept any frames from the MII.

When the receive datapath is stopped, the following events occur:

- If the RX is in the stopped state, RCR[GRS] is set
- The GRA interrupt is asserted when the transmitter and receiver are stopped
- Any ongoing receive transaction to the application (RX FIFO read) continues normally until the frame end of package (EOP) is reached. After this, the following occurs:
  - When Sleep mode is active, all further frames are discarded, flushing the RX FIFO
  - In Hardware Freeze mode, no further frames are delivered to the application and they stay in the receive FIFO.

Note

The assertion of GRS does not wait for an ongoing FIFO read transaction on the application side of the FIFO (FIFO read).

22.6.9.4.3 Graceful stop interrupt (GRA)

The graceful stopped interrupt (GRA) is asserted for the following conditions:

- In Sleep mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- In Hardware Freeze mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- The MAC transmit datapath is stopped for any other condition (GTS, TFC_PAUSE, pause received).

The GRA interrupt is triggered only once when the stopped state is entered. If the interrupt is cleared while the stop condition persists, no further interrupt is triggered.

22.6.10 IEEE 1588 functions

To allow for IEEE 1588 or similar time synchronization protocol implementations, the MAC is combined with a time-stamping module to support precise time-stamping of incoming and outgoing frames. Set ENET*n*_ECR[EN1588] to enable 1588 support.

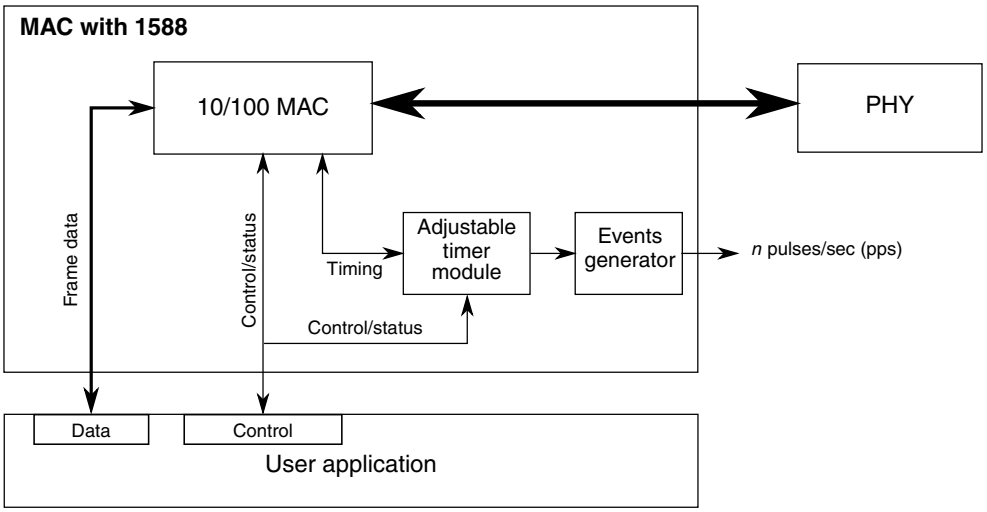


Figure 22-9. IEEE 1588 functions overview

22.6.10.1 Adjustable timer module

The adjustable timer module (TSM) implements the free-running counter (FRC), which generates the timestamps. The FRC operates with the time-stamping clock, which can be set to any value depending on your system requirements.

Through dedicated correction logic, the timer can be adjusted to allow synchronization to a remote master and provide a synchronized timing reference to the local system. The timer can be configured to cause an interrupt after a fixed time period, to allow synchronization of software timers or perform other synchronized system functions.

The timer is typically used to implement a period of one second; hence, its value ranges from 0 to (1×10^9)-1. The period event can trigger an interrupt, and software can maintain the seconds and hours time values as necessary.

22.6.10.1.1 Adjustable timer implementation

The adjustable timer consists of a programmable counter/accumulator and a correction counter. The periods of both counters and their increment rates are freely configurable, allowing very fine tuning of the timer.

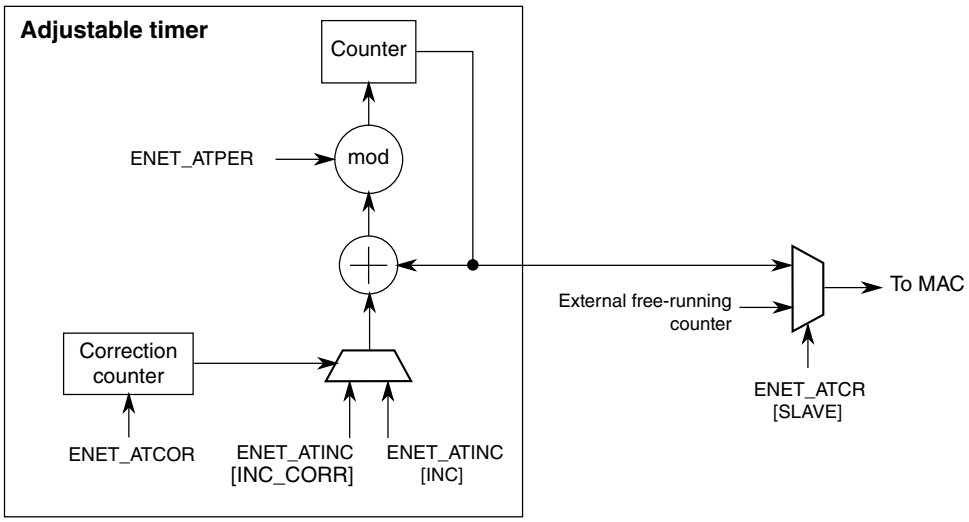


Figure 22-10. Adjustable timer implementation detail

The counter produces the current time. During each time-stamping clock cycle, a constant value is added to the current time as programmed in ENET*n*_ATINC. The value depends on the chosen time-stamping clock frequency. For example, if it operates at 125 MHz, setting the increment to eight represents 8 ns.

The period, configured in ENET*n*_ATPER, defines the modulo when the counter wraps. In a typical implementation, the period is set to 1×10^9 so that the counter wraps every second, and hence all timestamps represent the absolute nanoseconds within the one second period. When the period is reached, the counter wraps to start again respecting the period modulo. This means it does not necessarily start from zero, but instead the counter is loaded with the value (Current + Inc $-(1 \times 10^9)$), assuming the period is set to 1×10^9.

The correction counter operates fully independently, and increments by one with each time-stamping clock cycle. When it reaches the value configured in ENET*n*_ATCOR, it restarts and instructs the timer once to increment by the correction value, instead of the normal value.

The normal and correction increments are configured in ENET*n*_ATINC. To speed up the timer, set the correction increment more than the normal increment value. To slow down the timer, set the correction increment less than the normal increment value.

The correction counter only defines the distance of the corrective actions, not the amount. This allows very fine corrections and low jitter (in the range of 1 ns) independent of the chosen clock frequency.

By enabling slave mode (ENET*n*_ATCR[SLAVE] = 1), the timer is ignored and the current time is externally provided from one of the external modules. See the Chip Configuration details for which clock source is used. This is useful if multiple modules

within the system must operate from a single timer. When slave mode is enabled, you still must set ENET*n*_ATINC[INC] to the value of the master, since it is used for internal comparisons.

22.6.10.2 Transmit timestamping

Only 1588 event frames need to be time-stamped on transmit. The client application (for example, the MAC driver) should detect 1588 event frames and set TxBD[TS] together with the frame.

If TxBD[TS] is set, the MAC records the timestamp for the frame in ENET*n*_ATSTMP. ENET*n*_EIR[TS_AVAIL] is set to indicate that a new timestamp is available.

Software implements a handshaking procedure by setting TxBD[TS] when it transmits the frame for which a timestamp is needed, and then waits for ENET*n*_EIR[TS_AVAIL] to determine when the timestamp is available. The timestamp is then read from ENET*n*_ATSTMP. This is done for all event frames. Other frames do not use TxBD[TS] and, therefore, do not interfere with the timestamp capture.

22.6.10.3 Receive timestamping

When a frame is received, the MAC latches the value of the timer when the frame's start of frame delimiter (SFD) field is detected, and provides the captured timestamp on RxBD[1588 timestamp]. This is done for all received frames.

22.6.10.4 Time synchronization

The adjustable timer module is available to synchronize the local clock of a node to a remote master. It implements a free running 32-bit counter, and also contains an additional correction counter.

The correction counter increases or decreases the rate of the free running counter, enabling very fine granular changes of the timer for synchronization, yet adding only very low jitter when performing corrections.

The application software implements, in a slave scenario, the required control algorithm, setting the correction to compensate for local oscillator drifts and locking the timer to the remote master clock on the network.

The timer and all timestamp-related information should be configured to show the true nanoseconds value of a second (in other words, the timer is configured to have a period of one second). Hence, the values range from 0 to (1×10^9)–1. In this application, the seconds counter is implemented in software using an interrupt function that is executed when the nanoseconds counter wraps at 1×10^9.

22.6.10.5 Input Capture and Output Compare

The Input Capture Output Compare block can be used to provide precise hardware timing for input and output events.

22.6.10.5.1 Input capture

The TCCR*n* capture registers latch the time value when the corresponding external event occurs. An event can be a rising-, falling-, or either-edge of one of the 1588_TMR*n* signals. An event will cause the corresponding TCSR*n*[TF] timer flag to be set, indicating that an input capture has occurred. If the corresponding interrupt is enabled with the TCSR*n*[TIE] field, an interrupt can be generated.

22.6.10.5.2 Output compare

The TCCR*n* compare registers are loaded with the time at which the corresponding event should occur. When the ENET free-running counter value matches the output compare reference value in the TCCR*n* register, the corresponding flag, TCSR*n*[TF], is set, indicating that an output compare has occurred. The corresponding interrupt, if enabled by TCSR*n*[TIE], will be generated.The corresponding 1588_TMR*n* output signal will be asserted according to TCSR*n*[TMODE].

22.6.10.5.3 DMA requests

A DMA request can be enabled by setting TCSR*n*[TDRE]. The corresponding DMA request is generated when the TCSR*n*[TF] timer flag is set. When the DMA has completed, the corresponding TCSRn[TF] flag is cleared.

22.6.11 FIFO thresholds

The core FIFO thresholds are fully programmable to dynamically change the FIFO operation.

For example, store and forward transfer can be enabled by a simple change in the FIFO threshold registers.

The thresholds are defined in 64-bit words.

The receive and transmit FIFOs both have a depth of 256 words.

22.6.11.1 Receive FIFO

Four programmable thresholds are available, which can be set to any value to control the core operation as follows.

Table 22-29. Receive FIFO thresholds definition

Register	Description
ENET*n*_RSFL [RX_SECTION_FULL]	When the FIFO level reaches the ENET*n*_RSFL value, the MAC status signal is asserted to indicate that data is available in the receive FIFO (cut-through operation). Once asserted, if the FIFO empties below the threshold set with ENET*n*_RAEM and if the end-of-frame is not yet stored in the FIFO, the status signal is deasserted again. If a frame has a size smaller than the threshold (in other words, an end-of-frame is available for the frame), the status is also asserted. To enable store and forward on the receive path, clear ENET*n*_RSFL. The MAC status signal is asserted only when a complete frame is stored in the receive FIFO. When programming a non-zero value to ENET*n*_RSFL (cut-through operation) it should be greater than ENET*n*_RAEM.
ENET*n*_RAEM [RX_ALMOST_EMPTY]	When the FIFO level reaches the ENET*n*_RAEM value, and the end-of-frame has not been received, the core receive read control stops the FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. Set ENET*n*_RAEM to a minimum of six.
ENET*n*_RAFL [RX_ALMOST_FULL]	When the FIFO level approaches the maximum and there is no more space remaining for at least ENETn_RAFL number of words, the MAC control logic stops writing data in the FIFO and truncates the receive frame to avoid FIFO overflow. The corresponding error status is set when the frame is delivered to the application. Set ENET*n*_RAFL to a minimum of 4.
ENET*n*_RSEM [RX_SECTION_EMPTY]	When the FIFO level reaches the ENET*n*_RSEM value, an indication is sent to the MAC transmit logic, which generates an XOFF pause frame. This indicates FIFO congestion to the remote Ethernet client. When the FIFO level goes below the value programmed in ENET*n*_RSEM, an indication is sent to the MAC transmit logic, which generates an XON pause frame. This indicates the FIFO congestion is cleared to the remote Ethernet client. Clearing ENET*n*_RSEM disables any pause frame generation.

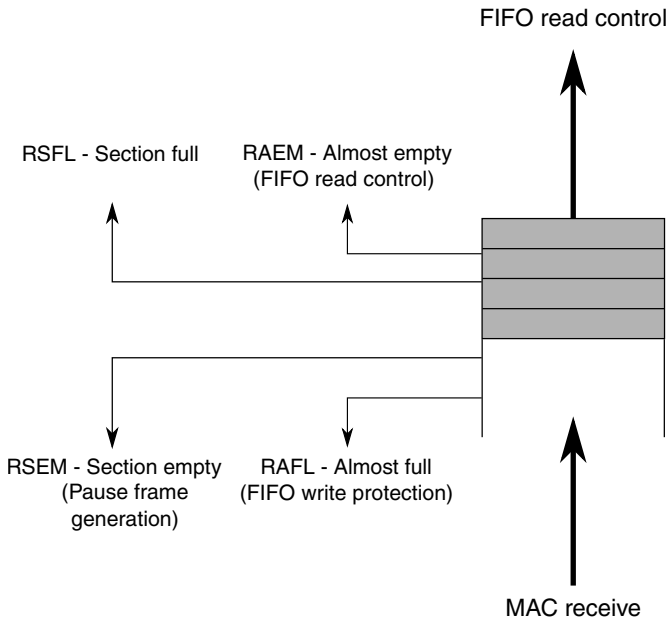


Figure 22-11. Receive FIFO overview

22.6.11.2 Transmit FIFO

Four programmable thresholds are available which control the core operation as described below.

Table 22-30. Transmit FIFO thresholds definition

Register	Description
ENET*n*_TAEM [TX_ALMOST _EMPTY]	When the FIFO level reaches the ENET*n*_TAEM value and no end-of-frame is available for the frame, the MAC transmit logic avoids a FIFO underflow by stopping FIFO reads and transmitting the Ethernet frame with an MII error indication. Set ENET*n*_TAEM to a minimum of 4.
ENET*n*_TAFL [TX_ALMOST _FULL]	When the FIFO level approaches the maximum, so that there is no more space for at least ENET*n*_TAFL number of words, the MAC deasserts its control signal to the application. If the application does not react on this signal, the FIFO write control logic avoids FIFO overflow by truncating the current frame and setting the error status. As a result, the frame is transmitted with an MII error indication. Set ENET*n*_TAFL to a minimum of 4. Larger values allow more latency for the application to react on the MAC control signal deassertion, before the frame is truncated. A typical setting is 8, which offers 3–4 clock cycles of latency to the application to react on the MAC control signal deassertion.
ENET*n*_TSEM [TX_SECTION _EMPTY]	When the FIFO level reaches the ENET*n*_TSEM value, a MAC status signal is deasserted to indicate that the transmit FIFO is getting full. This gives the ENET module an indication to slow or stop its write transaction to avoid a buffer overflow. This is a pure indication function to the application. It has no effect within the MAC. When ENET*n*_TSEM is 0, the signal is never deasserted.
ENET*n*_TFWR	When the FIFO level reaches the ENET*n*_TFWR value and when STRFWD is cleared, the MAC transmit control logic starts frame transmission before the end-of-frame is available in the FIFO (cut-through operation).

Table 22-30. Transmit FIFO thresholds definition

Register	Description
	If a complete frame has a size smaller than the ENET*n*_TFWR threshold, the MAC also transmits the frame to the line. To enable store and forward on the transmit path, set STRFWD. In this case, the MAC starts to transmit data only when a complete frame is stored in the transmit FIFO.

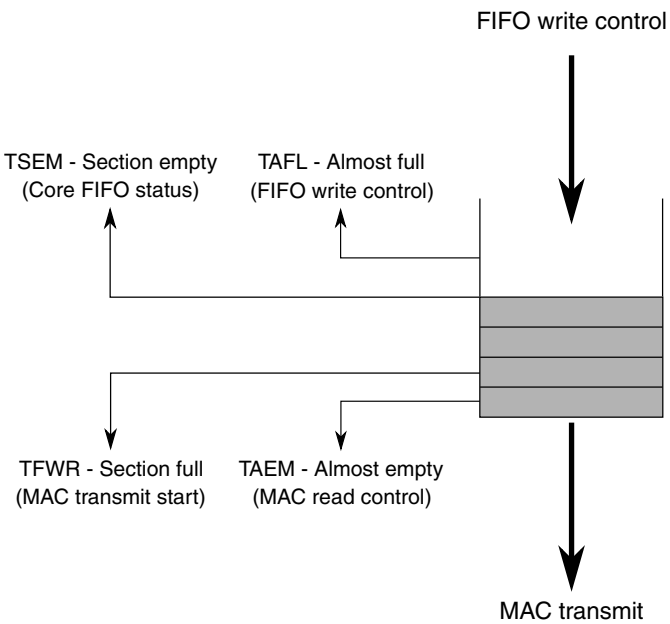


Figure 22-12. Transmit FIFO overview

22.6.12 Loopback options

The core implements external and internal loopback options, which are controlled by the ENET*n*_RCR register fields found here.

The core implements external and internal loopback options, which are controlled by the following ENET*n*_RCR register fields:

Table 22-31. Loopback options

Register field	Description
LOOP	Internal MII loopback. The MAC transmit is returned to the MAC receive. No data is transmitted to the external interfaces. In MII internal loopback, MII_TXCLK and MII_RXCLK must be provided with a clock signal (2.5 MHz for 10 Mbit/s, and 25 MHz for 100 Mbit/s))

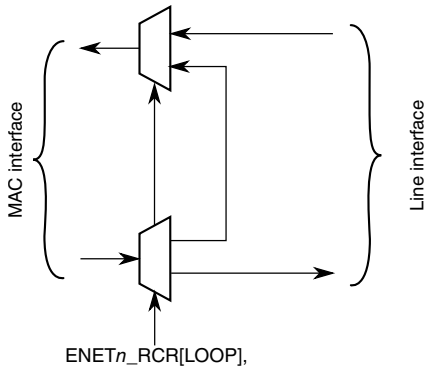


Figure 22-13. Loopback options

22.6.13 Legacy buffer descriptors

To support the Ethernet controller on previous Freescale devices, legacy FEC buffer descriptors are available. To enable legacy support, clear ENET*n*_ECR[1588EN].

NOTE

- The legacy buffer descriptor tables show the byte order for little-endian chips. DBSWP must be set to 1 after reset to enable little-endian mode.

22.6.13.1 Legacy receive buffer descriptor

The following table shows the legacy FEC receive buffer descriptor. Table 22-35 contains the descriptions for each field.

Table 22-32. Legacy FEC receive buffer descriptor (RxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data length															
Offset + 2	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 4	Rx data buffer pointer — low halfword															
Offset + 6	Rx data buffer pointer — high halfword															

22.6.13.2 Legacy transmit buffer descriptor

The following table shows the legacy FEC transmit buffer descriptor. Table 22-37 contains the descriptions for each field.

Table 22-33. Legacy FEC transmit buffer descriptor (TxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data Length															
Offset + 2	R	TO1	W	TO2	L	TC	ABC[1]	—	—	—	—	—	—	—	—	—
Offset + 4	Tx Data Buffer Pointer — low halfword															
Offset + 6	Tx Data Buffer Pointer — high halfword															

1. This field is not supported by the uDMA.

22.6.14 Enhanced buffer descriptors

This section provides a description of the enhanced operation of the driver/uDMA via the buffer descriptors.

It is followed by a detailed description of the receive and transmit descriptor fields. To enable the enhanced features, set ENET*n*_ECR[1588EN].

NOTE

The enhanced buffer descriptor tables show the byte order for little-endian chips. DBSWP must be set to 1 after reset to enable little-endian mode.

22.6.14.1 Enhanced receive buffer descriptor

The following table shows the enhanced uDMA receive buffer descriptor. Table 22-35 contains the descriptions for each field.

Table 22-34. Enhanced uDMA receive buffer descriptor (RxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data length															
Offset + 2	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 4	Rx data buffer pointer – low halfword															
Offset + 6	Rx data buffer pointer – high halfword															

Table continues on the next page...

Table 22-34. Enhanced uDMA receive buffer descriptor (RxBD) (continued)

Offset + 8	VPCP			—	—	—	—	—	—	—	ICE	PCR	—	VLAN	IPV6	FRAG
Offset + A	ME	—	—	—	—	PE	CE	UC	INT	—	—	—	—	—	—	—
Offset + C	Payload checksum															
Offset + E	Header length					—	—	—	Protocol type							
Offset + 10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – low halfword															
Offset + 16	1588 timestamp – high halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 22-35. Receive buffer descriptor field definitions

Word	Field	Description
Offset + 0	15–0 Data Length	Data length. Written by the MAC. Data length is the number of octets written by the MAC into this BD's data buffer if L is cleared (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the MAC once as the BD is closed.
Offset + 2	15 E	Empty. Written by the MAC (= 0) and user (= 1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 2	14 RO1	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 2	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in ENET*n*_RDSR.
Offset + 2	12 RO2	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 2	11 L	Last in frame. Written by the uDMA. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 2	10–9	Reserved, must be cleared.
Offset + 2	8 M	Miss. Written by the MAC. This field is set by the MAC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the this field to quickly determine whether the frame was destined to this station. This field is valid only if the L and PROM bits are set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.

Table continues on the next page...

Table 22-35. Receive buffer descriptor field definitions (continued)

Word	Field	Description
		The information needed for this field comes from the promiscuous_miss(ff_rx_err_stat[26]) sideband signal.
Offset + 2	7 BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 2	6 MC	Set if the DA is multicast and not BC.
Offset + 2	5 LG	Receive frame length violation. Written by the MAC. A frame length greater than RCR[MAX_FL] was recognized. This field is valid only if the L field is set. The receive data is not altered in any way unless the length exceeds TRUNC_FL bytes.
Offset + 2	4 NO	Receive non-octet aligned frame. Written by the MAC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error or a PHY error occurred. This field is valid only if the L field is set. If this field is set, the CR field is not set.
Offset + 2	3	Reserved, must be cleared.
Offset + 2	2 CR	Receive CRC or frame error. Written by the MAC. This frame contains a PHY or CRC error and is an integral number of octets in length. This field is valid only if the L field is set.
Offset + 2	1 OV	Overrun. Written by the MAC. A receive FIFO overrun occurred during frame reception. If this field is set, the other status fields, M, LG, NO, and CR, lose their normal meaning and are zero. This field is valid only if the L field is set.
Offset + 2	0 TR	Set if the receive frame is truncated (frame length >TRUNC_FL). If the TR field is set, the frame must be discarded and the other error fields must be ignored because they may be incorrect.
Offset + 4	15–0 Data buffer pointer low	Receive data buffer pointer, low halfword
0ffset + 6	15–0 Data buffer pointer high	Receive data buffer pointer, high halfword[1]
Offset + 8	15–13 VPCP	VLAN priority code point. This field is written by the uDMA to indicate the frame priority level. Valid values are from 0 (best effort) to 7 (highest). This value can be used to prioritize different classes of traffic (e.g., voice, video, data). This field is only valid if the L field is set.
Offset + 8	12–6	Reserved, must be cleared.
Offset + 8	5 ICE	IP header checksum error. This is an accelerator option. This field is written by the uDMA. Set when either a non-IP frame is received or the IP header checksum was invalid. An IP frame with less than 3 bytes of payload is considered to be an invalid IP frame. This field is only valid if the L field is set.
Offset + 8	4 PCR	Protocol checksum error. This is an accelerator option. This field is written by the uDMA. Set when the checksum of the protocol is invalid or an unknown protocol is found and checksumming could not be performed. This field is only valid if the L field is set.
Offset + 8	3	Reserved, must be cleared.
Offset + 8	2 VLAN	VLAN. This is an accelerator option. This field is written by the uDMA. It means that the frame has a VLAN tag. This field is valid only if the L field is set.

Table continues on the next page...

Table 22-35. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 8	1 IPV6	IPV6 Frame. This field is written by the uDMA. This field indicates that the frame has an IPv6 frame type. If this field is not set it means that an IPv4 or other protocol frame was received. This field is valid only if the L field is set.
Offset + 8	0 FRAG	IPv4 Fragment.This is an accelerator option.This field is written by the uDMA. It indicates that the frame is an IPv4 fragment frame. This field is only valid when the L field is set.
Offset + A	15 ME	MAC error. This field is written by the uDMA. This field means that the frame stored in the system memory was received with an error (typically, a receive FIFO overflow). This field is only valid when the L field is set.
Offset + A	14–11	Reserved, must be cleared.
Offset + A	10 PE	PHY Error. This field is written by the uDMA. Set to "1"when the frame was received with an Error character on the PHY interface. The frame is invalid. This field is valid only when the L field is set.
Offset + A	9 CE	Collision. This field is written by the uDMA. Set when the frame was received with a collision detected during reception. The frame is invalid and sent to the user application. This field is valid only when the L field is set.
Offset + A	8 UC	Unicast. This field is written by the uDMA, and means that the frame is unicast. This field is valid regardless of whether the L field is set.
Offset + A	7 INT	Generate RXB/RXF interrupt. This field is set by the user to indicate that the uDMA is to generate an interrupt on the *dma_int_rxb* / *dma_int_rxfevent*.
Offset + A	6–0	Reserved, must be cleared.
Offset + C	15–0 Payload checksum	Internet payload checksum. This is an accelerator option. It is the one's complement sum of the payload section of the IP frame. The sum is calculated over all data following the IP header until the end of the IP payload. This field is valid only when the L field is set.
Offset + E	15–11 Header length	Header length. This is an accelerator option. This field is written by the uDMA. This field is the sum of 32-bit words found within the IP and its following protocol headers. If an IP datagram with an unknown protocol is found, then the value is the length of the IP header. If no IP frame or an erroneous IP header is found, the value is 0. The following values are minimum values if no header options exist in the respective headers: • ICMP/IP: 6 (5 IP header, 1 ICMP header) • UDP/IP: 7 (5 IP header, 2 UDP header) • TCP/IP: 10 (5 IP header, 5 TCP header) This field is only valid if the L field is set.
Offset + E	10–8	Reserved, must be cleared.
Offset + E	7–0 Protocol type	Protocol type. This is an accelerator option. The 8-bit protocol field found within the IP header of the frame. It is valid only when ICE is cleared. This field is valid only when the L field is set.
Offset + 10	15–0	Reserved, must be cleared.
Offset + 12	15 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 12	14–0	Reserved, must be cleared.
Offset + 14	15–0	This value is written by the uDMA. It is only valid if the L field is set.
Offset + 16		

Table continues on the next page...

Table 22-35. Receive buffer descriptor field definitions (continued)

Word	Field	Description
	1588 timestamp	
Offset + 18 – Offset + 1E	15–0	Reserved, must be cleared.

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 64. The buffer must reside in memory external to the MAC. The Ethernet controller never modifies this value.

22.6.14.2 Enhanced transmit buffer descriptor

Table 22-36. Enhanced transmit buffer descriptor (TxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data length															
Offset + 2	R	TO1	W	TO2	L	TC	—	—	—	—	—	—	—	—	—	—
Offset + 4	Tx Data Buffer Pointer – low halfword															
Offset + 6	Tx Data Buffer Pointer – high halfword															
Offset + 8	TXE	—	UE	EE	FE	LCE	OE	TSE	—	—	—	—	—	—	—	—
Offset + A	—	INT	TS	PINS	IINS	—	—	—	—	—	—	—	—	—	—	—
Offset + C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – low halfword															
Offset + 16	1588 timestamp – high halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 22-37. Enhanced transmit buffer descriptor field definitions

Word	Field	Description
Offset + 0	15–0 Data Length	Data length, written by user. Data length is the number of octets the MAC should transmit from this BD's data buffer. It is never modified by the MAC.
Offset + 2	15 R	Ready. Written by the MAC and you.

Table continues on the next page...

Table 22-37. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
		0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The MAC clears this field after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this field is set.
Offset + 2	14 TO1	Transmit software ownership. This field is reserved for software use. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 2	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 2	12 TO2	Transmit software ownership. This field is reserved for use by software. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 2	11 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 2	10 TC	Transmit CRC. Written by user, and valid only when L is set. 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte This field is valid only when the L field is set.
Offset + 2	9 ABC	Append bad CRC. **Note:** This field is not supported by the uDMA and is ignored.
Offset + 2	8–0	Reserved, must be cleared.
Offset + 4	15–0 Data buffer pointer low	Tx data buffer pointer, low halfword
Offset + 6	15–0 Data buffer pointer high	Tx data buffer pointer, high halfword. The buffer must reside in memory external to the MAC. This value is never modified by the Ethernet controller. **NOTE:** For optimal performance, make the transmit buffer pointer evenly divisible by 64.
Offset + 8	15 TXE	Transmit error occurred. This field is written by the uDMA. This field indicates that there was a transmit error of some sort reported with the frame. Effectively this field is an OR of the other error fields including UE, EE, FE, LCE, OE, and TSE. This field is valid only when the L field is set.
Offset + 8	14	Reserved, must be cleared.
Offset + 8	13 UE	Underflow error. This field is written by the uDMA. This field indicates that the MAC reported an underflow error on transmit. This field is valid only when the L field is set.
Offset + 8	12 EE	Excess Collision error. This field is written by the uDMA. This field indicates that the MAC reported an excess collision error on transmit. This field is valid only when the L field is set.

Table continues on the next page...

Table 22-37. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 8	11 FE	Frame with error. This field is written by the uDMA. This field indicates that the MAC reported that the uDMA reported an error when providing the packet. This field is valid only when the L field is set.
Offset + 8	10 LCE	Late collision error. This field is written by the uDMA. This field indicates that the MAC reported that there was a Late Collision on transmit. This field is valid only when the L field is set.
Offset + 8	9 OE	Overflow error. This field is written by the uDMA. This field indicates that the MAC reported that there was a FIFO overflow condition on transmit. This field is only valid when the L field is set.
Offset + 8	8 TSE	Timestamp error. This field is written by the uDMA. This field indicates that the MAC reported a different frame type then a timestamp frame. This field is valid only when the L field is set.
Offset + 8	7–0	Reserved, must be cleared.
Offset + A	15	Reserved, must be cleared.
Offset + A	14 INT	Generate interrupt flags. This field is written by the user. This field is valid regardless of the L field and must be the same for all EBD for a given frame. The uDMA does not update this value.
Offset + A	13 TS	Timestamp. This field is written by the user. This indicates that the uDMA is to generate a timestamp frame to the MAC. This field is valid regardless of the L field and must be the same for all EBD for the given frame. The uDMA does not update this value.
Offset + A	12 PINS	Insert protocol specific checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the protocol checksum and overwrites the corresponding checksum field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + A	11 IINS	Insert IP header checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the IP header checksum and overwrites the corresponding header field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + A	10–0	Reserved, must be cleared.
Offset + C	15–0	Reserved, must be cleared.
Offset + E	15–0	Reserved, must be cleared.
Offset + 10	15–0	Reserved, must be cleared.
Offset + 12	15 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 12	14–0	Reserved, must be cleared.
Offset + 14	15–0 1588 timestamp	This value is written by the uDMA . It is valid only when the L field is set.
Offset + 16		
Offset + 18–Offset + 1E	15–0	Reserved, must be cleared.

22.6.15 Client FIFO application interface

The FIFO interface is completely asynchronous from the Ethernet line, and the transmit and receive interface can operate at a different clock rate.

All transfers to/from the user application are handled independently of the core operation, and the core provides a simple interface to user applications based on a two-signal handshake.

22.6.15.1 Data structure description

The data structure defined in the following tables for the FIFO interface must be respected to ensure proper data transmission on the Ethernet line. Byte 0 is sent to and received from the line first.

Table 22-38. FIFO interface data structure

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Word 0	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Word 1	Byte 15	Byte 14	Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8
...	...							

The size of a frame on the FIFO interface may not be a modulo of 64-bit.

The user application may not care about the Ethernet frame formats in full detail. It needs to provide and receive an Ethernet frame with the following structure:

- Ethernet MAC destination address
- Ethernet MAC source address
- Optional 802.1q VLAN tag (VLAN type and info field)
- Ethernet length/type field
- Payload

Frames on the FIFO interface do not contain preamble and SFD fields, which are inserted and discarded by the MAC on transmit and receive, respectively.

- On receive, CRC and frame padding can be stripped or passed through transparently.
- On transmit, padding and CRC can be provided by the user application, or appended automatically by the MAC independently for each frame. No size restrictions apply.

Note

On transmit, if ENET*n*_TCR[ADDINS] is set, bytes 6–11 of each frame can be set to any value, since the MAC overwrites the bytes with the MAC address programmed in the ENET*n*_PAUR and ENET*n*_PALR registers.

Table 22-39. FIFO interface frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–13	Length/type field
14–N	Payload data

VLAN-tagged frames are supported on both transmit and receive, and implement additional information (VLAN type and info).

Table 22-40. FIFO interface VLAN frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–15	VLAN tag and info
16–17	Length/type field
18–N	Payload data

Note

The standard defines that the LSB of the MAC address is sent/received first, while for all the other header fields — in other words, length/type, VLAN tag, VLAN info, and pause quanta — the MSB is sent/received first.

22.6.15.2 Data structure examples

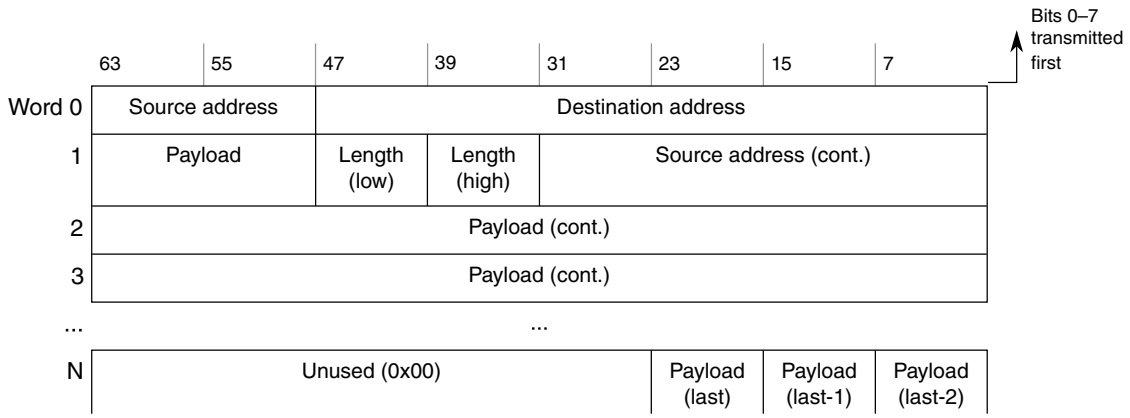


Figure 22-14. Normal Ethernet frame 64-bit mapping example

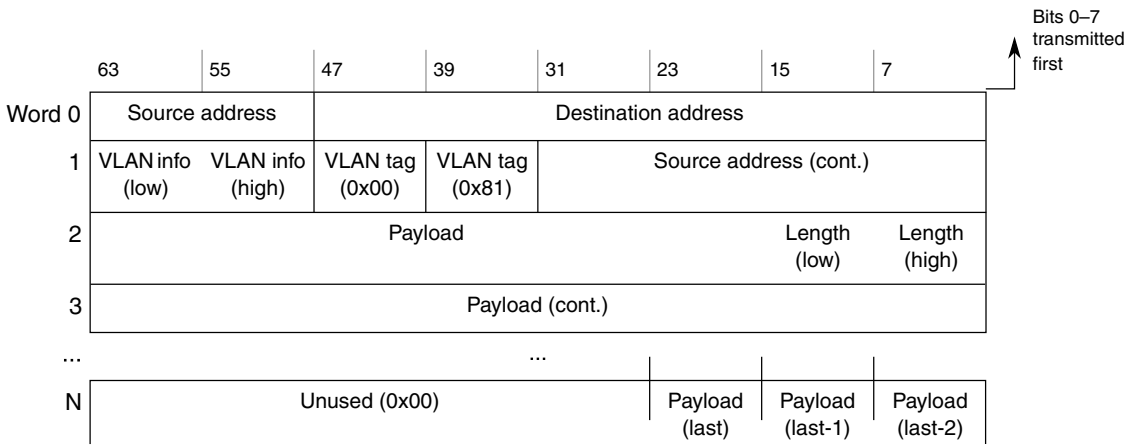


Figure 22-15. VLAN-tagged frame 64-bit mapping example

If CRC forwarding is enabled (CRCFWD = 0), the last four valid octets of the frame contain the FCS field. The non-significant bytes of the last word can have any value.

22.6.15.3 Frame status

A MAC layer status word and an accelerator status word is available in the receive buffer descriptor.

See Enhanced buffer descriptors for details.

The status is available with each frame with the last data of the frame.

If the frame status contains a MAC layer error (for example, CRC or length error), RxBD[ME] is also set with the last data of the frame.

22.6.16 FIFO protection

The following sections describe the FIFO protection mechanisms.

22.6.16.1 Transmit FIFO underflow

During a frame transfer, when the transmit FIFO reaches the almost empty threshold with no end-of-frame indication stored in the FIFO, the MAC logic:

- Stops reading data from the FIFO
- Asserts the MII error signal (MII_TXER) (1 in Figure 22-16) to indicate that the fragment already transferred is not valid
- Deasserts the MII transmit enable signal (MII_TXEN) to terminate the frame transfer (2)

After an underflow, when the application completes the frame transfer (3), the MAC transmit logic discards any new data available in the FIFO until the end of packet is reached (4) and sets the enhanced TxBD[UE] field.

The MAC starts to transfer data on the MII interface when the application sends a new frame with a start of frame indication (5).

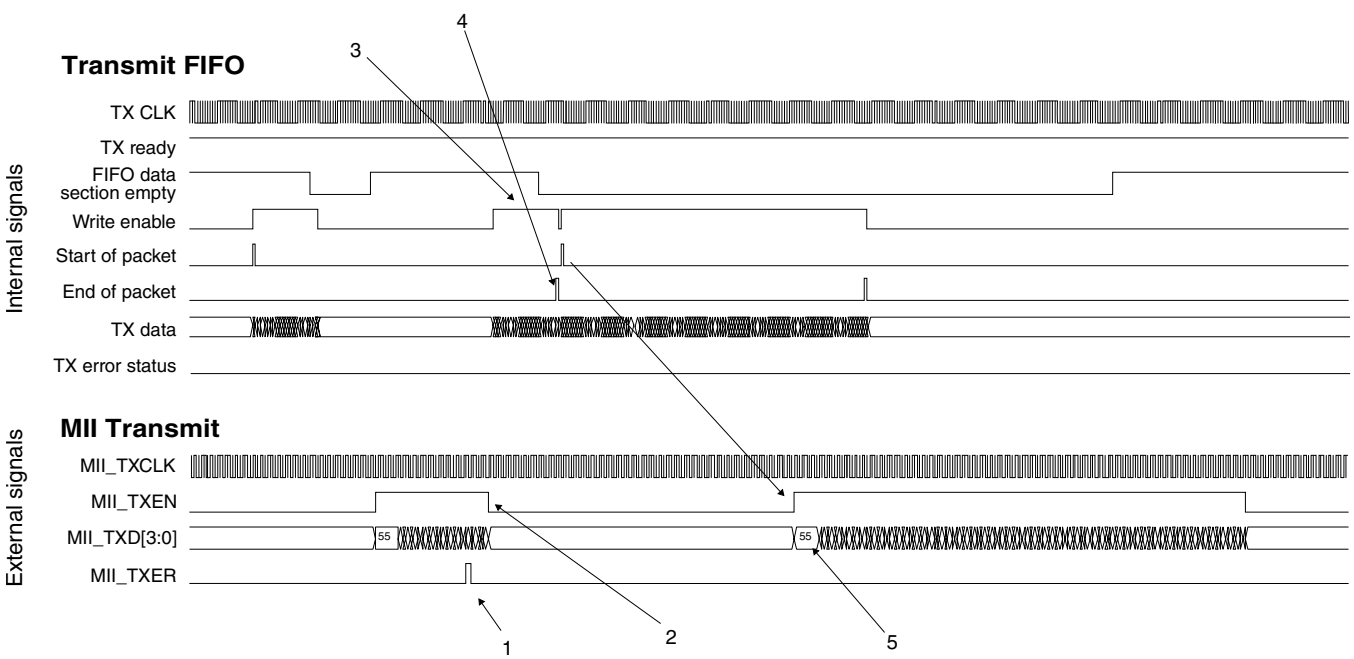


Figure 22-16. Transmit FIFO underflow protection

22.6.16.2 Transmit FIFO overflow

On the transmit path, when the FIFO reaches the programmable almost full threshold, the internal MAC ready signal is deasserted. The application should stop sending new data.

However, if the application keeps sending data, the transmit FIFO overflows, corrupting contents that were previously stored. The core logic sets the enhanced TxBD[OE] field for the next frame transmitted to indicate this overflow occurence.

Note

Overflow is a fatal error and must be addressed by resetting the core or clearing ENET*n*_ECR[ETHER_EN], to clear the FIFOs and prepare for normal operation again.

22.6.16.3 Receive FIFO overflow

During a frame reception, if the client application is not able to receive data (1), the MAC receive control truncates the incoming frame when the FIFO reaches the programmable almost-full threshold to avoid an overflow.

The frame is subsequently received on the FIFO interface with an error indication (enhanced RxBD[ME] field set together with receive end-of-packet) (2) with the truncation error status field set (3).

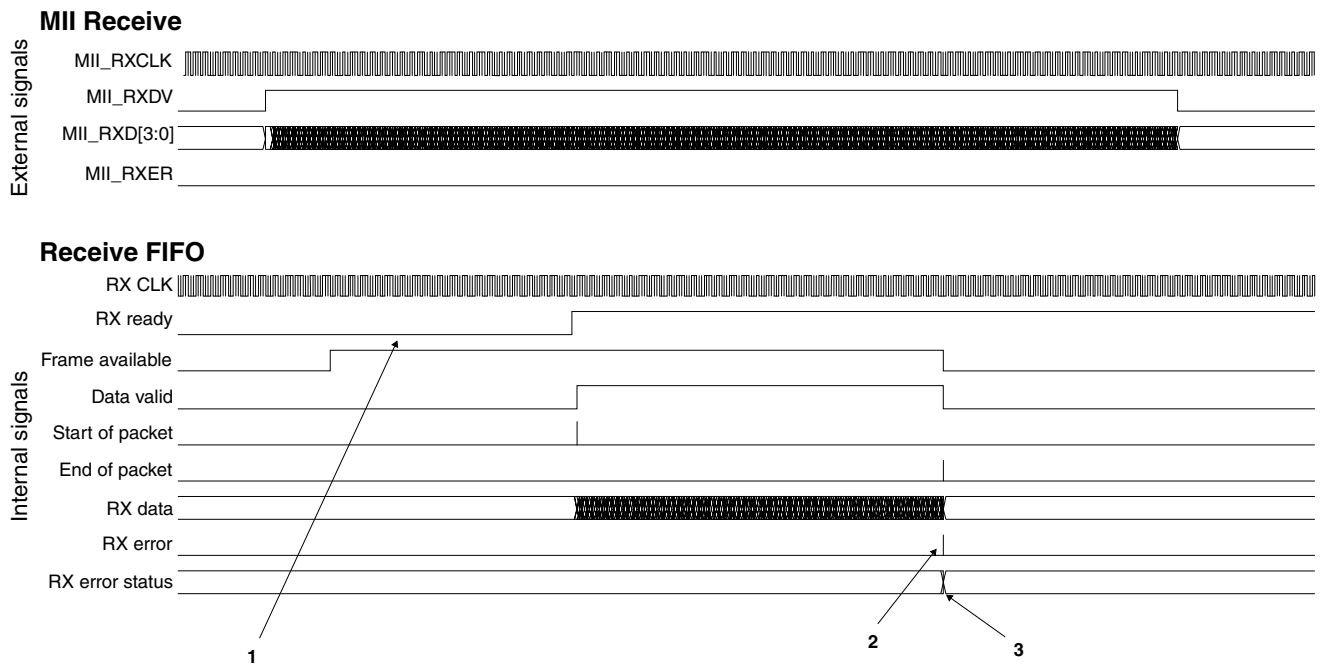


Figure 22-17. Receive FIFO overflow protection

22.6.17 PHY management interface

The MDIO interface is a two-wire management interface. The MDIO management interface implements a standardized method to access the PHY device management registers.

The core implements a master MDIO interface, which can be connected to up to 32 PHY devices.

22.6.17.1 MDIO clause 22 frame format

The core MDIO master controller communicates with the slave (PHY device) using frames that are defined in the following table.

A complete frame has a length of 64 bits made up of an optional 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC signal). The MDIO data signal is tri-stated between frames.

The core PHY management interface supports the standard MDIO specification (IEEE 802.3 Clause 22).

Table 22-41. MDIO clause 22 frame structure

ST	OP	PHYADR	REGADR	TA	DATA

Table 22-42. MDIO frame field descriptions

Field	Description
ST (2 bits)	Start indication field, programmed with ENET*n*_MMFR[ST] and equal to 01 for Standard MDIO (Clause 22).
OP (2 bits)	Opcode defines type of operation. Programmed with ENET*n*_MMFR[OP]. 01 Write operation 10 Read operation
PHYADR (5 bits)	Five-bit PHY device address, programmed with ENET*n*_MMFR[PA]. Up to 32 devices can be addressed.
REGADR (5 bits)	Five-bit register address, programmed with ENET*n*_MMFR[RA]. Each PHY can implement up to 32 registers.
TA (2 bits)	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
Data	Data, set by ENET*n*_MMFR[DATA]. Written to or read from the PHY

Table 22-42. MDIO frame field descriptions

Field	Description
(16 bits)	

22.6.17.2 MDIO clause 45 frame format

The extended MDIO frame structure defined in IEEE 802.3 Clause 45 introduces indirect addressing. First, a write transaction to an address register is done, followed by a write or read transaction which will put the 16-bit data in the register or retrieve the register contents respectively. A preamble of 32 bits of logical ones is sent prior to every transaction. The MDIO data signal is tri-stated between frames.

The extended MDIO defines four transactions, which are determined by the two-bit opcode field.

Table 22-43. MDIO clause 45 frame structure

ST	OP	PRTAD	DEVAD	TA	ADDR/DATA

All bits are transmitted from left to right (Preamble bits first) and all fields have their Most-Significant bit sent first (leftmost in above table). The complete frame has a length of 64 bits (32-bit preamble, 14-bit command, 2-bit bus direction change, 16-bit data). Each bit is transferred with the rising edge of the MDIO clock (MDC).

The fields and transactions are summarized in the following tables.

Table 22-44. MDIO clause 45 frame field descriptions

Field	Description
ST	Start indication. Indicates the end of the preamble and start of the frame. This value is 00 for extended MDIO (Clause 45) frames.
OP	Opcode defines if a read or write operation is performed and is programmed with ENET*n*_MMFR[OP]. See Table 22-45 for more information. 00 Address write 01 Write operation 10 Read inc. operation 11 Read operation
PRTAD	The port address specifies a MDIO port. Each Port can have up to 32 devices which each can have a separate set of registers.
DEVAD	Device address. Up to 32 devices can be addressed (within a port).
TA	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.

Table continues on the next page...

Table 22-44. MDIO clause 45 frame field descriptions (continued)

Field	Description
ADDR/DATA	16-bit address (for address write) or data, set by ENET*n*_MMFR[DATA], written to or read from the PHY.

Table 22-45. MDIO Clause 45 Transactions

Transaction Type	Description
Address	A write transaction to the internal address register of the device/port. The data section of the frame contains the value to be stored in the device's internal address "pointer" register for further transactions.
Write	Data write to a register. The 16 bit data will be written to the register identified by the device-internal address.
Read	Data is read from the register identified by the device-internal address.
Read inc.	Read with address postincrement. The register identified by the device-internal address is read. After this, the device-internal address is incremented. If the address register is all '1' (0xFFFF) no increment is done (i.e. increment does not wrap around).

22.6.17.3 MDIO clock generation

The MDC clock is generated from the internal bus clock (i.e., IPS bus clock) divided by the value programmed in ENET*n*_MSCR[MII_SPEED].

22.6.17.4 MDIO operation

To perform an MDIO access, set the MDIO command register (ENET*n*_MMFR) according to the description provided in MII Management Frame Register (ENETn_MMFR).

To check when the programmed access completes, read the ENET*n*_EIR[MII] field.

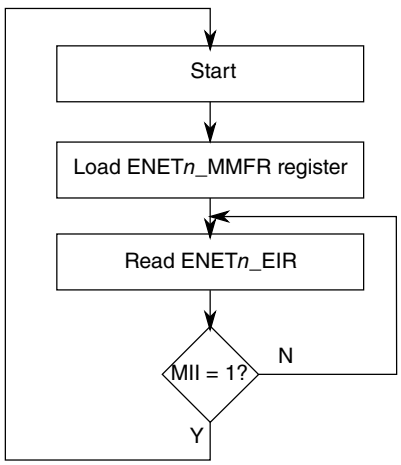


Figure 22-18. MDIO access overview

22.6.18 Ethernet interfaces

The following Ethernet interfaces are implemented:

- Fast Ethernet MII (Media Independent Interface)
- RMII 10/100 using interface converters/gaskets

The following table shows how to configure ENET registers to select each interface.

Mode	ECR[SPEED]	RCR[RMII_10T]	RCR[RMII_MODE]
MII - 10 Mbit/s	0	—	0
MII - 100 Mbit/s	0	—	0
RMII - 10 Mbit/s	0	1	1
RMII - 100 Mbit/s	0	0	1

22.6.18.1 RMII interface

In RMII receive mode, for normal reception following assertion of CRS_DV, RXD[1:0] is 00 until the receiver determines that the receive event has a proper start-of-stream delimiter (SSD).

The preamble appears (RXD[1:0]=01) and the MACs begin capturing data following detection of SFD.

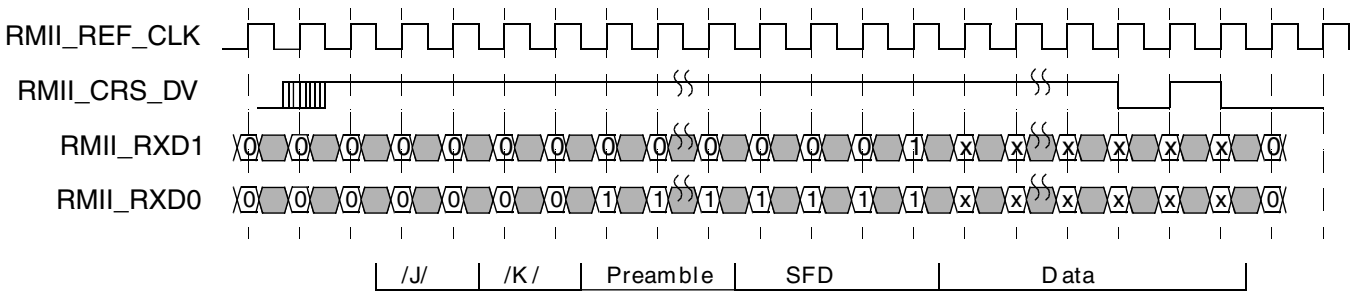


Figure 22-19. RMII receive operation

If a false carrier is detected (bad SSD), then RXD[1:0] is 10 until the end of the receive event. This is a unique pattern since a false carrier can only occur at the beginning of a packet where the preamble is decoded (RXD[1:0] = 01).

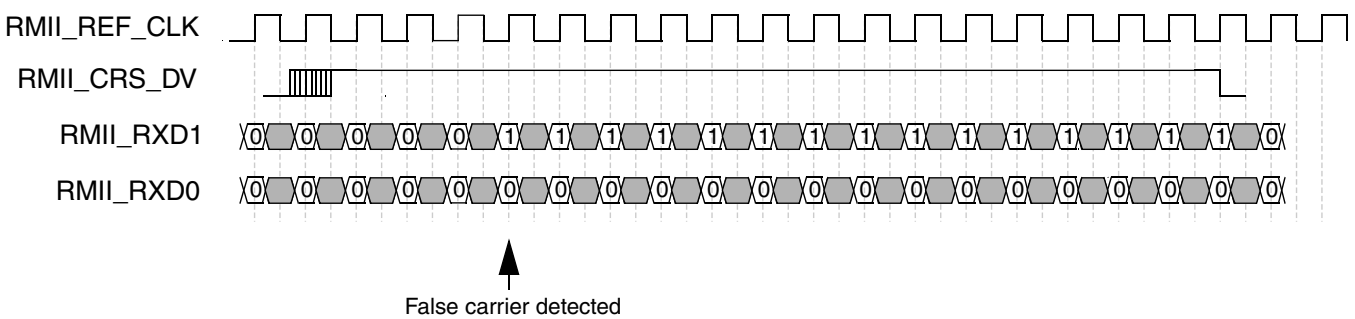


Figure 22-20. RMII receive operation with false carrier

In RMII transmit mode, TXD[1:0] provides valid data for each REF_CLK period while TXEN is asserted.

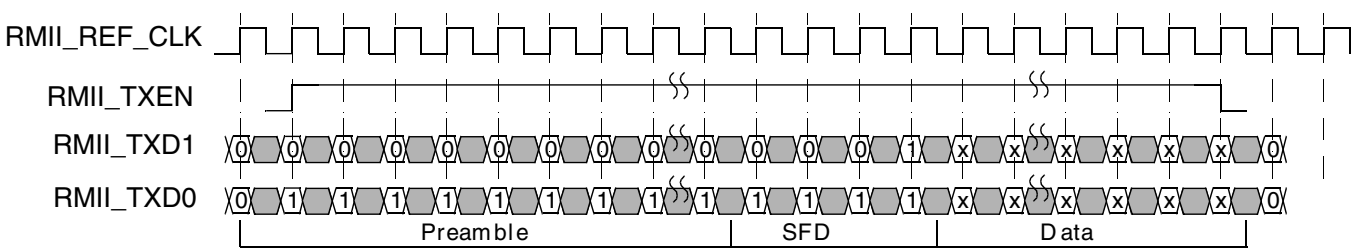


Figure 22-21. RMII transmit operation

22.6.18.2 MII Interface — transmit

On transmit, all data transfers are synchronous to MII_TXCLK rising edge. The MII data enable signal MII_TXEN is asserted to indicate the start of a new frame, and remains asserted until the last byte of the frame is present on the MII_TXD[3:0] bus.

Between frames, MII_TXEN remains deasserted.

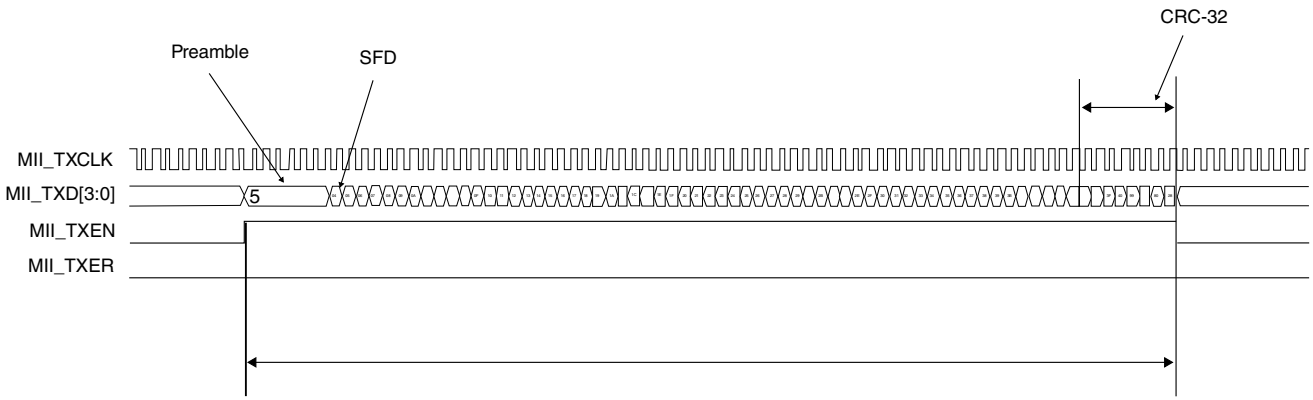


Figure 22-22. MII transmit operation

If a frame is received on the FIFO interface with an error (for example, RxBD[ME] set) the frame is subsequently transmitted with the MII_TXER error signal for one clock cycle at any time during the packet transfer.

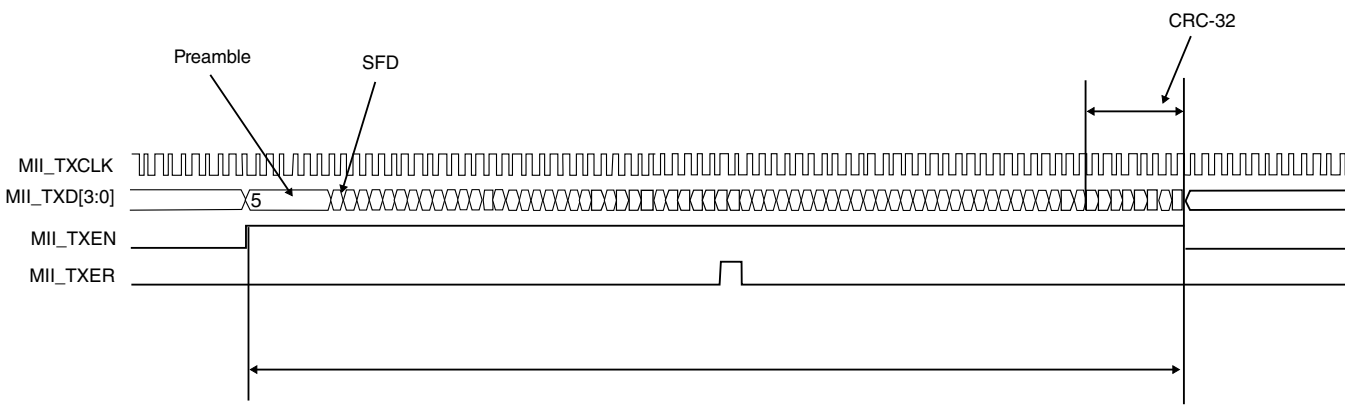


Figure 22-23. MII transmit operation — errored frame

22.6.18.2.1 Transmit with collision — half-duplex

When a collision is detected during a frame transmission (MII_COL asserted), the MAC stops the current transmission, sends a 32-bit jam pattern, and re-transmits the current frame.

(See Collision detection in half-duplex mode for details)

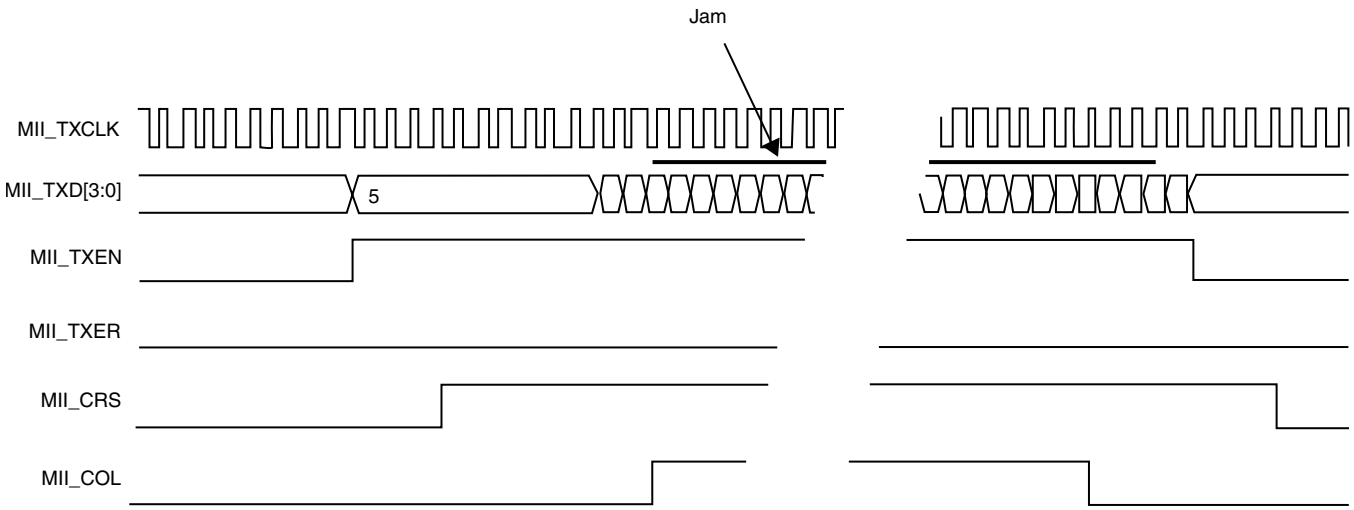


Figure 22-24. MII transmit operation — transmission with collision

22.6.18.3 MII interface — receive

On receive, all signals are sampled on the MII_RXCLK rising edge. The MII data enable signal, MII_RXDV, is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on MII_RXD[3:0] bus.

Between frames, MII_RXDV remains deasserted.

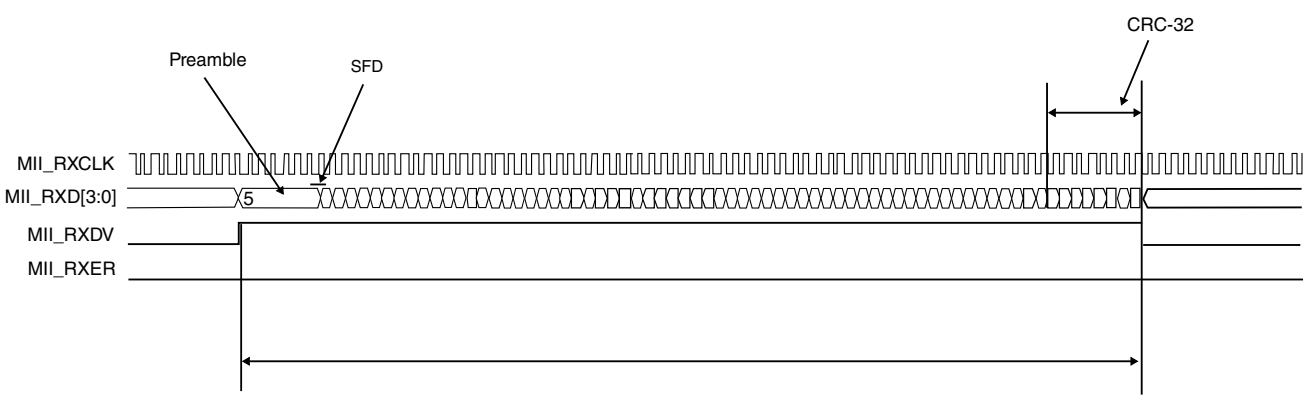


Figure 22-25. MII receive operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, MII_RXER, for at least one clock cycle at any time during the packet transfer.

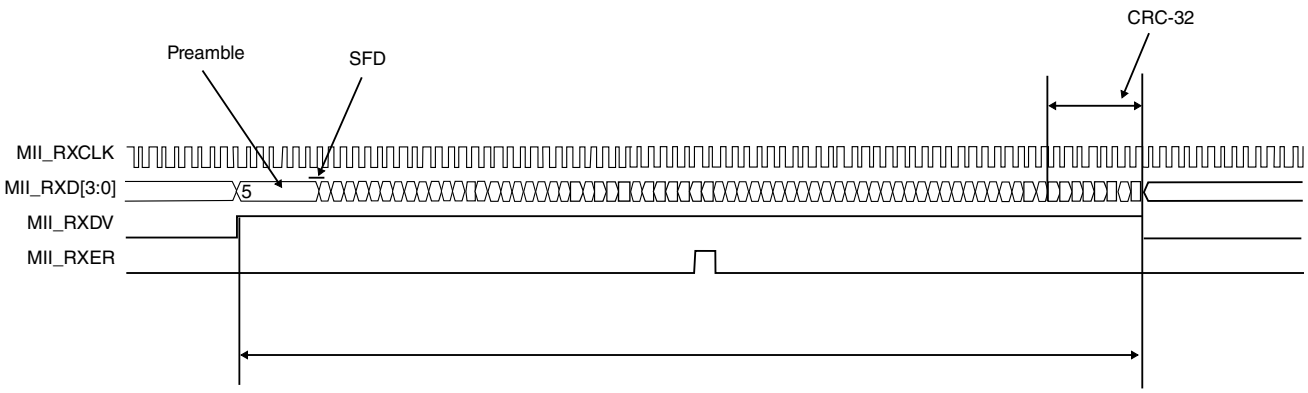


Figure 22-26. MII receive operation — errored frame

A frame received on the MII interface with a PHY error indication is subsequently transferred on the FIFO interface with RxBD[ME] set.

22.6.19 Interrupt coalescence

The purpose of the interrupt coalescing is to reduce the number of interrupts generated by the MAC so as to reduce the CPU loading.

To facilitate this interrupt coalescing, these registers are available with the same control and configuration fields.

- Transmit Interrupt Coalescing Register (ENET_TXIC)
- Receive Interrupt Coalescing Register (ENET_RXIC)

When coalescing is enabled by asserting the corresponding ICEN field and such interrupt is also enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt when the threshold number of frames is reached (defined by ICFT) or when the threshold timer expires (defined by ICTT).

When coalescing is disabled by de-asserting ICEN, but interrupt is enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt as they are received without using coalescing. Interrupt coalescing is done for each transmit and receive queue/class independently.

22.6.19.1 Interrupt coalescence setup

Interrupt coalescence supports both legacy and enhanced BDs. The following guidelines are recommended when setting up interrupt coalescence.

- When the MAC is configured for enhanced (IEEE 1588) mode, that is, enhanced BDs:
  - Set the INT bit in the enhanced received buffer descriptor to one.
  - Set the INT bit in the enhanced transmit buffer descriptor(s) to one.
- Clear the TXB and RXB fields in the EIMR register.

22.6.19.2 Updating the frame count threshold on-the-fly

To update the ICFT field in the RXIC and TXIC registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new threshold value to the ICFT field.
3. Set ICEN to one.

NOTE

The ICFT field can be updated on-the-fly without disabling the ICEN field. The hardware interrupt will continue and there is a possibility that an interrupt will occur depending on the state of the hardware counter and the previous ICFT value.

22.6.19.3 Updating the timer threshold on-the-fly

To update the ICTT field in the RXIC and TXIC registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new timer value to the ICTT field.
3. Set ICEN to one.

Chapter 23
Enhanced Periodic Interrupt Timer (EPIT)

23.1 Overview

EPIT is a 32-bit set-and-forget timer that is capable of providing precise interrupts at regular intervals with minimal processor intervention. EPIT begins counting after it is enabled by software.

The following figure shows the EPIT block diagram.

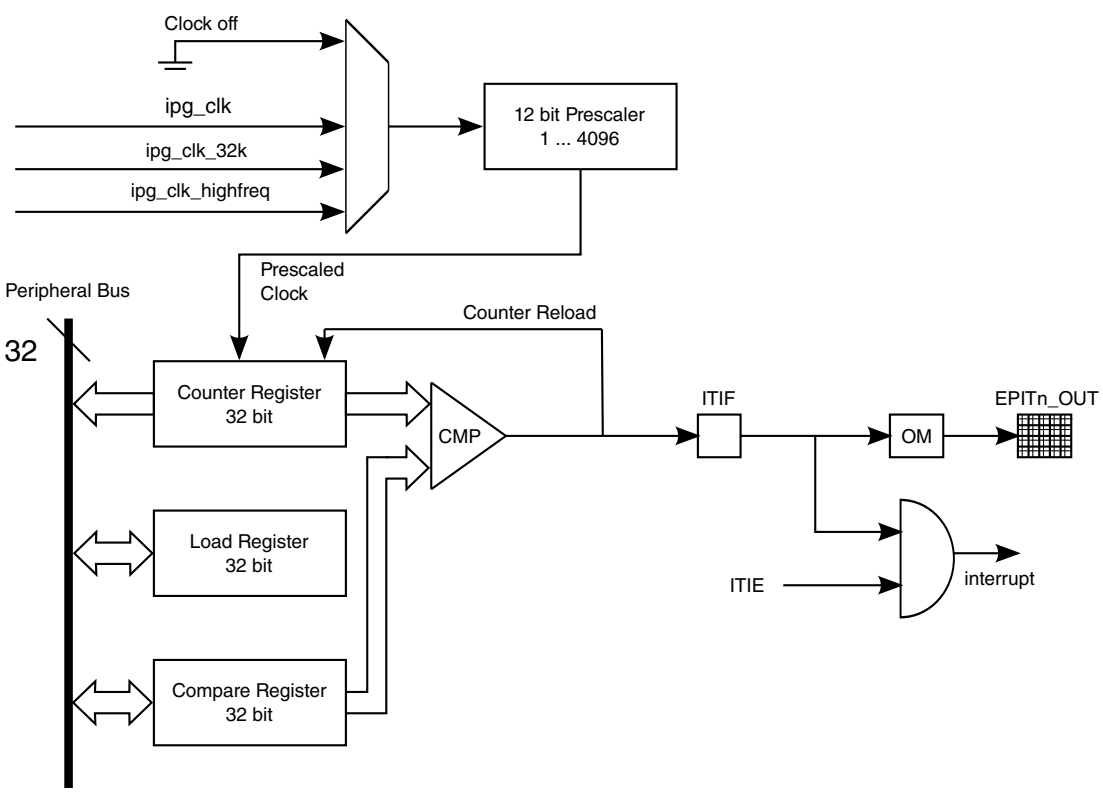


Figure 23-1. EPIT block diagram

23.1.1 EPIT features

EPIT has the following key features:

- 32-bit down counter with clock source selection
- 12-bit prescaler for division of input clock frequency
- Counter value that can be programmed on the fly
- Can be programmed to be active during low-power and debug modes
- Interrupt generation when counter reaches the compare value

23.1.2 EPIT modes and operations

EPIT supports the following modes: set-and-forget and free running. See the following sections for more information.

- Operating in set-and-forget mode
- Operating in free-running mode

See Operations for a description of the operations that EPIT supports.

23.2 External signals

The following table describes EPIT's I/O signals.

Table 23-1. EPIT External Signals

Signal	Description	Pad	Mode	Direction
EPIT1_OUT	Indicate the occurrence of EPIT1 output compare event through a specified transition.	UART3_RX_DATA	ALT8	O
		JTAG_TMS	ALT8	O
EPIT2_OUT	Indicate the occurrence of EPIT2 output compare event through a specified transition.	UART3_CTS_B	ALT8	O
		JTAG_TDO	ALT8	O

23.3 Clocks

The table found here describes the clock sources for EPIT.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 23-2. EPIT Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32 kHz)
ipg_clk_highfreq	perclk_clk_root	High-frequency reference clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

The clock that feeds the prescaler can be selected from among the following sources:

- **High-frequency reference clock (ipg_clk_highfreq)**

  This clock is provided by the Clock Control Module (CCM). This clock remains on during low-power mode when the peripheral clock is turned off, allowing EPIT to use this clock in low-power mode. In normal mode, the CCM synchronizes this clock to ahb_clk; in low-power mode, CCM switches to an unsynchronized version.

- **Low-frequency reference clock (ipg_clk_32k)**

  This 32 kHz reference clock is provided by the CCM. This clock remains on in low-power mode when the peripheral clock is turned off, so EPIT can use this clock during low-power mode. In normal mode, the CCM synchronizes this clock to ahb_clk; in low-power mode, CCM switches to an unsynchronized version. This clock is derived from the external 32 kHz crystal.

- **Peripheral clock (ipg_clk)**

  This is the peripheral clock (PER Clock) which is provided (and optionally gated) by the CCM. This clock is typically used in normal operations. In low-power modes, if the EPIT is programmed to be disabled (via STOPEN or WAITEN), then the peripheral clock can be switched off.

The clock input source is determined by the CLKSRC field in the control register. The clock input to the prescaler can also be disabled by setting CLKSRC to 0b00. **This field value should only be changed after first disabling the EPIT by clearing the EN bit in the EPIT_EPITCR**. For other programming requirements that apply while changing clock source, refer section Change of Clock Source.

The PRESCALER field in the control register is used to select the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value between 1 and 4096. A change in the value of the PRESCALER field is immediately reflected on its output clock frequency. The following figure shows the timing for a change in the prescaler value.

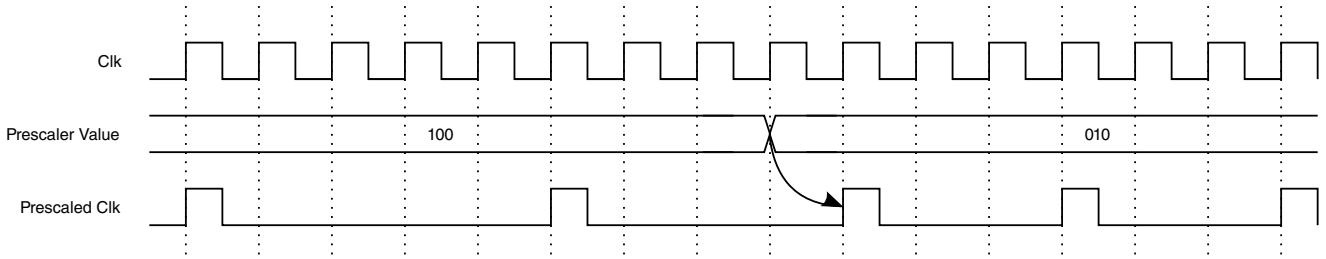


Figure 23-2. Prescaler Value Change Diagram

23.4 Functional Description

This section provides a complete functional description of the block.

23.4.1 Operating modes

EPIT can operate in either set-and-forget or free-running mode. Use EPIT_CR[RLD] to select the desired mode.

23.4.1.1 Operating in set-and-forget mode

To select this mode of operation, set the RLD bit in the control register (EPIT_CR).

In this mode, the counter obtains its data from the load register (EPIT_LR); it cannot be written to directly from the block data bus. Whenever the counter reaches zero, the value in EPIT_LR is loaded into the counter. This value is then decremented to zero.

To directly initialize the counter instead of waiting for the count to reach zero, set the EPIT counter overwrite enable bit (EPIT_CR[IOVW]) and write to EPIT_LR with the required initialization value.

23.4.1.2 Operating in free-running mode

To select this mode of operation, clear the RLD bit.

In this mode, the counter rolls over from 0000 0000h to FFFF FFFFh without reloading from the modulus register. After rolling over, the counter continues counting down.

To directly initalize the counter, set the EPIT counter overwrite enable bit (EPIT_EPITCL[IOVW]) and write to EPIT_EPITLR with the required initialization value.

23.4.2 Operations

EPIT has a single 32-bit down counter, which starts counting when the block is enabled by software.

The start value of the counter is loaded from the EPIT load register, which can be written to at any time by the processor. The value in the compare register determines the time that the interrupt occurs.

When EPIT is disabled (EN = 0), both the main counter and the prescaler counter freeze their count at their current count values. When EPIT is re-enabled (EN = 1), the ENMOD bit, which is a RW bit, decides the counter value:

- If ENMOD is set, the main counter is loaded with the load value (If RLD = 1)/ FFFF FFFFh (If RLD = 0) and the prescaler counter is reset (000h).
- If ENMOD is cleared, both main counter and prescaler counter restart counting from their frozen values.

If EPIT is programmed to be disabled in a low-power mode (STOP/WAIT), both the main counter and the prescaler counter freeze at their current count values when EPIT enters low-power mode. When EPIT exits the low-power mode, both the main counter and the prescaler counter start counting from their frozen values regardless of the ENMOD bit.

A hardware reset resets all EPIT registers to their respective reset values. There is a software reset which has the same effect on all registers except for the EN, ENMOD, STOPEN and WAITEN bits in the control register. The state of these bits are not affected by software reset. A software reset can be asserted even when the EPIT is disabled.

23.4.3 Compare Event

When the programmed value of EPIT_EPITCMPR matches the value in EPIT_EPITCNR a compare status flag is set, and an interrupt is generated if the OCIEN bit is set in the control register.

The compare output pin is set, cleared, toggled, or not affected at all depending on the setting of the output mode (OM) bits in the control register. If an interrupt is required at rollover (when the counter value reaches 0x0000_0000 and the new value is loaded) then the compare register value should be set equal to the load register value in set-and-forget mode, or equal to 0xFFFF_FFFF in free-running mode.

The following figure shows the timing for a compare event and interrupt.

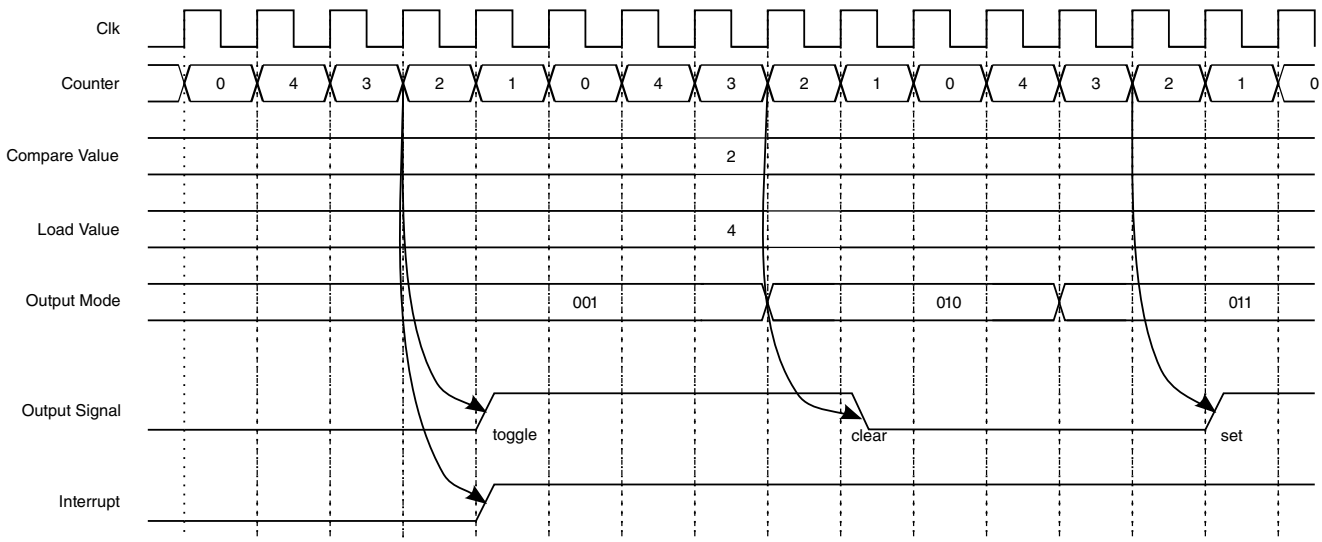


Figure 23-3. Compare Event and Interrupt Timing Diagram

EPIT will generate a compare event in the next count if the EPITx_CNR from the previous count equals the new EPITx_CMPR configured before re-enabling the EPIT in the next count. Even in case a new start counter value was updated in EPITx_LR before re-enabling the EPIT for the next round. To avoid this, configure the EPITx_CMPR to previous EPITx_CNR+1. Or, in set and forget mode, configure EPITx_LR with IOVW=1, before disabling EPIT. Also can do an extra disable/enable iteration to clear OCIF and update EPITx_CNR.

23.4.3.1 Counter Value Overwrite

The EPIT counter value can be overwritten to acquire a desired value at any point of time. The procedure for this is to set the IOVW bit in the control register and then write the desired value into the load register.

This results in the load register acquiring that value and also the counter being overwritten with it. If the EPIT is running the counter resumes counting from the overwritten value.

23.4.3.2 Low-Power Mode Behavior

The EPIT timer's behavior in low-power modes depends on which clock source is being used.

If the selected clock source is available and the corresponding low-power enable bit is set, then the EPIT continues to function in the low-power mode. If the EPIT is programmed to be disabled in a low-power mode (STOP/WAIT), then main counter and the prescaler counter freeze at the current count values when the EPIT enters low-power mode. When the EPIT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit.

23.4.3.3 Debug Mode Behavior

In debug mode, the user has the option to run or halt the EPIT timers. If the DBGEN bit is reset in the EPIT Control Register, the timer is halted.

When debug mode is exited, the timer operation reverts to what it was prior to entering debug mode.

23.5 Initialization/ Application Information

23.5.1 Change of Clock Source

The CLKSRC field in EPIT_EPITCR determines the clock source. This field value should be changed only after disabling the EPIT (EN = 0).

Below is the software sequence which must be followed while changing clock source.

1. Disable the EPIT - set EN=0 in EPIT_EPITCR.
2. Disable EPIT ouput - program OM=00 in the EPIT_EPITCR.
3. Disable EPIT interrupts.
4. Program CLKSRC to desired clock source in EPIT_EPITCR.
5. Clear the EPIT status register (EPIT_EPITSR), that is, write "1" to clear (w1c).
6. Set ENMOD= 1 in the EPIT_EPITCR, to bring the EPIT Counter to defined state (EPIT_EPITLR value or 0xFFFF_FFFF).
7. Enable EPIT - set (EN=1) in the EPIT_EPITCR
8. Enable the EPIT interrupts.

23.6 EPIT Memory Map/Register Definition

The EPIT includes five user-accessible 32-bit registers. The following table summarizes these registers and their addresses.

Peripheral bus write access to the EPIT control register (EPITCR) and the EPIT load register (EPITLR) results in one cycle of wait state, while other valid peripheral bus accesses are with 0 wait state.

EPIT memory map

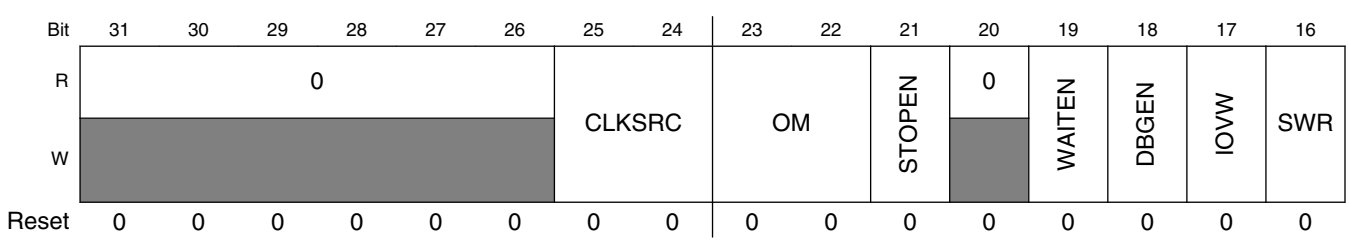
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20D_0000	Control register (EPIT1_CR)	32	R/W	0000_0000h	23.6.1/1020
20D_0004	Status register (EPIT1_SR)	32	R/W	0000_0000h	23.6.2/1023
20D_0008	Load register (EPIT1_LR)	32	R/W	FFFF_FFFFh	23.6.3/1023
20D_000C	Compare register (EPIT1_CMPR)	32	R/W	0000_0000h	23.6.4/1024
20D_0010	Counter register (EPIT1_CNR)	32	R	FFFF_FFFFh	23.6.5/1024
20D_4000	Control register (EPIT2_CR)	32	R/W	0000_0000h	23.6.1/1020
20D_4004	Status register (EPIT2_SR)	32	R/W	0000_0000h	23.6.2/1023
20D_4008	Load register (EPIT2_LR)	32	R/W	FFFF_FFFFh	23.6.3/1023
20D_400C	Compare register (EPIT2_CMPR)	32	R/W	0000_0000h	23.6.4/1024
20D_4010	Counter register (EPIT2_CNR)	32	R	FFFF_FFFFh	23.6.5/1024

23.6.1 Control register (EPITx_CR)

The EPIT control register (EPIT_CR) is used to configure the operating settings of the EPIT. It contains the clock division prescaler value and also the interrupt enable bit. Additionally, it contains other control bits which are described below.

Peripheral Bus Write access to EPIT Control Register (EPIT_CR) results in one cycle of the wait state, while other valid peripheral bus accesses are with 0 wait state.

Address: Base address + 0h offset



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

EPITx_CR field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 CLKSRC	Select clock source These bits determine which clock input is to be selected for running the counter. This field value should only be changed when the EPIT is disabled by clearing the EN bit in this register. For other programming requirements while changing clock source, refer to Change of Clock Source. 00 Clock is off 01 Peripheral clock 10 High-frequency reference clock 11 Low-frequency reference clock
23–22 OM	EPIT output mode.This bit field determines the mode of EPIT output on the output pin. 00 EPIT output is disconnected from pad 01 Toggle output pin 10 Clear output pin 11 Set output pin
21 STOPEN	EPIT stop mode enable. This read/write control bit enables the operation of the EPIT during stop mode. This bit is reset by a hardware reset and unaffected by software reset. 0 EPIT is disabled in stop mode 1 EPIT is enabled in stop mode
20 Reserved	This read-only field is reserved and always has the value 0.
19 WAITEN	This read/write control bit enables the operation of the EPIT during wait mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled in wait mode 1 EPIT is enabled in wait mode
18 DBGEN	This bit is used to keep the EPIT functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode.This bit is reset by hardware reset. A software reset does not affect this bit. 0 Inactive in debug mode 1 Active in debug mode
17 IOVW	EPIT counter overwrite enable. This bit controls the counter data when the modulus register is written. When this bit is set, all writes to the load register overwrites the counter contents and the counter starts subsequently counting down from the programmed value. 0 Write to load register does not result in counter value being overwritten. 1 Write to load register results in immediate overwriting of counter value.

Table continues on the next page...

EPITx_CR field descriptions (continued)

Field	Description
16 SWR	Software reset. The EPIT is reset when this bit is set to 1. It is a self clearing bit. This bit is set when the block is in reset state and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values, except for the EN, ENMOD, STOPEN, WAITEN and DBGEN bits in this control register 0 EPIT is out of reset 1 EPIT is undergoing reset
15–4 PRESCALAR	Counter clock prescaler value. This bit field determines the prescaler value by which the clock is divided before it goes to the counter 0x000 Divide by 1 0x001 Divide by 2... 0xFFF Divide by 4096
3 RLD	Counter reload control. This bit is cleared by hardware reset. It decides the counter functionality, whether to run in free-running mode or set-and-forget mode. 0 When the counter reaches zero it rolls over to 0xFFFF_FFFF (free-running mode) 1 When the counter reaches zero it reloads from the modulus register (set-and-forget mode)
2 OCIEN	Output compare interrupt enable. This bit enables the generation of interrupt on occurrence of compare event. 0 Compare interrupt disabled 1 Compare interrupt enabled
1 ENMOD	EPIT enable mode. When EPIT is disabled (EN=0), both main counter and prescaler counter freeze their count at current count values. ENMOD bit is a r/w bit that determines the counter value when the EPIT is enabled again by setting EN bit. If ENMOD bit is set, then main counter is loaded with the load value (If RLD=1)/ 0xFFFF_FFFF (If RLD=0) and prescaler counter is reset, when EPIT is enabled (EN=1). If ENMOD is programmed to 0 then both main counter and prescaler counter restart counting from their frozen values when EPIT is enabled (EN=1). If EPIT is programmed to be disabled in a low-power mode (STOP/WAIT/ DEBUG), then both the main counter and the prescaler counter freeze at their current count values when EPIT enters low-power mode. When EPIT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 Counter starts counting from the value it had when it was disabled. 1 Counter starts count from load value (RLD=1) or 0xFFFF_FFFF (If RLD=0)
0 EN	This bit enables the EPIT. EPIT counter and prescaler value when EPIT is enabled (EN = 1), is dependent upon ENMOD and RLD bit as described for ENMOD bit. It is recommended that all registers be properly programmed before setting this bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled 1 EPIT is enabled

23.6.2 Status register (EPITx_SR)

The EPIT status register (EPIT_SR) has a single status bit for the output compare event. The bit is a write 1 to clear bit.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															OCIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPITx_SR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 OCIF	Output compare interrupt flag. This bit is the interrupt flag that is set when the content of counter equals the content of the compare register (EPIT_CMPR). The bit is a write 1 to clear bit. 0 Compare event has not occurred 1 Compare event occurred

23.6.3 Load register (EPITx_LR)

The EPIT load register (EPIT_LR) contains the value that is to be loaded into the counter when EPIT counter reaches zero if the RLD bit in EPIT_CR is set. If the IOVW bit in the EPIT_CR is set then a write to this register overwrites the value of the EPIT counter register in addition to updating this registers value. This overwrite feature is active even if the RLD bit is not set.

Address: Base address + 8h offset

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

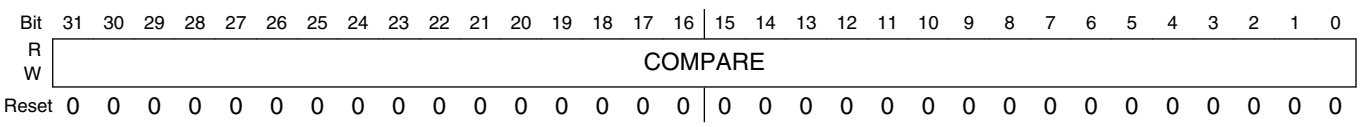
EPITx_LR field descriptions

Field	Description
LOAD	Load value. Value that is loaded into the counter at the start of each count cycle.

23.6.4 Compare register (EPITx_CMPR)

The EPIT compare register (EPIT_CMPR) holds the value that determines when a compare event is generated.

Address: Base address + Ch offset



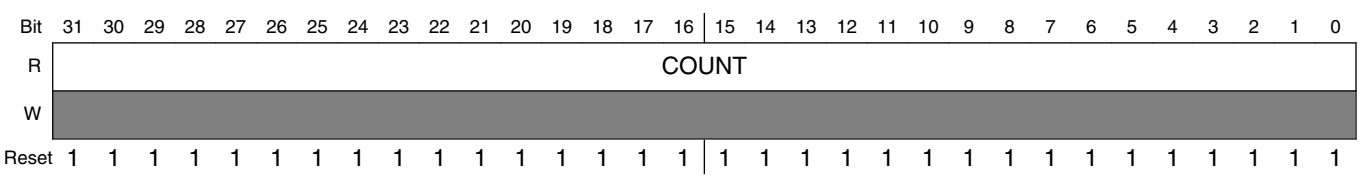
EPITx_CMPR field descriptions

Field	Description
COMPARE	Compare Value. When the counter value equals this bit field value a compare event is generated.

23.6.5 Counter register (EPITx_CNR)

The EPIT counter register (EPIT_CNR) contains the current count value and can be read at any time without disturbing the counter. This is a read-only register and any attempt to write into it generates a transfer error. But if the IOVW bit in EPIT_CR is set, the value of this register can be overwritten with a write to EPIT_LR. This change is reflected when this register is subsequently read.

Address: Base address + 10h offset



EPITx_CNR field descriptions

Field	Description
COUNT	Counter value. This contains the current value of the counter.

Chapter 24
Flexible Controller Area Network (FLEXCAN)

24.1 Overview

The Flexible Controller Area Network (FLEXCAN) module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification.

The CAN protocol was primarily designed to be used as a vehicle serial data bus meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FLEXCAN module is a full implementation of the CAN protocol specification, which supports both standard and extended message frames. 64 Message Buffers are supported.

24.1.1 Block Diagram

A general block diagram is shown in the figure below,which describes the main sub-blocks implemented in the FLEXCAN module, including the associated memory for storing Mailboxes, Rx Global Mask Registers, Rx Individual Mask Registers, Rx FIFO and Rx FIFO ID Filters.

Support for 64 Mailboxes and 6-deep Rx FIFO is provided. The functions of the sub-modules are described in subsequent sections.

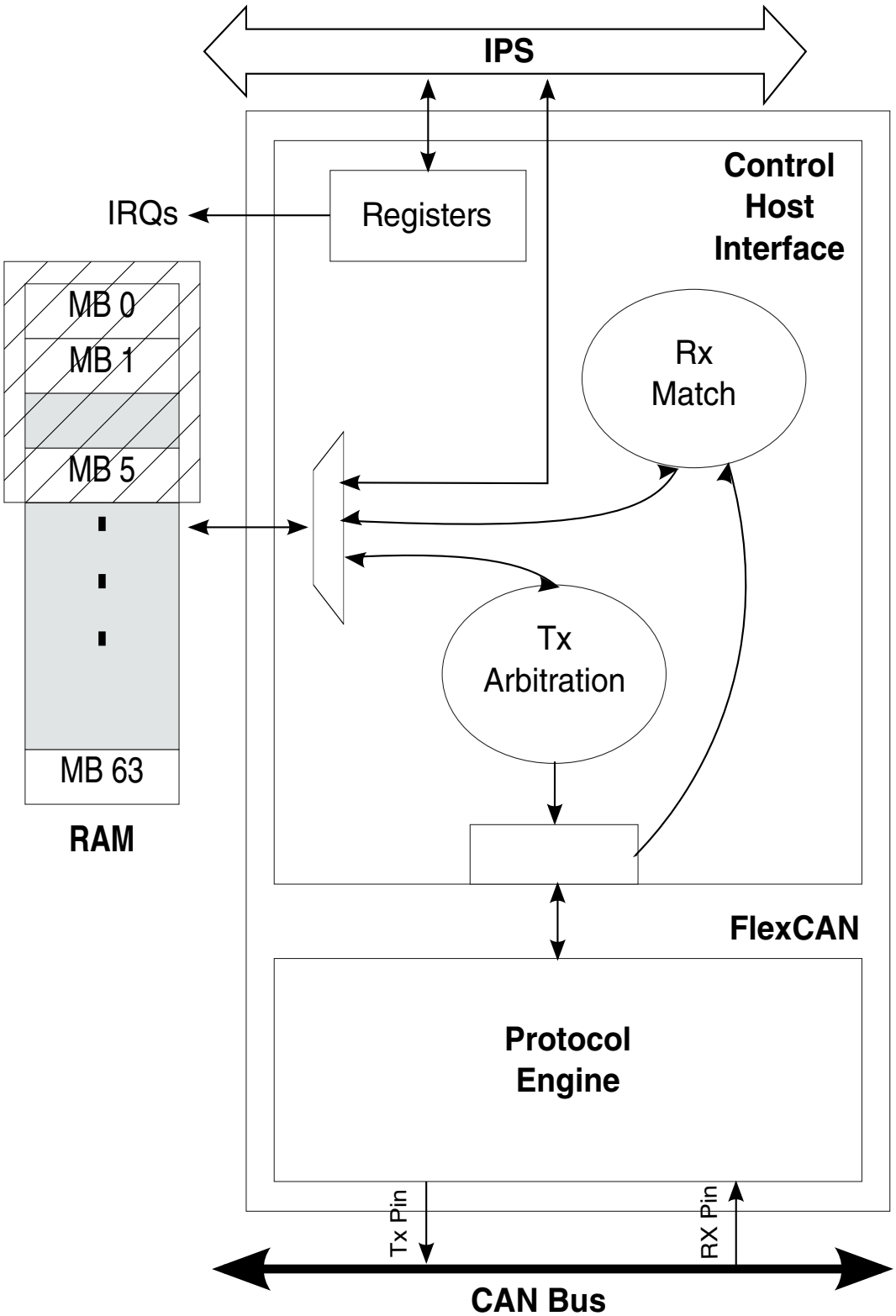


Figure 24-1. FLEXCAN Block Diagram

24.1.2 FLEXCAN Module Features

The FLEXCAN module includes these distinctive legacy features:

- Version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mb/sec
  - Content-related addressing
- Flexible Mailboxes of eight bytes data length
- Each Mailbox is configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Mailbox
- Full featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Transmission abort capability
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard or 512 partial (8 bits) IDs, with up to 32 individual masking capability
- 100% backwards compatibility with previous FLEXCAN version
- Unused structures space can be used as general purpose RAM space
- Listen only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity
- Configurable Glitch filter width to filter the noise on CAN bus when waking up
- Remote request frames may be handled automatically or by software.
- ID filter configuration in Normal Mode
- CAN bit time settings and configuration bits can only be written in Freeze Mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- SYNC bit status to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Selectable priority between Mailboxes and Rx FIFO during matching process

24.1.3 Modes of Operation

The FLEXCAN module has four functional modes: Normal Mode (User and Supervisor), Freeze Mode, Listen-Only Mode and Loop-Back Mode. There are also two low power modes: Disable Mode and Stop Mode.

- Normal Mode (User or Supervisor):

In Normal Mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze Mode:

It is enabled when the FRZ bit in the MCR Register is asserted. If enabled, Freeze Mode is entered when the HALT bit in MCR is set or when Debug Mode is requested at MCU level and the FRZ_ACK bit in the MCR Register is asserted by the FlexCAN . In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze Mode for more information.

- Listen-Only Mode:

The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back Mode:

The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FLEXCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The FLEXCAN_RX input pin is ignored and the FLEXCAN_TX output goes to the recessive state (logic '1'). FLEXCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FLEXCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Module Disable Mode:

This low power mode is entered when the MDIS bit in the MCR Register is asserted and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. Exit from this mode is done by negating the MDIS bit in the MCR Register. See Module Disable Mode for more information.

- Stop Mode:

This low power mode is entered when Stop Mode is requested at ARM level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the ARM that the clocks can be shut down globally. Exit from this mode happens when the Stop Mode request is removed or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop Mode for more information.

24.2 External Signals

The FLEXCAN module has two I/O signals.

Table 24-1. FLEXCAN External Signals

Signal	Description	Pad	Mode	Direction
FLEXCAN1_RX	FLEXCAN receive pin. This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_RX_DATA1	ALT4	I
		LCD_DATA09	ALT8	
		SD1_DATA1	ALT3	
		UART3_RTS_B	ALT2	
FLEXCAN1_TX	FLEXCAN transmit pin. This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_RX_DATA0	ALT4	O
		LCD_DATA08	ALT8	
		SD1_DATA0	ALT3	
		UART3_CTS_B	ALT2	
FLEXCAN2_RX	FLEXCAN receive pin. This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_TX_DATA0	ALT4	I
		LCD_DATA11	ALT8	
		SD1_DATA3	ALT3	
		UART2_RTS_B	ALT2	
FLEXCAN2_TX	FLEXCAN transmit pin. This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_RX_EN	ALT4	O
		LCD_DATA10	ALT8	
		SD1_DATA2	ALT3	
		UART2_CTS_B	ALT2	

24.3 Clocks

The table found here describes the clock sources for FLEXCAN.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 24-2. FLEXCAN Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_chi	ipg_clk_root	CHI clock
ipg_clk_pe	can_clk_root	Protocol Engine clock
ipg_clk_pe_nogate	can_clk_root	Protocol Engine clock (no gating)
ipg_clk_s	ipg_clk_root	Peripheral access clock
mem_ram_CLK	ipg_clk_root	RAM clock

24.4 Message Buffer Structure

Message Buffer Address: Base + 0x0080-0x047C

The Message Buffer structure used by the FLEXCAN module is represented in the figure found here.

Both Extended and Standard Frames (29-bit Identifier and 11-bit Identifier, respectively) used in the CAN specification are represented.

Table 24-3. Message Buffer Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0					CODE					SRR	IDE	RTR	DLC				TIME STAMP															
0x4	PRIO				ID Standard												ID Extended															
0x8	DATA BYTE 0								DATA BYTE 1								DATA BYTE 2								DATA BYTE 3							
0xC	DATA BYTE 4								DATA BYTE 5								DATA BYTE 6								DATA BYTE 7							

CODE - Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FLEXCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in the following tables. See Functional Description for additional information.

Table 24-4. Message Buffer Code for Rx buffers

CODE Description	Rx Code BEFORE receive New Frame	SRV[1]	Rx Code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE- MB is not active.	INACTIVE	-	-	-	MB does not participate in the matching process.
0b0100: EMPTY - MB is active and empty.	EMPTY	-	FULL	-	When a frame is received successfully (after move-in process. Refer to Move-in for details), the CODE field is automatically updated to FULL.
0b0010: FULL - MB is full.	FULL	Yes	FULL	-	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. Refer to Matching Process for matching details related to FULL code.
		No	OVERRUN	-	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. Refer to Matching Process for details about overrun behavior.
0b0110: OVERRUN - MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	-	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB, the code returns to FULL.
		No	OVERRUN	-	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Matching Process for details about overrun behavior.
0b1010: RANSWER[4] - A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.	RANSWER	-	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received, after that a MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). Refer to Matching Process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
			-	1	This code is ignored during matching and arbitration process. Refer to Matching Process for details.

Table continues on the next page...

Table 24-4. Message Buffer Code for Rx buffers (continued)

CODE Description	Rx Code BEFORE receive New Frame	SRV[1]	Rx Code AFTER successful reception[2]	RRS[3]	Comment
CODE[0]=1b1: BUSY - FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[5]	-	FULL	-	Indicates that the MB is being updated, it will be negated automatically and does not interfere on the next CODE.
			OVERRUN	-	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered successful reception after the frame to be moved to MB (move-in process). Refer to Move-in for details)
3. Remote Request Stored bit from CTRL2 register. Refer to CTRL2 for details.
4. Code 4'b1010 is not considered as a Tx and a MB with this code should not to be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 24-5. Message Buffer Code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE - MB is not active	INACTIVE	-	-	MB does not participate in the arbitration process.
0b1001: ABORT - MB is aborted	ABORT	-	-	MB does not participate in the arbitration process. .
0b1100: DATA - MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE - MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER - MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	-	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of match to a remote request frame. The remote response frame will be transmitted unconditionally once and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. Refer to Matching Process and Arbitration process for details.

SRR - Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to '1' by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FLEXCAN receives this bit as dominant, then it is interpreted as arbitration loss.

1= Recessive value is compulsory for transmission in Extended Format frames

0= Dominant is not a valid value for transmission in Extended Format frames

IDE - ID Extended Bit

This bit identifies whether the frame format is standard or extended. It is also used as part of the reception filter.

1= Frame format is extended

0= Frame format is standard

RTR - Remote Transmission Request

This bit affects the behavior of Remote Frames and is part of the reception filter. Refer to the tables above and RRS bit in Control 2 Register (FLEXCAN_CTRL2) for additional details.

If FLEXCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FLEXCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission.

1= Indicates the current MB has a Remote Frame to be transmitted if MB is Tx. If the MB is Rx then incoming Remote Request Frames may be stored.

0= Indicates the current MB has a Data Frame to be transmitted. In Rx MB it may be considered in matching processes.

DLC - Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x08 through 0x0F of the MB space (see the first table above). In reception, this field is written by the FLEXCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the ARM and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the Frame to be transmitted is a Remote Frame and does not include the data field, regardless of the Length field. The DLC field indicates which DATA BYTEs are valid as shown in the table below.

TIME STAMP - Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO - Local priority

This 3-bit field is only used when MCR[LPRIO_EN] bit is asserted and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID - Frame Identifier

In Standard Frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0-7 - Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (n) is valid only if *n* is less than DLC as shown in the table below.

For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 24-6. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0-1
3	DATA BYTE 0-2
4	DATA BYTE 0-3
5	DATA BYTE 0-4
6	DATA BYTE 0-5
7	DATA BYTE 0-6
8	DATA BYTE 0-7

24.5 Rx FIFO Structure

When the MCR[RFEN] bit is set, the memory area from $80 to $DC (which is normally occupied by MBs 0 to 5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a Message Buffer. This output contains the oldest message received and not read yet. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, that starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6 up to 37) depending on the CTRL2[RFFN] field setting, contains the ID Filter Table (configurable from 8 to 128 memory positions) that specifies filtering criteria for accepting frames into the FIFO.Table 24-7 shows the Rx FIFO data structure.

Each ID Filter Table Element occupies an entire 32-bit word and can be compounded by one, two or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. Table 24-8, Table 24-9 and Table 24-10 show the IDAF indexation. Table 24-11 show the three different formats that the IDAF can assume, depending on the MCR[IDAM] field setting. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Out of reset, the ID Filter Table flexible memory area defaults to 0xE0 and only extends to 0xFC, which corresponds to MBs 6 to 7 for RFFN=0.

Table 24-7. Rx FIFO Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80										SRR	IDE	RTR	DLC				TIME STAMP															
0x84				ID Standard													ID Extended															
0x88	Data Byte 0									Data Byte 1							Data Byte 2								Data Byte 3							
0x8C	Data Byte 4									Data Byte 5							Data Byte 6								Data Byte 7							
0x90 to 0xDC	Reserved																															
0xE0	ID Filter Table Element 0																															
0xE4	ID Filter Table Element 1																															
0xE8 to 0x2D4	ID Filter Table Elements 2 through 125																															
0x2D8	ID Filter Table Element 126																															
0x2DC	ID Filter Table Element 127																															

Table 24-8. Position of ID Filter Table Elements in format A

Element	31																												0
0	IDAF0																												
1	IDAF1																												
2 through 125	Identifier Acceptance Filter 2 through 125																												
126	IDAF126																												
127	IDAF127																												

Table 24-9. Position of ID Filter Table Elements in format B

Element	31															16	15												0
0	IDAF0																IDAF1												
1	IDAF2																IDAF3												
2 through 125	Identifier Acceptance Filter 4 through 251																												
126	IDAF252																IDAF253												
127	IDAF254																IDAF255												

Table 24-10. Position of ID Filter Table Elements in format C

Element	31							24	23							16	15							8	7						0
0	IDAF0								IDAF1								IDAF2								IDAF3						
1	IDAF4								IDAF5								IDAF6								IDAF7						
2 through 125	Identifier Acceptance Filter 8 through 503																														
126	IDAF504								IDAF505								IDAF506								IDAF507						
127	IDAF508								IDAF509								IDAF510								IDAF511						

Table 24-11. Identifier Acceptance Filter Format A,B and C

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	RTR	IDE	RXIDA (Standard = 29-19, Extended = 29-1)																													
B	RTR	IDE	RXIDB_0 (Standard = 29-19, Extended = 29-16)														RTR	IDE	RXIDB_1 (Standard = 13-3, Extended = 13-0)														
C	RXIDC_0								RXIDC_1								RXIDC_2								RXIDC_3						

Table 24-11. Identifier Acceptance Filter Format A,B and C

	(Std/Ext = 31-24)	(Std/Ext = 23-16)	(Std/Ext = 15-8)	(Std/Ext = 7-0)

RTR - Remote Frame

This bit specifies whether Remote Request Frames are accepted into the FIFO if they match the target ID in Formats A and B. If Format C is chosen the acceptance does not depend on whether the frame is a Remote Request Frame or not.

1= Remote Frames can be accepted and data frames are rejected

0= Remote Frames are rejected and data frames can be accepted

IDE - Extended Frame

Specifies if either Extended or Standard Format frames are accepted into the FIFO if they match the target ID in Formats A and B. If Format C is chosen the acceptance does not depend on whether the frame is of the Extended or Standard Format.

1= Extended frames can be accepted and standard frames are rejected

0= Extended frames are rejected and standard frames can be accepted

RXIDA - Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the Standard Format (IDAF's or incoming frame's IDE bit is negated), only the 11 most significant bits (29 to 19) are used for frame identification. In the Extended Format (both IDAF's and incoming frame's IDE are asserted), all bits are used.

RXIDB_0, RXIDB_1 - Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the Standard Format (IDAF's or incoming frame's IDE bit is negated), the 11 most significant bits (29 to 19 and 13 to 3) are used for frame identification. In the Extended Format (both IDAF's and incoming frame's IDE are asserted), all 14 bits of the field are compared with the 14 most significant bits of the Identifier of the incoming frame. The 15 least significant bits of the Identifier of an incoming Extended Format frame do not affect the acceptance.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 - Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both Standard Format and Extended Format, all 8 bits of the field are compared to the 8 most significant bits of the Identifier of the incoming frame. The 3 least significant bits of the Identifier of an incoming Standard Format frame and the 21 least significant bits of the Identifier of an incoming Extended Format frame do not affect the acceptance.

24.6 Functional Description

This section provides a complete functional description of the block.

24.6.1 Functional Overview

The FLEXCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames.

The mailbox system consists of a set of 64 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message Buffer Structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a *matching* algorithm makes it possible to store received frames only into MBs that have the same ID. A masking scheme makes it possible to match the ID programmed on the MB with a range of Identifiers on received CAN frames. For transmission, an *arbitration* algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 24-4). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 24-5).

24.6.2 Transmit Process

In order to transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the procedure found here.

1. Check if the respective interruption bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted (see

Transmission Abort Mechanism). If backwards compatibility is desired (MCR[AEN] bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Message Buffer Inactivation).

3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control and Code fields of the Control and Status word to activate the MB.

Once the MB is activated, it will participate into the arbitration process and eventually be transmitted according to its priority.

At the end of the successful transmission, the value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the MB's Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB in step four (see Table 24-4 and Table 24-5 in Message Buffer Structure)

When the Abort feature is enabled (MCR[AEN] bit is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked, therefore the CPU is not able to update it until it negates the Interrupt Flag. It means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

24.6.3 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest Mailbox number and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CTRL2[TASD] field value. See Control 2 Register (FLEXCAN_CTRL2) for details.
- During the error delimiter field of the CAN frame
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When ARM write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.

- When CHI is in Idle state and ARM writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state
- Upon leaving Freeze Mode or Low Power Mode

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CTRL1[LBUF] and MCR[LPRIO_EN] bits settings.

24.6.3.1 Lowest Mailbox number first

If CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. MCR[LPRIO_EN] bit has no effect when CTRL1[LBUF] is asserted.

24.6.3.2 Highest Mailbox priority first

If CTRL1[LBUF] bit is negated then the arbitration process searches the active Tx Mailbox with the highest priority, and this Mailbox would have a higher probability to win the arbitration on CAN bus.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest priority Tx Mailbox is the one that has the least arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values the lowest Mailbox number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIO_EN] bit setting.

24.6.3.2.1 Local Priority disabled

If MCR[LPRIO_EN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see Table 24-12) in such a way that the Local Priority is disabled.

Table 24-12. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)

Table continues on the next page...

Table 24-12. Composition of the arbitration value when Local Priority is disabled (continued)

Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

24.6.3.2.2 Local Priority enabled

If Local Priority is desired MCR[LPRIO_EN] must be asserted.

In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the table below).

Table 24-13. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18](11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

Once the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called "move-out" and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Freeze Mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (Data Length Code) field value is greater than that.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. Arbitration start point depends on instantiation parameters NUMBER_OF_MB and TASD. Additionally, TASD value may be changed (see Control 2 Register (FLEXCAN_CTRL2)) to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT=124 to 128. Arbitration start point depends on instantiation parameters NUMBER_OF_MB and TASD. Additionally, TASD value may be changed (see Control 2 Register (FLEXCAN_CTRL2)) to optimize the arbitration start point.
- During C/S write by CPU in Bus Idle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after arbitration process has finished, then TX arbitration machine begins a new arbitration process.
- Arbitration winner deactivation during a valid arbitration window.
- Upon Leave Freeze Mode. If there is a re-synchronization during WaitForBusIdle arbitration process is restarted.

Arbitration process stops in the following situation:

- All Mailboxes were scanned.
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish Tx arbitration process. For instance, a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus.
- Low Power or Freeze Mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time.
- C/S write during arbitration if write is performed in a MB which number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx Arbitration process in progress.
- Rx Match has just updated a Rx Code to Tx Code.
- Entering Bus off state.

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- C/S write during arbitration if write is performed in a MB which number is higher than the Tx arbitration pointer.

24.6.4 Receive Process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the steps listed here.

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Message Buffer Inactivation), preferably with a *safe inactivation* (see Transmission Abort Mechanism);
2. Write the ID word;
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

Once the Mailbox is activated in the third step, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in process* (see Move-in) as follows:

1. The received Data field (8 bytes at most) is stored;
2. The received Identifier field is stored;
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox Time Stamp field;
4. The received SRR, IDE, RTR and DLC fields are stored;
5. The CODE field in the Control and Status word is updated. (see Table 24-4 and Table 24-5 in Section Message Buffer Structure)
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for the CPU servicing (read) the frame received in a Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox;
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Message Buffer Lock Mechanism;
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in;
4. Acknowledge the proper flag at IFLAG registers;
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should synchronize to frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the

CODE field will not return to EMPTY. It will remain FULL. If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received message matching the filter of that Mailbox may be lost.

In summary: never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in a Mailbox may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists a matching Rx Mailbox, provided the MCR[SRX_DIS] bit is not asserted. If MCR[SRX_DIS] bit is asserted, FlexCAN will not store messages transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN messages through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze Mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see Interrupt Masks 1 Register (FLEXCAN_IMASK1), bit IFLAG[BUF5I] - Frames available in Rx FIFO), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional - needed only if a mask was used for IDE and RTR bits);
2. Read the ID field (optional - needed only if a mask was used);
3. Read the Data field;
4. Read the RXFIR register (optional);
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to IFLAG[BUF5I] bit (mandatory - releases the MB and allows the CPU to read the next Rx FIFO entry)

24.6.5 Matching Process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus.

If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. In any case, the matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- if the received frame is a remote frame, the start point is the CRC field of the frame;
- if the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame;
- if the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame;

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the SMB will be transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results will be transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures* . The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. Please, refer to the following table for details.

Table 24-14. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID][1]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). In case of SMB[IDE] to be negated, the ID is only 11 bits (ID Standard). Please, refer to Message Buffer Structure for ID details. For FIFO structure, the ID depends on IDAM. Please, refer to Rx FIFO Structure for IDAM details.
2. cmp: Compares the SMB contents with the MB contents regardless the masks.
3. no_cmp: The SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the SMB contents with MB contents taking into account the masks.

5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- the CODE field of the Mailbox is EMPTY;
- the CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the ARM and unlocked as described in Message Buffer Lock Mechanism);
- the CODE field of the Mailbox is either FULL or OVERRUN and an inactivation is performed. (see Message Buffer Inactivation)
- the Rx FIFO is not full.

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the MCR[IRMQ] bit. If it is negated the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- if the Rx FIFO is a matched structure and is free-to-receive then the Rx FIFO is the matching winner regardless of the scan for Mailboxes;
- otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above.

If the selected priority is Mailboxes first:

- if a free-to-receive matched Mailbox is found, it is the matching winner regardless the scan for Rx FIFO;
- if no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO;
- if both conditions above are not satisfied and a non free-to-receive matched Mailbox is found then the matching winner determination is conditioned by the MCR[IRMQ] bit:
  - if MCR[IRMQ] bit is negated the matching winner is the first matched Mailbox;
  - if MCR[IRMQ] bit is asserted the matching winner is the Rx FIFO if it is a free-to-receive matched structure, otherwise the matching winner is the last non free-to-receive matched Mailbox.

Please, refer to the table below for a summary of matching possibilities.

If a non-safe Mailbox inactivation (see Message Buffer Inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not "free-to-receive", so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

Table 24-15. Matching Possibilities and Resulting Reception Structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception Structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMB	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMB	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)

Table continues on the next page...

Table 24-15. Matching Possibilities and Resulting Reception Structures (continued)

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception Structure	Description
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMB	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMB	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMB	Overrun

1. It is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. It is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as the FIFO didn't exist (CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for ARM to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. ARM can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Please refer to Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK) . During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze Mode , otherwise they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RGXMASK, RX14MASK, RX15MASK and RXFGMASK) for backward compatibility. This alternate masking scheme is enabled when the IRMQ bit in the MCR Register is negated.

24.6.6 Move Process

There are two types of move process, namely move-in and move-out.

24.6.6.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead.

The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching Process) and all of the following conditions are true:

- the CAN bus has reached or let past either:
  - the second bit of Intermission field next to the frame that carried the message that is in the Rx SMB;
  - the first bit of an overload frame next to the frame that carried the message that is in the Rx SMB;
- there is no ongoing matching process;
- the destination Mailbox is not locked by ARM;
- there is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the document and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- the destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished;
- there is a previous pending move-in to the same destination Mailbox.
- the Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled;
- any CAN protocol error is detected.

Note that the pending move-in is not cancelled if the module enters in Freeze or Low Power Mode. It only stays on hold waiting for exiting Low Power Mode and to be unlocked. If an MB is unlocked during Freeze Mode, the move-in happens immediately.

The move-in process consists of the following steps:

1. if the message is destined to the Rx FIFO, push IDHIT into the RXFIR FIFO;
2. reads the words DATA0-3 and DATA4-7 from the Rx SMB;
3. writes it in the words DATA0-3 and DATA4-7 of the Rx Mailbox;
4. reads the words Control/Status and ID from the Rx SMB;
5. writes it in the words Control/Status and ID of the Rx Mailbox, updating the CODE field according to Table 24-4.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Message Buffer Inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed in such a way that ARM beware that the Message Buffer content is temporarily incoherent.

24.6.6.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Arbitration process).

The move-out occurs in the following conditions:

- the first bit of Intermission field;
- during Bus off field when TX Error Counter is in the 124 to 128 range;
- during BusIdle field
- during Wait For Bus Idle field

The move-out process is not atomic. Only ARM has priority to access the memory concurrently out of BusIdle state. In BusIdle, the move-out has the lowest priority to the concurrent memory accesses.

24.6.7 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the ARM must obey the rules described in

Any form of ARM accessing an MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

24.6.7.1 Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform ARM if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

In order to abort a transmission, ARM must write a specific abort code (0b1001) to the CODE field of the Control and Status word. The active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a Mailbox that is currently being transmitted, or to a Mailbox that was already loaded into the SMB for transmission, the write operation is blocked and the MB is kept active, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze Mode
- The module enters in BusOff state
- There is an overload frame

If none of conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register and an interrupt to the ARM is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. In the other hand, if one of the above conditions is reached, the frame is not transmitted, therefore the abort code is written into the CODE field, the interrupt flag is set in the IFLAG and an interrupt is (optionally) generated to ARM.

If ARM writes the ABORT code before the transmission begins internally, then the write operation is not blocked, therefore the MB is updated and the interrupt flag is set. In this way ARM just needs to read the abort code to make sure the active MB was *safely inactivated* . Although the AEN bit is asserted and ARM wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

1. ARM checks the corresponding IFLAG and clears it, if asserted.
2. ARM writes 0b1001 into the CODE field of the C/S word.
3. ARM waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
4. ARM reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).

5. It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

24.6.7.2 Message Buffer Inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing ARM to rely on Mailbox data coherence after having updated it, even in Normal Mode.

If a Mailbox is inactivated it does not participate neither in the arbitration nor in the matching process until it is reactivated. See Transmit Process and Receive Process for more detailed instruction on how to inactivate and reactivate a Mailbox.

In order to inactivate a Mailbox ARM must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

As the user is not able to synchronize the CODE field update with the FlexCAN internal processes an inactivation can lead to undesirable results:

- a frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter;
- a frame containing the message within the inactivated Tx Mailbox may be transmitted without notice.

In order to eliminate such risk and perform a *safe inactivation* ARM must use the following mechanism along with the inactivation itself:

- for Tx Mailboxes, the Transmission Abort (see Transmission Abort Mechanism);

The inactivation automatically unlocks the Mailbox (see Message Buffer Lock Mechanism).

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on FIFO region by FlexCAN. ARM must keep the data coherence into FIFO region when RFEN is asserted.

24.6.7.3 Message Buffer Lock Mechanism

Besides MB inactivation, FlexCAN has another data coherence mechanism for the receive process. When ARM reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that ARM wants to read the whole MB in an

atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when ARM reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code or when ARM writes into C/S word from locked MB. The MB locking is done to prevent a new frame to be written into the MB while ARM is reading it.

The locking mechanism only applies to Rx MBs that are not part of FIFO and have a code different than INACTIVE (0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the ARM decides to read MB number 5 and at the same time another message with the same ID is arriving. When ARM reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If ARM reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If ARM inactivates a locked Rx Mailbox, then its lock status is negated and the Mailbox is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the Mailbox.An MB is unlocked when ARM reads the Free Running Timer Register (see Free Running Timer Register (FLEXCAN_TIMER)), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during any of the low power modes (see in Modes of Operation specific information on Module Disable or Stop modes) and it will take place only when the module resumes to Normal or Freeze modes.

1. In previous FlexCAN versions, reading the C/S word locks the MB even if it is EMPTY. This behavior is maintained when the IRMQ bit is negated.

24.6.8 Rx FIFO

The receive-only FIFO is enabled by asserting the RFEN bit in the MCR.

The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature. The FIFO is 6-message deep, therefore when the FIFO is enabled, the memory region occupied by the first 6 Message Buffers is reserved for use of the FIFO engine (see Rx FIFO Structure). ARM can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The IFLAG[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, ARM can read the message (accessing the output of the FIFO as a Message Buffer) and the RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the RXFIR with the attributes of that message, reissuing the interrupt to ARM. Otherwise, the flag remains negated. The output of the FIFO is only valid when the IFLAG[BUF5I] is asserted.

The IFLAG[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until ARM clears it.

The IFLAG[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the ARM clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO Structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read by accessing the RXFIR register. The RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid whilst the IFLAG[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to thirty two elements of the ID Filter Table are individually affected by the Individual Mask Registers (RXIMR0 - RXIMR31), according to CTRL2[RFFN] setting (refer to Control 2 Register (FLEXCAN_CTRL2)), allowing very powerful filtering criteria to be defined. If the MCR[IRMQ] bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO ID Filter Table is affected by RXFGMASK.

24.6.9 CAN Protocol Related Features

24.6.9.1 Remote Frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by writing the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame was received and matched a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this

mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.

- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the ARM. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

24.6.9.2 Overload Frames

FLEXCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

24.6.9.3 Time Stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

Note that the Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in Control 1 Register (FLEXCAN_CTRL1).

24.6.9.4 Protocol Timing

The FLEXCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See Control 1 Register (FLEXCAN_CTRL1) .

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments[2] (reference Table 24-16):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

2. For further explanation of the underlying concepts please refer to ISO/DIS 11519-1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

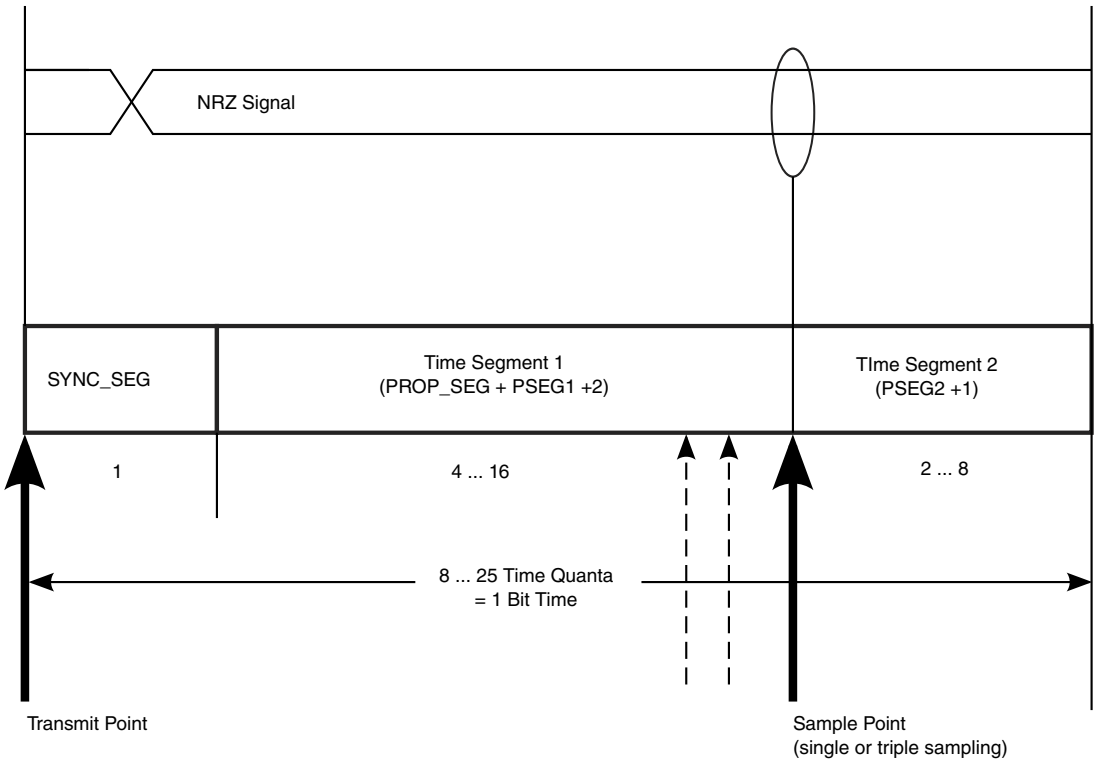


Figure 24-2. Segments within the Bit Time

Whenever CAN bit is used as a measure of duration (e.g. MCR[FRZ_ACK] and MCR[LPM_ACK] in Module Configuration Register (FLEXCAN_MCR)), the number of peripheral clocks in one CAN bit can be calculated as:

$$NCCP = \frac{f_{sys} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)}{f_{CANCLK}}$$

where:

NCCP is the number of peripheral clocks in one CAN bit;

f_{CANCLK} is the Protocol Engine (PE) Clock in Hz;

f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;

PSEG1 is the value in CTRL1[PSEG1] field;

PSEG2 is the value in CTRL1[PSEG2] field;

PROPSEG is the value in CTRL1[PROPSEG] field;

PRESDIV is the value in CTRL1[PRESDIV] field.

For example, 180 CAN bits = 180 x NCCP peripheral clock periods.

Figure 24-2 gives an overview of the CAN compliant segment settings and the related parameter values.

Table 24-16. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 24-17. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

24.6.9.5 Arbitration and Matching Timing

During normal reception and transmission of frames, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

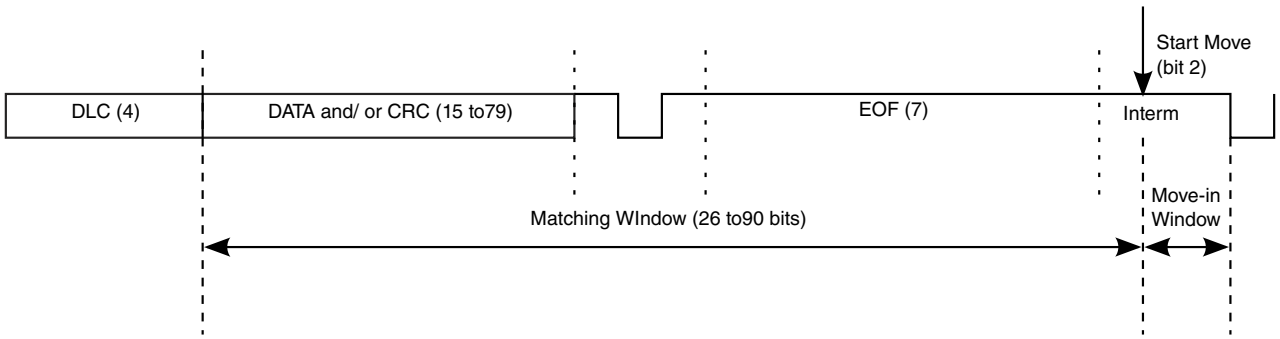


Figure 24-3. Matching and Move-In Time Windows

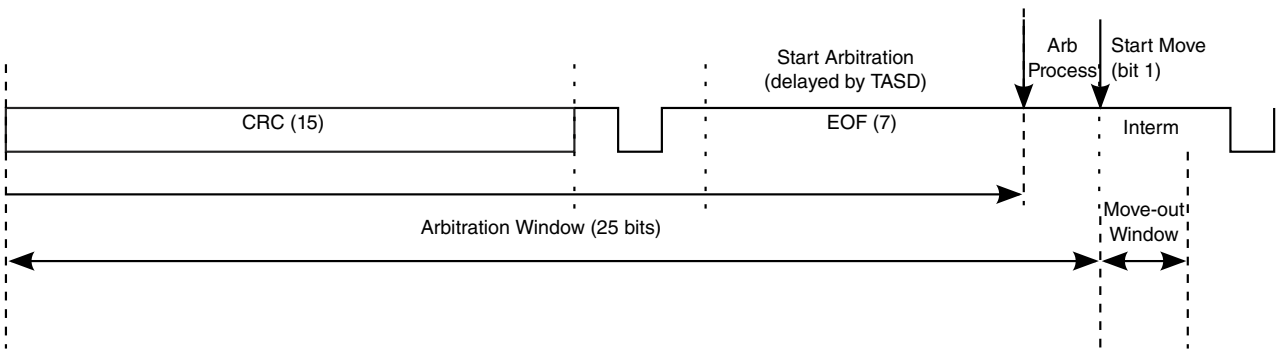


Figure 24-4. Arbitration and Move-Out Time Windows

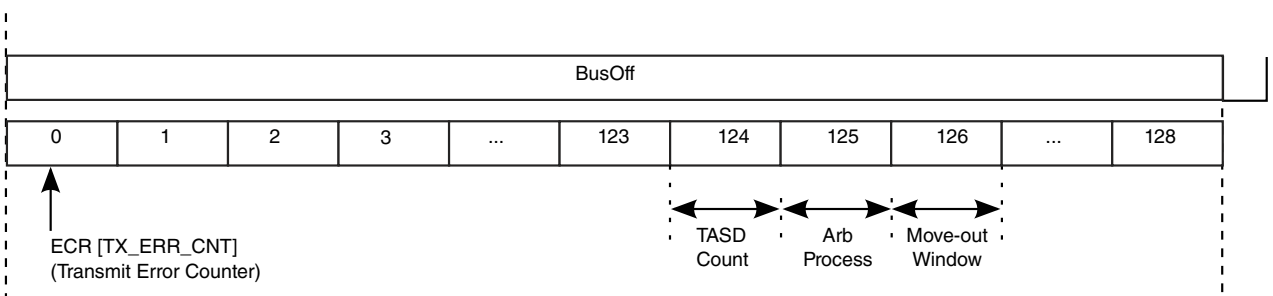


Figure 24-5. Arbitration at the end of Bus Off and Move-Out Time Windows

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time window. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 24-17
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, i.e. the PLL can not be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in the following table.

Table 24-18. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate

Number of Message Buffers	RFEN	Minimum Number of Peripheral Clocks per CAN bit
16 and 32	0	16
64	0	25
16	1	16
32	1	17
64	1	30

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in Table 24-18 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

24.6.10 Modes of Operation Details

The FlexCAN module has four functional modes (Normal Mode, Freeze Mode, Listen-Only Mode and Loop-Back Mode) and two low power modes (Disable Mode and Stop Mode).

See in Modes of Operation an introductory description of all these modes of operation. The following sub-sections bring functional details on Freeze mode and the low power modes.

24.6.10.1 Freeze Mode

This mode is requested by ARM through the assertion of the HALT bit in the MCR Register or when the MCU is put into Debug Mode . In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in any of the low power modes (Disable, Stop). The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The ARM must only consider the FlexCAN in Freeze Mode when both request and acknowledgement conditions are satisfied.

When Freeze Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. Pending move-in is not taken in account
- Ignores the FLEXCAN_RX input pin and drives the FLEXCAN_TX pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze Mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CTRL1[CLK_SRC] bit that can be read but cannot be written.

Exiting Freeze Mode is done in one of the following ways:

- ARM negates the FRZ bit in the MCR Register
- The ARM is removed from Debug Mode and the HALT bit is negated

The FRZ_ACK bit is negated after protocol engine recognizes the negation of freeze request. Once out of Freeze Mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

24.6.10.2 Module Disable Mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the ARM through the assertion of the MDIS bit in the MCR Register and the acknowledgement is obtained through the assertion by the FlexCAN of the LPM_ACK bit in the same register. The ARM must only consider the FlexCAN in Disable Mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze Mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit. The ability to shut down the clocks depends on how FlexCAN is integrated into the MCU. If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. Pending move-in is not taken in account
- Ignores its FLEXCAN_RX input pin and drives its FLEXCAN_TX pin as recessive

- May shut down the clocks to the PE and CHI sub-modules, depending on how FlexCAN is integrated into the MCU
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the ARM to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode depending on how FlexCAN RAM is integrated into the ARM. Exiting from this mode is done by negating the MDIS bit by ARM, which make FlexCAN requests to resume the clocks and negates the LPM_ACK bit after CAN protocol engine recognizes the negation of disable mode requested by ARM.

24.6.10.3 Stop Mode

This is a system low power mode in which system clocks can be stopped for maximum power savings.. To enter stop mode, the CPU should manually assert a global Stop Mode request (see the CAN1_STOP_REQ and CAN2_STOP_REQ bit in the register IOMUXC_GPR4)and check the acknowledgement asserted by the FlexCAN (see the CAN1_STOP_ACK and CAN2_STOP_ACK in the register IOMUXC_GPR4) . The CPU must only consider the FlexCAN in Stop Mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop Mode request during Freeze Mode, it sets the LPM_ACK bit, negates the FRZ_ACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally. If Stop Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish.Pending move-in is not taken in account
- Ignores its FLEXCAN_RX input pin and drives its FLEXCAN_TX pin as recessive
- Sets the NOT_RDY and LPM_ACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Exiting Stop Mode is done in one of the following ways:

- ARM resuming the clocks and removing the Stop Mode request
- ARM resuming the clocks and Stop Mode request as a result of the Self Wake mechanism

In the Self Wake mechanism, if the SLF_WAK bit in MCR Register was set at the time FlexCAN entered Stop Mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR Register and, if enabled by the WAK_MSK bit in MCR, generates a Wake Up interrupt to the ARM. Upon receiving the interrupt, the ARM should resume the clocks and remove the Stop Mode request manually. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the FLEXCAN_RX input line while in Stop Mode. See the WAK_SRC bit in Module Configuration Register (FLEXCAN_MCR) . This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments, the glitch filter width can be set in Glitch Filter Width Register (FLEXCAN_GFWR).

24.6.11 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to Ored interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning and Wake Up)).

The number of actual sources depends on the configured number of message buffers.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the ARM writes it to '1' (unless another interrupt is generated at the same time).

If the Rx FIFO is enabled (bit RFEN on MCR set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFLAG1 becomes the "FIFO Overflow" flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the "Frames Available in FIFO flag" and bits 4-0 are unused. See Interrupt Flags 1 Register (FLEXCAN_IFLAG1) for more information.

A combined interrupt for all MBs is also generated by an Or of all the interrupt sources from MBs. This interrupt gets generated when any of the Mailboxes or FIFO generates an interrupt. The ARM must read the IFLAG Registers to determine which MB or FIFO caused the interrupt.

The other 5 interrupt sources (Bus Off, Error, Tx Warning, Rx Warning and Wake Up generate interrupts like the MB ones, and can be read from both the Error and Status Register 1 and 2. The Bus Off, Error, Tx Warning and Rx Warning interrupt mask bits are located in the Control 1 Register and the Wake-Up interrupt mask bit is located in the MCR.

24.7 Initialization/Application Information

This section provide instructions for initializing the FLEXCAN module.

24.7.1 FLEXCAN Initialization Sequence

The FLEXCAN module may be reset in two ways:

- SOC level hard reset which resets all memory mapped registers asynchronously
- SOFT_RST bit in MCR, which resets some of the memory mapped registers synchronously

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

After the module is enabled (MDIS bit negated), FLEXCAN automatically goes to Freeze Mode. In Freeze Mode, FLEXCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR Register are set. The FLEXCAN_TX pin is in recessive state and FLEXCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FLEXCAN is put into Freeze Mode. The following is a generic initialization sequence applicable to the FLEXCAN module:

- Initialize the Module Configuration Register
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRN_EN bit

    - If required, disable frame self reception by setting the SRX_DIS bit
    - Enable the FIFO by setting the RFEN bit
    - Enable the abort mechanism by setting the AEN bit
    - Enable the local priority feature by setting the LPRIO_EN bit
- Initialize the Control Register
    - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
    - Determine the bit rate by programming the PRESDIV field
    - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
    - The Control and Status word of all Message Buffers must be initialized
    - If FIFO was enabled, the 8-entry ID table must be initialized
    - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in CTRL Register (for Bus Off and Error interrupts) and in MCR Register for Wake-Up interrupt
- Negate the HALT bit in MCR

Starting with the last event, FLEXCAN attempts to synchronize to the CAN bus.

24.8 FLEXCAN Memory Map/Register Definition

The complete memory map for a FLEXCAN module with 64 MBs capability is shown in the following table. Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV bit in the MCR Register. The MCR register allows only Supervisor access regardless the SUPV bit state.

The FLEXCAN module stores CAN messages for transmission and reception using a Mailboxes and Rx FIFO structure.

FLEXCAN memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
209_0000	Module Configuration Register (FLEXCAN1_MCR)	32	R/W	5980_000Fh	24.8.1/1068
209_0004	Control 1 Register (FLEXCAN1_CTRL1)	32	R/W	0000_0000h	24.8.2/1073
209_0008	Free Running Timer Register (FLEXCAN1_TIMER)	32	R/W	0000_0000h	24.8.3/1076

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
209_0010	Rx Mailboxes Global Mask Register (FLEXCAN1_RXMGMASK)	32	R/W	FFFF_FFFFh	24.8.4/1076
209_0014	Rx Buffer 14 Mask Register (FLEXCAN1_RX14MASK)	32	R/W	FFFF_FFFFh	24.8.5/1077
209_0018	Rx Buffer 15 Mask Register (FLEXCAN1_RX15MASK)	32	R/W	FFFF_FFFFh	24.8.6/1078
209_001C	Error Counter Register (FLEXCAN1_ECR)	32	R/W	0000_0000h	24.8.7/1079
209_0020	Error and Status 1 Register (FLEXCAN1_ESR1)	32	R/W	0000_0000h	24.8.8/1080
209_0024	Interrupt Masks 2 Register (FLEXCAN1_IMASK2)	32	R/W	0000_0000h	24.8.9/1084
209_0028	Interrupt Masks 1 Register (FLEXCAN1_IMASK1)	32	R/W	0000_0000h	24.8.10/ 1084
209_002C	Interrupt Flags 2 Register (FLEXCAN1_IFLAG2)	32	R/W	0000_0000h	24.8.11/ 1085
209_0030	Interrupt Flags 1 Register (FLEXCAN1_IFLAG1)	32	R/W	0000_0000h	24.8.12/ 1085
209_0034	Control 2 Register (FLEXCAN1_CTRL2)	32	R/W	0000_0000h	24.8.13/ 1087
209_0038	Error and Status 2 Register (FLEXCAN1_ESR2)	32	R	0000_0000h	24.8.14/ 1093
209_0044	CRC Register (FLEXCAN1_CRCR)	32	R	0000_0000h	24.8.15/ 1095
209_0048	Rx FIFO Global Mask Register (FLEXCAN1_RXFGMASK)	32	R/W	FFFF_FFFFh	24.8.16/ 1096
209_004C	Rx FIFO Information Register (FLEXCAN1_RXFIR)	32	R	0000_0000h	24.8.17/ 1097
209_0880	Rx Individual Mask Registers (FLEXCAN1_RXIMR0_RXIMR63)	32	R/W	0000_0000h	24.8.18/ 1098
209_09E0	Glitch Filter Width Registers (FLEXCAN1_GFWR)	32	R/W	0000_007Fh	24.8.19/ 1098
209_4000	Module Configuration Register (FLEXCAN2_MCR)	32	R/W	5980_000Fh	24.8.1/1068
209_4004	Control 1 Register (FLEXCAN2_CTRL1)	32	R/W	0000_0000h	24.8.2/1073
209_4008	Free Running Timer Register (FLEXCAN2_TIMER)	32	R/W	0000_0000h	24.8.3/1076
209_4010	Rx Mailboxes Global Mask Register (FLEXCAN2_RXMGMASK)	32	R/W	FFFF_FFFFh	24.8.4/1076
209_4014	Rx Buffer 14 Mask Register (FLEXCAN2_RX14MASK)	32	R/W	FFFF_FFFFh	24.8.5/1077
209_4018	Rx Buffer 15 Mask Register (FLEXCAN2_RX15MASK)	32	R/W	FFFF_FFFFh	24.8.6/1078
209_401C	Error Counter Register (FLEXCAN2_ECR)	32	R/W	0000_0000h	24.8.7/1079
209_4020	Error and Status 1 Register (FLEXCAN2_ESR1)	32	R/W	0000_0000h	24.8.8/1080
209_4024	Interrupt Masks 2 Register (FLEXCAN2_IMASK2)	32	R/W	0000_0000h	24.8.9/1084
209_4028	Interrupt Masks 1 Register (FLEXCAN2_IMASK1)	32	R/W	0000_0000h	24.8.10/ 1084
209_402C	Interrupt Flags 2 Register (FLEXCAN2_IFLAG2)	32	R/W	0000_0000h	24.8.11/ 1085

Table continues on the next page...

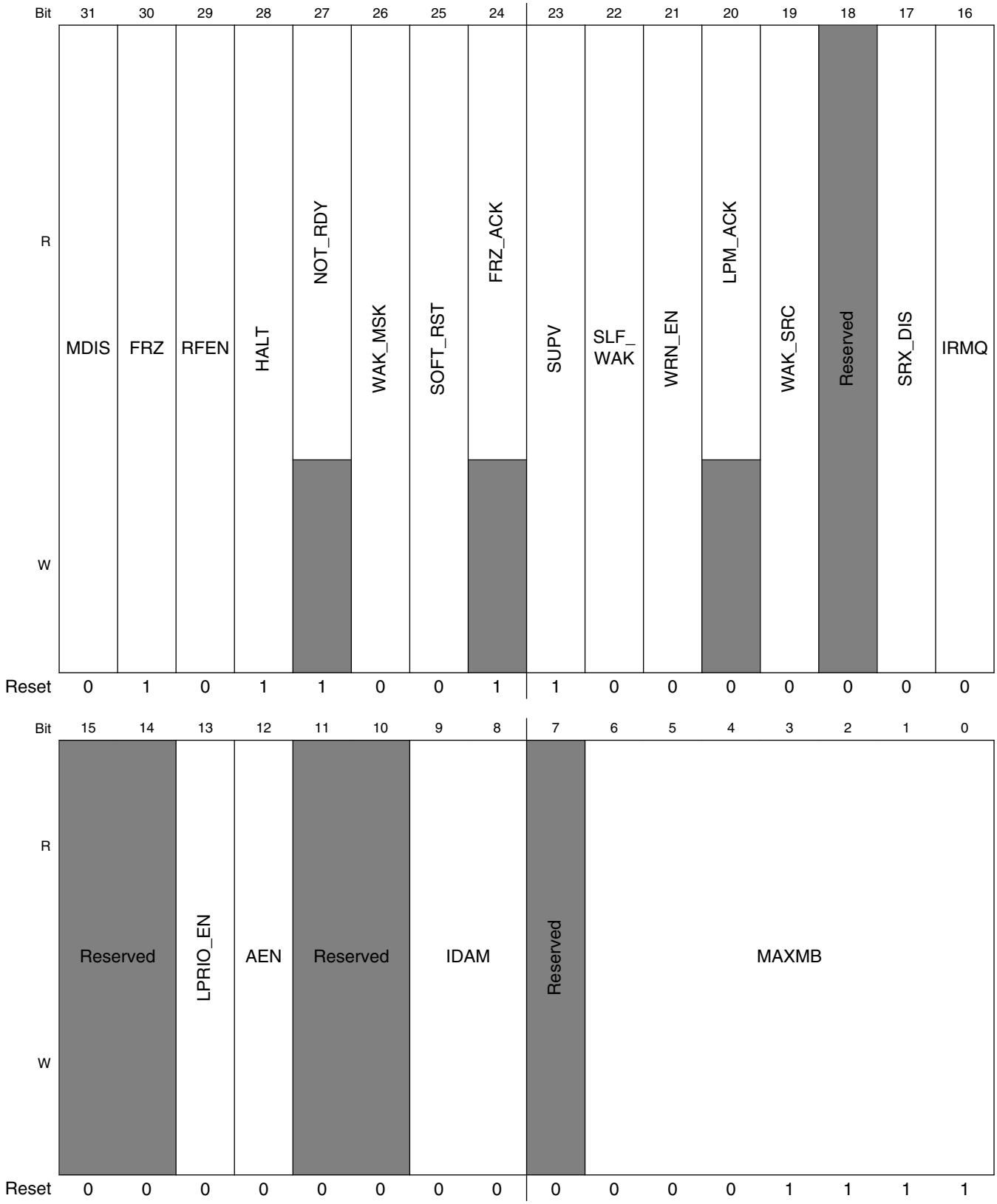
FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
209_4030	Interrupt Flags 1 Register (FLEXCAN2_IFLAG1)	32	R/W	0000_0000h	24.8.12/ 1085
209_4034	Control 2 Register (FLEXCAN2_CTRL2)	32	R/W	0000_0000h	24.8.13/ 1087
209_4038	Error and Status 2 Register (FLEXCAN2_ESR2)	32	R	0000_0000h	24.8.14/ 1093
209_4044	CRC Register (FLEXCAN2_CRCR)	32	R	0000_0000h	24.8.15/ 1095
209_4048	Rx FIFO Global Mask Register (FLEXCAN2_RXFGMASK)	32	R/W	FFFF_FFFFh	24.8.16/ 1096
209_404C	Rx FIFO Information Register (FLEXCAN2_RXFIR)	32	R	0000_0000h	24.8.17/ 1097
209_4880	Rx Individual Mask Registers (FLEXCAN2_RXIMR0_RXIMR63)	32	R/W	0000_0000h	24.8.18/ 1098
209_49E0	Glitch Filter Width Registers (FLEXCAN2_GFWR)	32	R/W	0000_007Fh	24.8.19/ 1098

24.8.1 Module Configuration Register (FLEXCANx_MCR)

This register defines global system configurations, such as the module operation mode (e.g., low power) and maximum message buffer configuration.

Address: Base address + 0h offset



FLEXCANx_MCR field descriptions

Field	Description
31 MDIS	This bit controls whether FLEXCAN is enabled or not. When disabled, FLEXCAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. This is the only bit in MCR not affected by soft reset. See Module Disable Mode for more information. 1 Disable the FLEXCAN module 0 Enable the FLEXCAN module
30 FRZ	The FRZ bit specifies the FLEXCAN behavior when the HALT bit in the MCR Register is set or when Debug Mode is requested at ARM level. When FRZ is asserted, FLEXCAN is enabled to enter Freeze Mode. Negation of this bit field causes FLEXCAN to exit from Freeze Mode. 1 Enabled to enter Freeze Mode 0 Not enabled to enter Freeze Mode
29 RFEN	This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements.RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 24-18 (see Arbitration and Matching Timing).This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FIFO enabled 0 FIFO not enabled
28 HALT	Assertion of this bit puts the FLEXCAN module into Freeze Mode. The ARM should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FLEXCAN before this bit is cleared. Freeze Mode can not be entered while FLEXCAN is in any of the low power modes.See Freeze Mode for more information 1 Enters Freeze Mode if the FRZ bit is asserted. 0 No Freeze Mode request.
27 NOT_RDY	This read-only bit indicates that FLEXCAN is either in Disable Mode, Stop Mode or Freeze Mode. It is negated once FLEXCAN has exited these modes. 1 FLEXCAN module is either in Disable Mode, Stop Mode or Freeze Mode 0 FLEXCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode
26 WAK_MSK	This bit enables the Wake Up Interrupt generation. 1 Wake Up Interrupt is enabled 0 Wake Up Interrupt is disabled
25 SOFT_RST	When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR1, ESR2, IMASK1, IMASK2, IFLAG1, IFLAG2 and CRCR. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL1, CTRL2, RXIMR0_RXIMR63, RXGMASK, RX14MASK, RX15MASK, RXFGMASK, RXFIR and all Message Buffers The SOFT_RST bit can be asserted directly by the ARM when it writes to the MCR Register. It may take some time to fully propagate its effect. The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied.

Table continues on the next page...

FLEXCANx_MCR field descriptions (continued)

Field	Description
	1 Reset the registers 0 No reset request
24 FRZ_ACK	This read-only bit indicates that FLEXCAN is in Freeze Mode and its prescaler is stopped. The Freeze Mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FLEXCAN has actually entered Freeze Mode. If Freeze Mode request is negated, then this bit is negated once the FLEXCAN prescaler is running again. If Freeze Mode is requested while FLEXCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See Freeze Mode for more information 1 FLEXCAN in Freeze Mode, prescaler stopped 0 FLEXCAN not in Freeze Mode, prescaler running
23 SUPV	This bit configures some of the FLEXCAN registers to be either in Supervisor or User Mode. Reset value of this bit is '1', so the affected registers start with Supervisor access allowance only. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FlexCAN is in Supervisor Mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location 0 FlexCAN is in User Mode. Affected registers allow both Supervisor and Unrestricted accesses
22 SLF_WAK	This bit enables the Self Wake Up feature when FLEXCAN is in Stop Mode. If this bit had been asserted by the time FLEXCAN entered Stop Mode, then FLEXCAN will look for a recessive to dominant transition on the bus during these modes. If a transition from recessive to dominant is detected during Stop Mode, then FLEXCAN generates, if enabled to do so, a Wake Up interrupt to the ARM so that it can resume the clocks globally and FlexCAN can request to resume the clocks. This bit can not be written while the module is in Stop Mode. 1 FLEXCAN Self Wake Up feature is enabled 0 FLEXCAN Self Wake Up feature is disabled
21 WRN_EN	When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 TWRN_INT and RWRN_INT bits are set when the respective error counter transition from <96 to ≥ 96. 0 TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters.
20 LPM_ACK	This read-only bit indicates that FLEXCAN is either in Disable Mode or Stop Mode. Either of these low power modes can not be entered until all current transmission or reception processes have finished, so the ARM can poll the LPM_ACK bit to know when FLEXCAN has actually entered low power mode. See Module Disable Mode, and Stop Mode for more information 1 FLEXCAN is either in Disable Mode, or Stop mode 0 FLEXCAN not in any of the low power modes
19 WAK_SRC	This bit defines whether the integrated low-pass filter is applied to protect the FLEXCAN_RX input from spurious wake up. See Stop Mode for more information. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FLEXCAN uses the filtered FLEXCAN_RX input to detect recessive to dominant edges on the CAN bus 0 FLEXCAN uses the unfiltered FLEXCAN_RX input to detect recessive to dominant edges on the CAN bus.
18 -	This field is reserved. Reserved

Table continues on the next page...

FLEXCANx_MCR field descriptions (continued)

Field	Description
17 SRX_DIS	This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Self reception disabled 0 Self reception enabled
16 IRMQ	This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK and RX15MASK, RXFGMASK. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Individual Rx masking and queue feature are enabled. 0 Individual Rx masking and queue feature are disabled.For backward compatibility, the reading of C/S word locks the MB even if it is EMPTY.
15–14 -	This field is reserved. Reserved
13 LPRIO_EN	This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to extend the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Local Priority enabled 0 Local Priority disabled
12 AEN	This bit is supplied for backwards compatibility reasons. When asserted, it enables the Tx abort feature. This feature guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can only be written in Freeze mode as it is blocked by hardware in other modes.Write Abort code into Rx Mailboxes can cause unpredictable results when the MCR[AEN] is asserted. 1 Abort enabled 0 Abort disabled
11–10 -	This field is reserved. Reserved
9–8 IDAM	This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown below. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Rx FIFO Structure. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 00 Format A One full ID (standard or extended) per ID filter Table element. 01 Format B Two full standard IDs or two partial 14-bit extended IDs per ID filter Table element. 10 Format C Four partial 8-bit IDs (standard or extended) per ID filter Table element. 11 Format D All frames rejected.
7 -	This field is reserved. Reserved
MAXMB	This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field can only be written in Freeze Mode as it is blocked by hardware in other modes Number of the last MB = MAXMB.

Table continues on the next page...

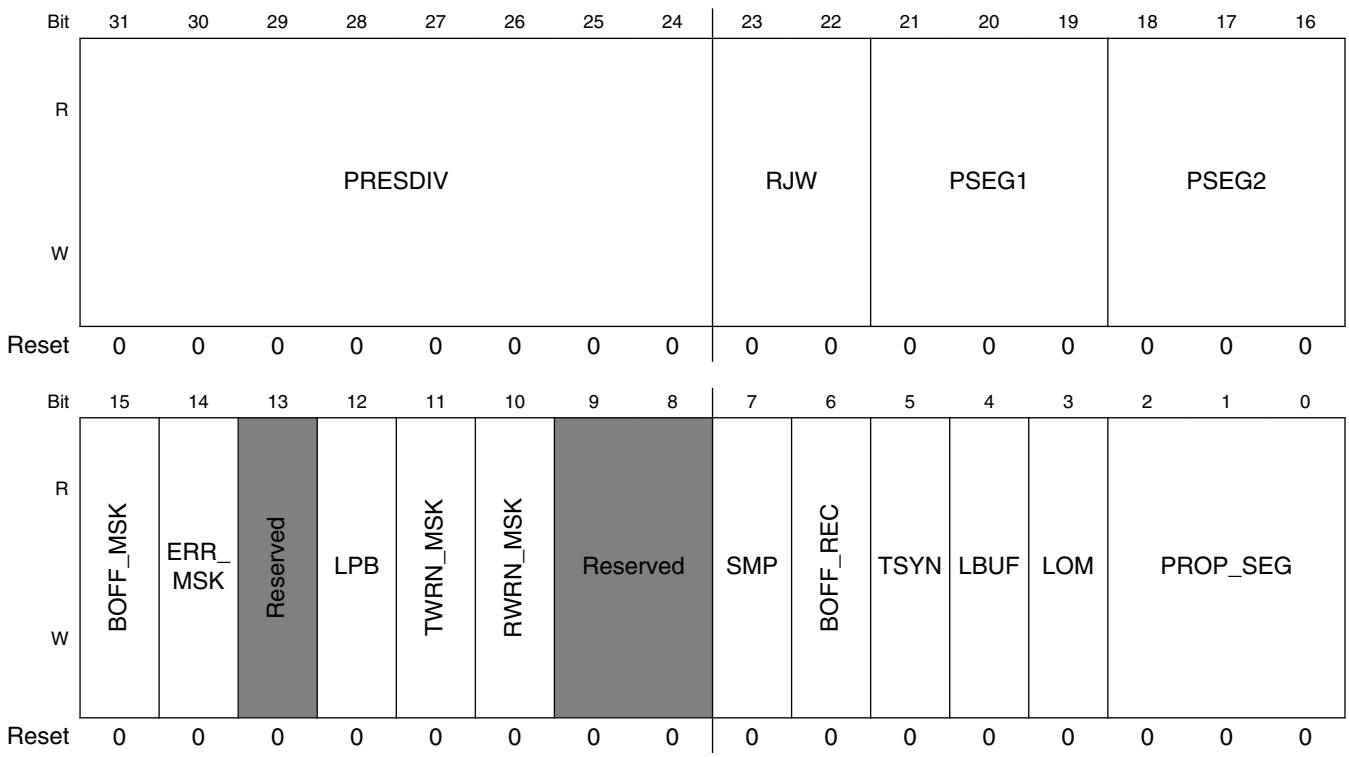
FLEXCANx_MCR field descriptions (continued)

Field	Description
	NOTE: Additionally, the value of MAXMB must encompass the FIFO size defined by CTRL2[RFFN] MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 24-18 (see Arbitration and Matching Timing).

24.8.2 Control 1 Register (FLEXCANx_CTRL1)

This register is defined for specific FLEXCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back Mode, Listen Only Mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: Base address + 4h offset



FLEXCANx_CTRL1 field descriptions

Field	Description
31–24 PRESDIV	This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256.For more information refer to Protocol Timing. This field can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

FLEXCANx_CTRL1 field descriptions (continued)

Field	Description
	Sclock frequency = CPI clock frequency / (PRESDIV+1)
23–22 RJW	This 2-bit field defines the maximum number of time quanta[1] that a bit time can be changed by one re-synchronization. The valid programmable values are 0-3. This field can only be written in Freeze mode as it is blocked by hardware in other modes Resync Jump Width = RJW + 1.
21–19 PSEG1	This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta.
18–16 PSEG2	This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta.
15 BOFF_MSK	This bit provides a mask for the Bus Off Interrupt. 1 Bus Off interrupt enabled 0 Bus Off interrupt disabled
14 ERR_MSK	This bit provides a mask for the Error Interrupt. 1 Error interrupt enabled 0 Error interrupt disabled
13 -	This field is reserved. Reserved
12 LPB	This bit configures FlexCAN to operate in Loop-Back Mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The FLEXCAN_RX input pin is ignored and the FLEXCAN_TX output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Loop Back enabled 0 Loop Back disabled
11 TWRN_MSK	This bit provides a mask for the Tx Warning Interrupt associated with the TWRN_INT flag in the Error and Status Register. This bit is read as zero when MCR[WRN_EN] bit is negated. This bit can only be written if MCR[WRN_EN] bit is asserted. 1 Tx Warning Interrupt enabled 0 Tx Warning Interrupt disabled
10 RWRN_MSK	This bit provides a mask for the Rx Warning Interrupt associated with the RWRN_INT flag in the Error and Status Register. This bit is read as zero when MCR[WRN_EN] bit is negated. This bit can only be written if MCR[WRN_EN] bit is asserted. 1 Rx Warning Interrupt enabled 0 Rx Warning Interrupt disabled
9–8 -	This field is reserved. Reserved
7 SMP	This bit defines the sampling mode of CAN bits at the FLEXCAN_RX. This bit can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

FLEXCANx_CTRL1 field descriptions (continued)

Field	Description
	1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used 0 Just one sample is used to determine the bit value
6 BOFF_REC	This bit defines how FLEXCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FLEXCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 1 Automatic recovering from Bus Off state disabled 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B
5 TSYN	This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FLEXCAN stations with a special "SYNC" message (i.e., global network time). If the RFEN bit in MCR is set (FIFO enabled), the first available Mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Timer Sync feature enabled 0 Timer Sync feature disabled
4 LBUF	This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIO_EN bit does not affect the priority arbitration.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Lowest number buffer is transmitted first 0 Buffer with highest priority is transmitted first
3 LOM	This bit configures FLEXCAN to operate in Listen Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. Listen-Only Mode acknowledgement can be obtained by the state of ESR1[FLT_CONF] field which is Passive Error when Listen-Only Mode is entered. There can be some delay between the Listen-Only Mode request and acknowledge. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FLEXCAN module operates in Listen Only Mode 0 Listen Only Mode is deactivated
PROP_SEG	This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Propagation Segment Time = (PROPSEG + 1) * Time-Quanta. Time-Quantum = one Sclock period.

1. One time quantum is equal to the Sclock period.

24.8.3 Free Running Timer Register (FLEXCANx_TIMER)

This register represents a 16-bit free running counter that can be read and written by the ARM. The timer starts from $0000 after Reset, counts linearly to $FFFF, and wraps around.

The timer is clocked by the FLEXCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During Freeze Mode, disable, and stop mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CTRL1[TSYN] is asserted the Timer is reset whenever a message is received in the first available Mailbox, according to CTRL2[RFFN] setting.

ARM can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure. For additional details, refer to Message Buffer Lock Mechanism.

Address: Base address + 8h offset

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

FLEXCANx_TIMER field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
TIMER	TIMER

24.8.4 Rx Mailboxes Global Mask Register (FLEXCANx_RXMGMASK)

RXMGMASK is provided for legacy support. Asserting the MCR[IRMQ] bit causes the RXMGMASK Register to have no effect on the module operation.

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

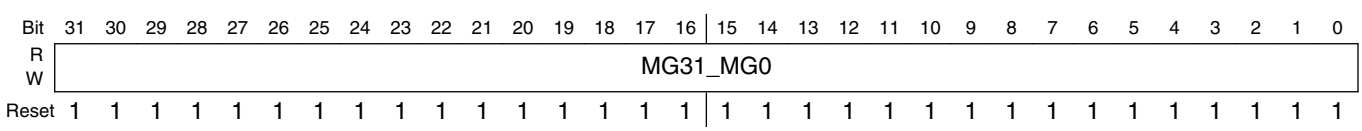
This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Table 24-19. Rx Mailboxes Global Mask usage

SMB[RTR][1]	CTRL2[RRS]	CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	reserved
0	-	0	- Note[2]	- Note[3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If CTRL2[EACEN] bit is negated the RTR bit of Mailbox is never compared with the RTR bit of the Incoming Frame (Rx SMB[RTR]).
3. If CTRL2[EACEN] bit is negated the IDE bit of Mailbox is always compared with the IDE bit of the Incoming Frame (Rx SMB[IDE]).

Address: Base address + 10h offset



FLEXCANx_RXMGMASK field descriptions

Field	Description
MG31_MG0	These bits mask the Mailbox filter bits as shown in the figure above. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE which are located in the Control and Status word of the Mailbox. Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK) shows in detail which MG bits mask each Mailbox filter field. 1 The corresponding bit in the filter is checked against the one received 0 the corresponding bit in the filter is "don't care"

24.8.5 Rx Buffer 14 Mask Register (FLEXCANx_RX14MASK)

RX14MASK is provided for legacy support, asserting the MCR[IRMQ] bit causes the RX14MASK to have no effect on the module operation.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX14M31_RX14M0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RX14MASK field descriptions

Field	Description
RX14M31_ RX14M0	These bits mask Mailbox 14 filter bits in the same fashion as RXMGMASK masks other Mailboxes filters (see Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)) 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

24.8.6 Rx Buffer 15 Mask Register (FLEXCANx_RX15MASK)

RX15MASK is provided for legacy support, asserting the MCR[IRMQ] bit causes the RX15MASK Register to have no effect on the module operation.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX15M31_RX15M0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RX15MASK field descriptions

Field	Description
RX15M31_ RX15M0	These bits mask Mailbox 15 filter bits in the same fashion as RXMGMASK masks other Mailboxes filters (see Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)). 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

24.8.7 Error Counter Register (FLEXCANx_ECR)

This register has 2 8-bit fields reflecting the value of two FLEXCAN error counters: Transmit Error Counter (Tx_Err_Counter field) and Receive Error Counter (Rx_Err_Counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FLEXCAN module. Both counters are read only except in Freeze Mode, where they can be written by the ARM.

FLEXCAN responds to any bus state as described in the protocol, e.g. transmit 'Error Active' or 'Error Passive' flag, delay its transmission start time ('Error Passive') and avoid any influence on the bus when in 'Bus Off' state. The following are the basic rules for FLEXCAN bus state transitions.

- If the value of Tx_Err_Counter or Rx_Err_Counter increases to be greater than or equal to 128, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Passive' state.
- If the FLEXCAN state is 'Error Passive', and either Tx_Err_Counter or Rx_Err_Counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Active' state.
- If the value of Tx_Err_Counter increases to be greater than 255, the FLT_CONF field in the Error and Status Register is updated to reflect 'Bus Off' state, and an interrupt may be issued. The value of Tx_Err_Counter is then reset to zero.
- If FLEXCAN is in 'Bus Off' state, then Tx_Err_Counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Err_Counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, the FLT_CONF field in the Error and Status Register is updated to be 'Error Active' and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the Tx_Err_Counter value.
- If during system start-up, only one node is operating, then its Tx_Err_Counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the Error and Status Register). After the transition to 'Error Passive' state, the Tx_Err_Counter does not increment anymore by acknowledge errors. Therefore the device never goes to the 'Bus Off' state.
- If the Rx_Err_Counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next

successful message reception, the counter is set to a value between 119 and 127 to resume to 'Error Active' state.

Address: Base address + 1Ch offset

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

FLEXCANx_ECR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 Rx_Err_Counter	Rx_Err_Counter
Tx_Err_Counter	Tx_Err_Counter

24.8.8 Error and Status 1 Register (FLEXCANx_ESR1)

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the ARM.

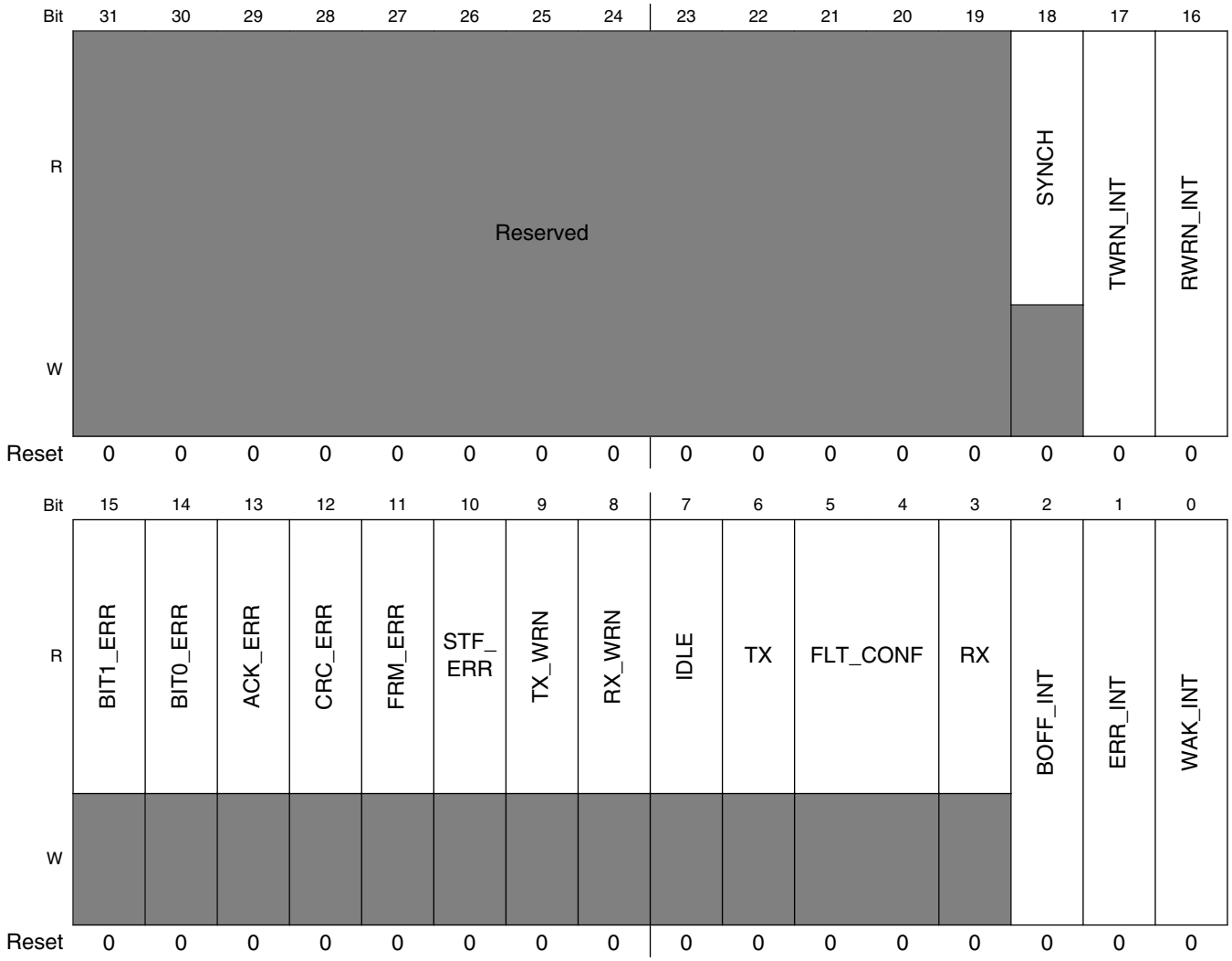
The ARM read action clears bits 15-10, therefore the reported *error conditions*(bits 15-10) are those that occurred since the last time the ARM read this register. Bits 9-3 are status bits.

Some bits in this register are read-only and some are not.

Table 24-20. FlexCAN State

SYNCH	IDLE	TX	RX	FlexCAN state
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving
other combinations				Reserved

Address: Base address + 20h offset



FLEXCANx_ESR1 field descriptions

Field	Description
31–19 -	This field is reserved. Reserved
18 SYNCH	This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. Refer to Table 24-20 1 FlexCAN is synchronized to the CAN bus 0 FlexCAN is not synchronized to the CAN bus
17 TWRN_INT	If the WRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transition from '0' to '1', meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRN_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. When WRN_EN is negated, this flag is masked. ARM must clear this flag before disabling the bit. Otherwise it will be set when the WRN_EN is set again. Writing '0' has no effect. This flag is not generated during "Bus Off" state. This bit is not updated during Freeze mode. 1 The Tx error counter transition from < 96 to >= 96 0 No such occurrence

Table continues on the next page...

FLEXCANx_ESR1 field descriptions (continued)

Field	Description
16 RWRN_INT	If the WRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transition from '0' to '1', meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRN_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. When WRN_EN is negated, this flag is masked. ARM must clear this flag before disabling the bit. Otherwise it will be set when the WRN_EN is set again. Writing '0' has no effect. This bit is not updated during Freeze mode. 1 The Rx error counter transition from < 96 to >= 96 0 No such occurrence
15 BIT1_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 1 At least one bit sent as recessive is received as dominant 0 No such occurrence
14 BIT0_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 1 At least one bit sent as dominant is received as recessive 0 No such occurrence
13 ACK_ERR	This bit indicates that an Acknowledge Error has been detected by the transmitter node, i.e., a dominant bit has not been detected during the ACK SLOT. 1 An ACK error occurred since last read of this register 0 No such occurrence
12 CRC_ERR	This bit indicates that a CRC Error has been detected by the receiver node, i.e., the calculated CRC is different from the received. 1 A CRC error occurred since last read of this register. 0 No such occurrence
11 FRM_ERR	This bit indicates that a Form Error has been detected by the receiver node, i.e., a fixed-form bit field contains at least one illegal bit. 1 A Form Error occurred since last read of this register 0 No such occurrence
10 STF_ERR	This bit indicates that a Stuffing Error has been detected. 1 A Stuffing Error occurred since last read of this register. 0 No such occurrence.
9 TX_WRN	This bit indicates when repetitive errors are occurring during message transmission. 1 TX_Err_Counter ≥ 96 0 No such occurrence
8 RX_WRN	This bit indicates when repetitive errors are occurring during message reception. 1 Rx_Err_Counter ≥ 96 0 No such occurrence
7 IDLE	This bit indicates when CAN bus is in IDLE state.Refer to Table 24-20.

Table continues on the next page...

FLEXCANx_ESR1 field descriptions (continued)

Field	Description
	1 CAN bus is now IDLE 0 No such occurrence
6 TX	This bit indicates if FLEXCAN is transmitting a message.Refer to Table 24-20. 1 FLEXCAN is transmitting a message 0 FLEXCAN is receiving a message
5–4 FLT_CONF	If the LOM bit in the Control Register is asserted, after some delay that depends on the CAN bit timing the FLT_CONF field will indicate "Error Passive". The very same delay affects the way how FLT_CONF reflects an update to ECR register by the ARM. It may be necessary up to one CAN bit time to get them coherent again. Since the Control Register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted. This 2-bit field indicates the Confinement State of the FLEXCAN module, as shown in below: 00 Error Active 01 Error Passive 1x Bus off
3 RX	This bit indicates if FlexCAN is receiving a message. Refer to Table 24-20. 1 FLEXCAN is transmitting a message 0 FLEXCAN is receiving a message
2 BOFF_INT	This bit is set when FLEXCAN enters 'Bus Off' state. If the corresponding mask bit in the Control Register (BOFF_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. Writing '0' has no effect. 1 FLEXCAN module entered 'Bus Off' state 0 No such occurrence
1 ERR_INT	This bit indicates that at least one of the Error Bits (bits 15-10) is set. If the corresponding mask bit in the Control Register (ERR_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'.Writing '0' has no effect. 1 Indicates setting of any Error Bit in the Error and Status Register 0 No such occurrence
0 WAK_INT	When FLEXCAN is Stop Mode and a recessive to dominant transition is detected on the CAN bus and if the WAK_MSK bit in the MCR Register is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. When SLF_WAK is negated, this flag is masked. ARM must clear this flag before disabling the bit. Otherwise it will be set when the SLF_WAK is set again. Writing '0' has no effect 1 Indicates a recessive to dominant transition received on the CAN bus when the FLEXCAN module is in Stop Mode 0 No such occurrence

24.8.9 Interrupt Masks 2 Register (FLEXCANx_IMASK2)

This register allows any number of a range of 32 Message Buffer Interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (i.e. when the corresponding IFLAG2 bit is set).

Address: Base address + 24h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63M_BUF32M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IMASK2 field descriptions

Field	Description
BUF63M_ BUF32M	Each bit enables or disables the respective FLEXCAN Message Buffer (MB32 to MB63) Interrupt. Setting or clearing a bit in the IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 1 The corresponding buffer Interrupt is enabled 0 The corresponding buffer Interrupt is disabled

24.8.10 Interrupt Masks 1 Register (FLEXCANx_IMASK1)

This register allows to enable or disable any number of a range of 32 Message Buffer Interrupts. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (i.e., when the corresponding IFLAG1 bit is set).

Address: Base address + 28h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF31M_BUF0M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IMASK1 field descriptions

Field	Description
BUF31M_BUF0M	Each bit enables or disables the respective FLEXCAN Message Buffer (MB0 to MB31) Interrupt. Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set

FLEXCANx_IMASK1 field descriptions (continued)

Field	Description
	1 The corresponding buffer Interrupt is enabled 0 The corresponding buffer Interrupt is disabled

24.8.11 Interrupt Flags 2 Register (FLEXCANx_IFLAG2)

This register defines the flags for 32 Message Buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing it to '1'. Writing '0' has no effect.Before updating MCR[MAXMB] field, ARM must treat the IFLAG2 bits which MB value is greater than the MCR[MAXMB] to be updated, otherwise they will keep set and be inconsistent with the amount of MBs available.

Address: Base address + 2Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63I_BUF32I																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IFLAG2 field descriptions

Field	Description
BUF63I_BUF32I	Each bit flags the respective FLEXCAN Message Buffer (MB32 to MB63) interrupt. 1 The corresponding buffer has successfully completed transmission or reception 0 No such occurrence

24.8.12 Interrupt Flags 1 Register (FLEXCANx_IFLAG1)

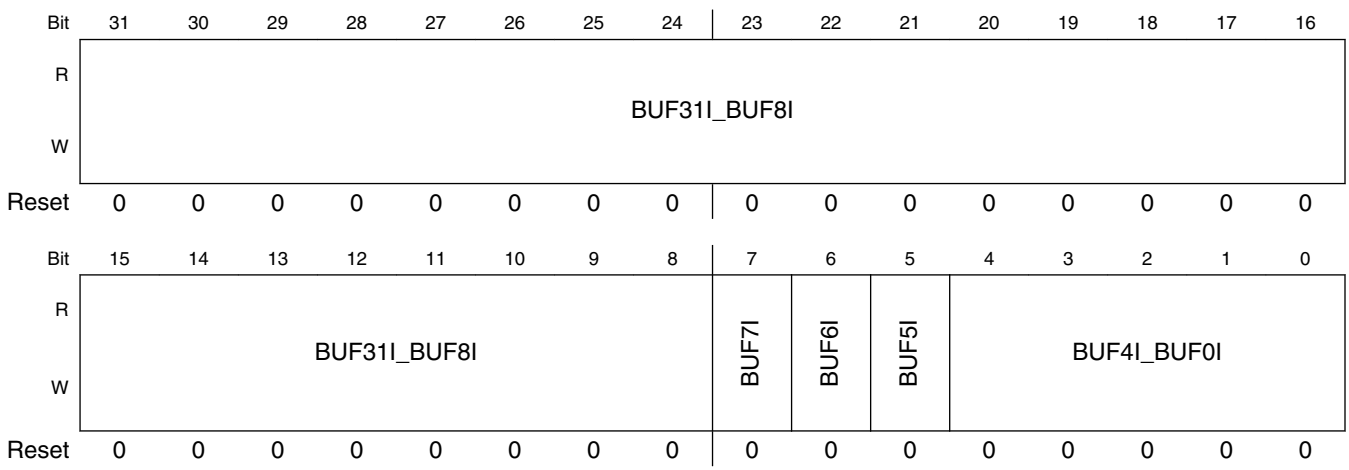
This register defines the flags for 32 Message Buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The Interrupt flag must be cleared by writing it to '1'. Writing '0' has no effect.

When the RFEN bit in the MCR is set (Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) is changed to support the FIFO operation. BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, while BUF4I to BUF0I are not used. Before enabling the RFEN, ARM must service the IFLAGS asserted

in the Rx FIFO region (see Rx FIFO). Otherwise, these IFLAGS will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the RFEN is negated, the FIFO flags must be cleared. The same care must be taken when a RFFN value is selected extending Rx FIFO filters beyond MB7 (see Control 2 Register (FLEXCAN_CTRL2)). For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAGS must be cleared.

Before updating MCR[MAXMB] field, ARM must service the IFLAG1 which MB value is greater than the MCR[MAXMB] to be updated, otherwise they will keep set and be inconsistent with the amount of MBs available.

Address: Base address + 30h offset



FLEXCANx_IFLAG1 field descriptions

Field	Description
31–8 BUF31I_BUF8I	Each bit flags the respective FLEXCAN Message Buffer (MB8 to MB31) interrupt. 1 The corresponding MB has successfully completed transmission or reception 0 No such occurrence
7 BUF7I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB7. If the MCR[RFEN] bit is asserted, this flag indicates that a message was lost because Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes. 1 MB7 completed transmission/reception or FIFO overflow 0 No such occurrence
6 BUF6I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB6. If the MCR[RFEN] bit is asserted, this flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4 it will not assert again until the number of unread messages within the Rx FIFO is decreased to equal or less than 4. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes.

Table continues on the next page...

FLEXCANx_IFLAG1 field descriptions (continued)

Field	Description
	1 MB6 completed transmission/reception or FIFO almost full 0 No such occurrence
5 BUF5I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB5. If the Rx FIFO is enabled, this flag indicates that at least one frame is available to be read from the Rx FIFO. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes. 1 MB5 completed transmission/reception or frames available in the FIFO 0 No such occurrence
BUF4I_BUF0I	If the Rx FIFO is not enabled, these bits flag the interrupts for MB0 to MB4. If the Rx FIFO is enabled, these flags are not used and must be considered as reserved locations. These flags are cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes. 1 Corresponding MB completed transmission/reception 0 No such occurrence

24.8.13 Control 2 Register (FLEXCANx_CTRL2)

This register contains control bits for CAN errors, FIFO features and mode selection.

Table 24-21. Rx FIFO Filters

RFFN[3:0]	Number of Rx FIFO filters	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes[1]	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks[2]	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask[2]
0x0	8	MB 0-7	MB 8-63	Elements 0-7	none
0x1	16	MB 0-9	MB 10-63	Elements 0-9	Elements 10-15
0x2	24	MB 0-11	MB 12-63	Elements 0-11	Elements 12-23
0x3	32	MB 0-13	MB 14-63	Elements 0-13	Elements 14-31
0x4	40	MB 0-15	MB 16-63	Elements 0-15	Elements 16-39
0x5	48	MB 0-17	MB 18-63	Elements 0-17	Elements 18-47
0x6	56	MB 0-19	MB 20-63	Elements 0-19	Elements 20-55
0x7	64	MB 0-21	MB 22-63	Elements 0-21	Elements 22-63
0x8	72	MB 0-23	MB 24-63	Elements 0-23	Elements 24-71
0x9	80	MB 0-25	MB 26-63	Elements 0-25	Elements 26-79
0xA	88	MB 0-27	MB 28-63	Elements 0-27	Elements 28-87
0xB	96	MB 0-29	MB 30-63	Elements 0-29	Elements 30-95
0xC	104	MB 0-31	MB 32-63	Elements 0-31	Elements 32-103
0xD	112	MB 0-33	MB 34-63	Elements 0-31	Elements 32-111
0xE	120	MB 0-35	MB 36-63	Elements 0-31	Elements 32-119
0xF	128	MB 0-37	MB 38-63	Elements 0-31	Elements 32-127

1. The number of the last remaining available mailboxes is defined by the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available.

Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows:

$$(SETUP_MB - 6) \times 4$$

where SETUP_MB is MAXMB.

The number of remaining Mailboxes available will be:

$$SETUP_MB - 8 - (RFFN \times 2)$$

If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value, the exceeding ones will not be functional.Unshaded regions in Table 24-22 indicate the valid combinations of MAXMB, RFEN and RFFN, shaded regions are not functional.

Table 24-22. Valid Combinations of MAXMB, RFEN and RFFN

RFFN	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RFEN	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MAXMB																	
0 - 6																	
7 - 8																	
9 - 10																	
11 - 12																	
13 -14																	
15 - 16																	
17 -18																	
19 - 20																	
21 - 22																	

Table continues on the next page...

Table 24-22. Valid Combinations of MAXMB, RFEN and RFFN (continued)

RFFN	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RFEN	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MAXMB																	
23 - 24																	
25 - 26																	
27 - 28																	
29 -30																	
31 - 32																	
33 - 34																	
35 - 36																	
37 - 63																	

Address: Base address + 34h offset



FLEXCANx_CTRL2 field descriptions

Field	Description
31 -	must be written as 0
30–29 -	This field is reserved. Reserved

Table continues on the next page...

FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
28 WRMFRZ	Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode. 1 Enable unrestricted write access to FlexCAN memory 0 Keep the write access restricted in some regions of FlexCAN memory
27–24 RFFN	This 4-bit field defines the number of Rx FIFO filters according to Table 24-21. The maximum selectable number of filters is determined by the ARM. This field can only be written in Freeze mode as it is blocked by hardware in other modes. RFFN defines a number of Message Buffers occupied by Rx FIFO and ID Filter (see Table 24-21) that **may not exceed** the number of available Mailboxes present in module, defined by MCR[MAXMB].Default RFFN value is 0x0, which leads to a total of 8 Rx FIFO filters, occupies the first 8 Message Buffers (MB 0-7) and makes available the next Message Buffers (MB 8-63) for Mailboxes. As a second example, when RFFN is set to 0xD, there will be 112 Rx FIFO filters, located in MB 0-33, and MB 34-63 are available for Mailboxes. Notice that, in this case, individual masks (RXIMR) will just cover Rx FIFO filters in 0-31 range, and filters 32-111 will use RXFGMASK. In case of reducing the number of last Message Buffers, MCR[MAXMB] (see Module Configuration Register (FLEXCAN_MCR)) can be adjusted by the application to minimum of 33, in order to give room to the Rx FIFO and its ID Filter Table defined by RFFN. On the contrary, if the application sets MCR[MAXMB] to 16, for instance, the maximum RFFN is limited to 0x4. RFFN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 24-18 (see Arbitration and Matching Timing).
23–19 TASD	This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field is useful to optimize the transmit performance based on factors such as: peripheral/serial clock ratio, CAN bit timing and number of MBs. The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and CAN baud rate and inversely proportional to the peripheral clock frequency. The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. Therefore, if there are few MBs and the system/serial clock ratio is high and the CAN baud rate is low then the arbitration can be delayed and vice-versa. If TASD is 0 then the arbitration start is not delayed, thus ARM has less time to configure a Tx MB for the next arbitration, but more time is reserved for arbitration. In the other hand, if TASD is 24 then ARM can configure a Tx MB later and less time is reserved for arbitration. If too little time is reserved for arbitration the FlexCAN may be not able to find winner MBs in time to compete with other nodes for the CAN bus. If the arbitration ends too much time before the first bit of Intermission field then there is a chance that ARM reconfigure some Tx MBs and the winner MB is not the best to be transmitted. The reset value is different on various platforms, according to their peripheral clock frequency, number of MBs and target CAN baud rate. The optimal configuration for TASD can be calculated as:

Table continues on the next page...

FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
	$$TASD = 25 - \left\{ \frac{f_{CANCLK} \times [MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)] \times 2}{f_{sys} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)} \right\}$$ where: f_{CANCLK} is the Protocol Engine (PE) Clock in Hz; PE clock is derrived from CAN_CLK_ROOT in CCM. See Clock Root Generator f_{SYS} is the peripheral clock in Hz; MAXMB is the value in CTRL1[MAXMB] field; RFEN is the value in CTRL1[RFEN] bit; RFFN is the value in CTRL2[RFFN] field; PSEG1 is the value in CTRL1[PSEG1] field; PSEG2 is the value in CTRL1[PSEG2] field; PROPSEG is the value in CTRL1[PROPSEG] field; PRESDIV is the value in CTRL1[PRESDIV] field. Please refer to Arbitration process and Protocol Timing for more details.
18 MRP	If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Matching starts from Mailboxes and continues on Rx FIFO 0 Matching starts from Rx FIFO and continues on Mailboxes
17 RRS	If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated. If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Remote Request Frame is stored 0 Remote Response Frame is generated
16 EACEN	This bit controls the comparison of IDE and RTR bits within Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Enables the comparison of both Rx Mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply. 0 Rx Mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits.
-	This field is reserved.

Table continues on the next page...

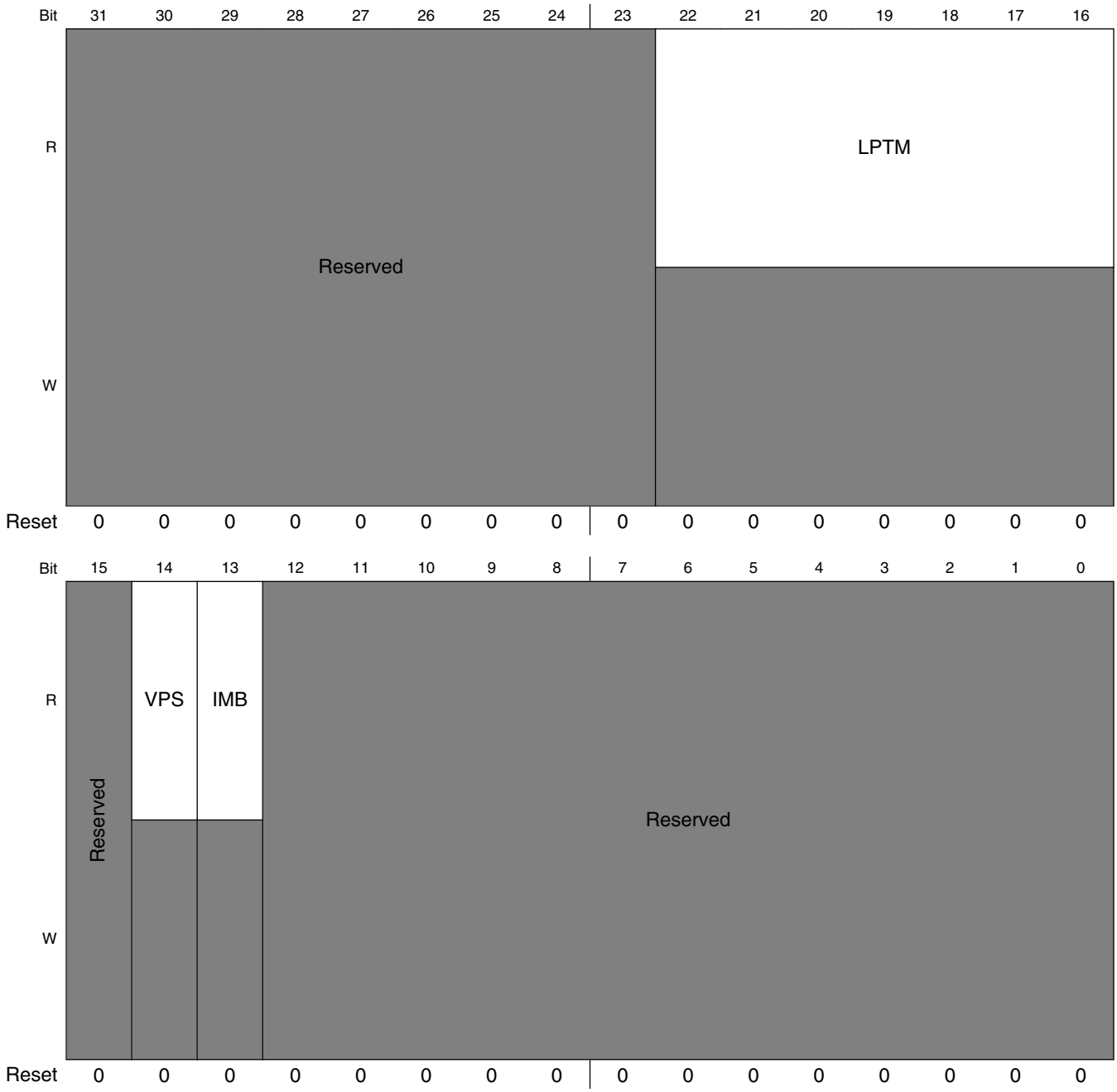
FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
	Reserved

24.8.14 Error and Status 2 Register (FLEXCANx_ESR2)

This register reflects various interrupt flags and some general status.

Address: Base address + 38h offset



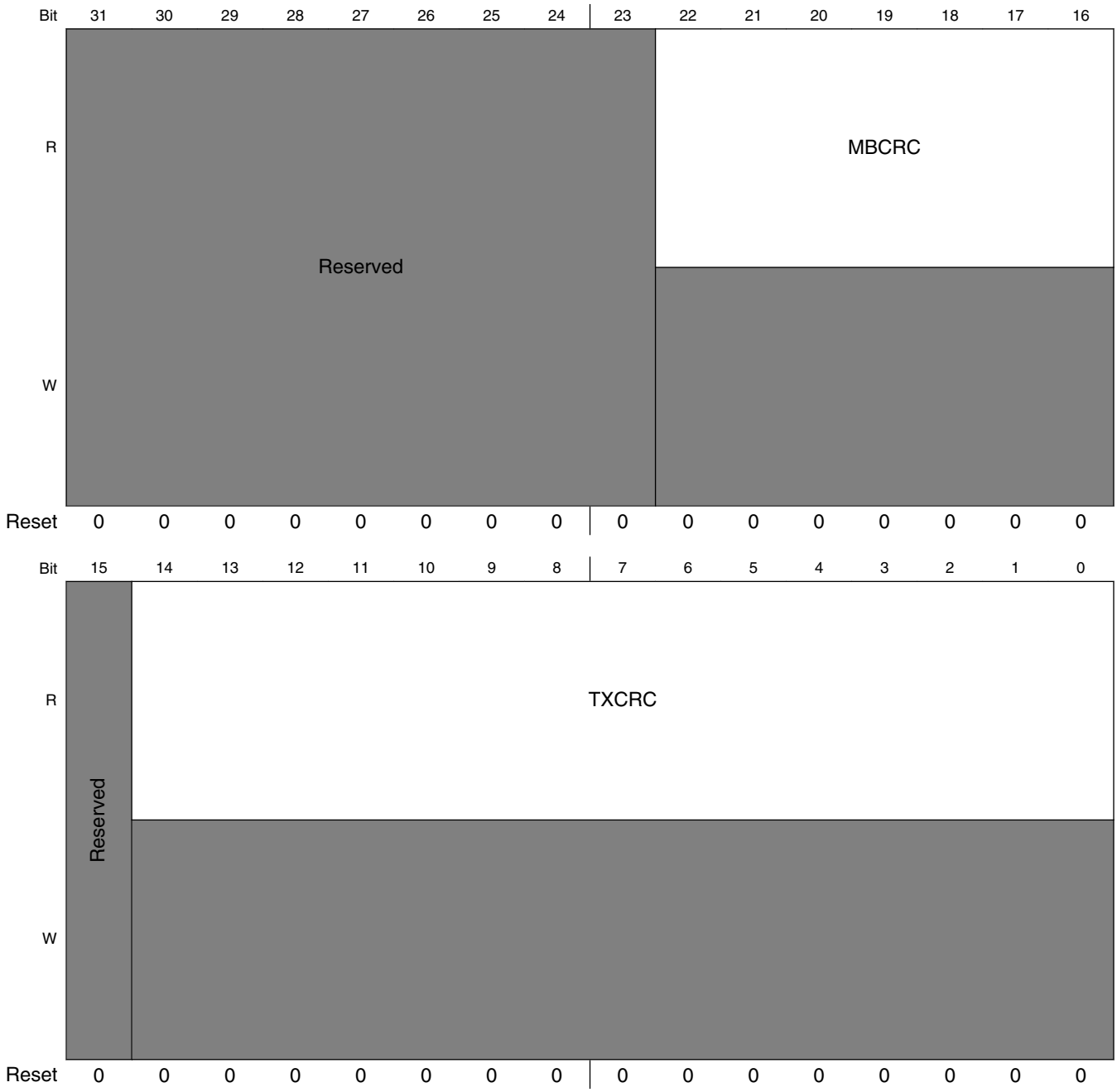
FLEXCANx_ESR2 field descriptions

Field	Description
31–23 -	This field is reserved. Reserved
22–16 LPTM	If ESR2[VPS] is asserted, his 7-bit field indicates the lowest number inactive Mailbox (refer to IMB bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CTRL1[LBUF] bit value. If CTRL1[LBUF] bit is negated then the Mailbox indicated is the one which has the greatest arbitration value (see Highest Mailbox priority first). If CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
15 -	This field is reserved. Reserved
14 VPS	This bit indicates whether IMB and LPTM contents are currently valid or not. VPS is asserted upon every complete Tx arbitration process unless the ARM writes to Control and Status word of a Mailbox that has already been scanned (i.e. it is behind Tx Arbitration Pointer) during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox which is being transmitted then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox.ESR2[VPS] is not affected by any ARM write into Control Status (C/S) of a MB which is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of a MB that is been transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked. 1 Contents of IMB and LPTM are valid 0 Contents of IMB and LPTM are invalid
13 IMB	If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases: (1) During arbitration, if a LPTM is found and it is inactive. (2) If IMB is not asserted and a frame is transmitted successfully. (3) This bit is cleared in all start of arbitration (see Arbitration process). LPTM mechanism have the following behavior: if a MB is successfully transmitted and ESR2[IMB]=0 (no inactive Mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM]. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox.
-	This field is reserved. Reserved

24.8.15 CRC Register (FLEXCANx_CRCR)

This register provides information about the CRC of transmitted messages

Address: Base address + 44h offset



FLEXCANx_CRCR field descriptions

Field	Description
31–23 -	This field is reserved. Reserved
22–16 MBCRC	This field indicates the number of the Mailbox corresponding to the value in TXCRC field.
15 -	This field is reserved. Reserved
TXCRC	This field indicates the CRC value of the last message transmitted. This field is updated at the same time the Tx Interrupt Flag is asserted.

24.8.16 Rx FIFO Global Mask Register (FLEXCANx_RXFGMASK)

If Rx FIFO is enabled RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze Mode as it is blocked by hardware in other modes.

Table 24-23. Rx FIFO Global Mask usage

Rx FIFO ID Filter Table Elements Format (MCR[IDAM])	Identifier Acceptance Filter fields					
	RTR	IDE	RXIDA	RXIDB	RXIDC	reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31] FGM[15]	FGM[30] FGM[14]	-	FGM[29:16] FGM[13:0] 1		-
C	-	-		-	FGM[31:24] FGM[23:16] FGM[15:8] FGM[7:0] 2	

1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

Address: Base address + 48h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FGM31_FGM0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RXFGMASK field descriptions

Field	Description
FGM31_FGM0	These bits mask the ID Filter Table elements bits in a perfect alignment.Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK) shows in detail which FGM bits mask each IDAF field. Clear this register has the effect of disabling the ID Filter. 1 The corresponding bit in the filter is checked 0 The corresponding bit in the filter is "don't care"

24.8.17 Rx FIFO Information Register (FLEXCANx_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which ARM accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. Refer to Rx FIFO to find instructions on reading this register.

Address: Base address + 4Ch offset

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

FLEXCANx_RXFIR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
IDHIT	This 9-bit field indicates which Identifier Acceptance Filter (see Rx FIFO Structure) was hit by the received message that is in the output of the Rx FIFO. (refer to Rx FIFO for details) If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the IFLAG[BUF5I] is asserted.

24.8.18 Rx Individual Mask Registers (FLEXCANx_RXIMR0_RXIMR63)

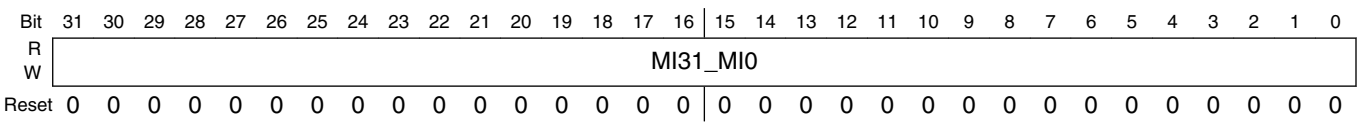
RXIMR are used as acceptance masks for ID filtering in Rx MBs and the Rx FIFO. If the Rx FIFO is not enabled, one mask register is provided for each available Mailbox, providing ID masking capability on a per Mailbox basis.

When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), up to 32 Rx Individual Mask Registers can apply to the Rx FIFO ID Filter Table elements on a one-to-one correspondence depending on CTRL2[RFFN] setting. Refer to Control 2 Register (FLEXCAN_CTRL2) for details.

RXIMR can only be written by the ARM while the module is in Freeze Mode, otherwise they are blocked by hardware.

The Individual Rx Mask Registers are not affected by reset and must be explicitly initialized prior to any reception.

Address: Base address + 880h offset



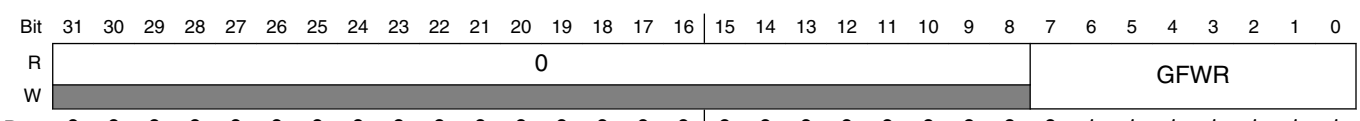
FLEXCANx_RXIMR0_RXIMR63 field descriptions

Field	Description
MI31_MI0	These bits mask both Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filter refer to Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK). For Rx FIFO ID Filter Table element refer to Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK). 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

24.8.19 Glitch Filter Width Registers (FLEXCANx_GFWR)

The Glitch Filter just takes effects when FLEXCAN enters the STOP mode.

Address: Base address + 9E0h offset



FLEXCANx_GFWR field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
GFWR	It determines the Glitch Filter Width. The width will be divided from Oscillator clock by GFWR values. By default, it is 5.33 µs when the oscillator is 24 MHz. Filter Pulse Width = [(GFWR FIELD + 1) x (1 / Osc. Frequency)]

Chapter 25
General Power Controller (GPC)

25.1 Overview

The General Power Control (GPC) block includes the sub-blocks listed here.

- DVFS - Dynamic Voltage & Frequency Scaling load tracking for CPU
- CPU Power Gating Control (PGC)

Each sub-block has its own IP registers.

GPC determines wake-up irq for exiting STOP mode (with or without CPU power gating).

Figure 25-1. GPC Block Diagram

25.2 Clocks

The table found here describes the clock sources for GPC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 25-1. GPC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
pgc_clk	ipg_clk_root	PGC peripheral clock
sys_clk	ipg_clk_root	Module clock

25.3 DVFS overview

The DVFS allows simple dynamic voltage frequency scaling.

The frequency of the core clock domain and the voltage of the core power domain can be changed on the fly while all modules (including the MCU) continue their normal operation. The frequency of the core clock domain can be changed by temporally switching to an alternate PLL clock, or by changing the post dividers division factors.

The DVFS load tracking block allows hardware tracking on the core load and generate an interrupt when a frequency change is requested.

NOTE

DVFS is a monitor that only provides an interrupt when CPU load exceeds the predefined value and does not send any request to make a change of voltage and frequency. This can be done by the user in the CPU interrupt routine or SDMA routine.

Figure 25-2. DVFS_core diagram

25.3.1 Features

The DVFS load tracking block includes the following features:

- Configurable include/exclude of input signals:
  - per-core ARM standby signal (idle / non-idle)
  - 16 general purpose bits (common for all 4 tracking modules)
    - Configurable weight to each bit.
- Configurable generated clocks and averaging time slicing (respond time).
- Configurable panic mode respond logic (for frequency up).
- Programmable buffer for last 4, 8, 12, or 16 load tracking samples of core 0 (only).

25.4 DVFS output event/interrupt configuration

Event/interrupt will be always high as long as LBFL is '1' and is not cleared by SW. Unless DVFEV (always event) is asserted. Then the event/interrupt will be toggled up and down at every toggle of div_3_clk.

25.4.1 Interrupts

DVFS generates an interrupt that indicates that frequency and voltage update is needed. The user has to read the FSVAIM bits in order to know which change needs to be done.

25.4.2 DVFS Change Request Sequence Diagrams

The following figures describe the sequence on DVFS interrupt.

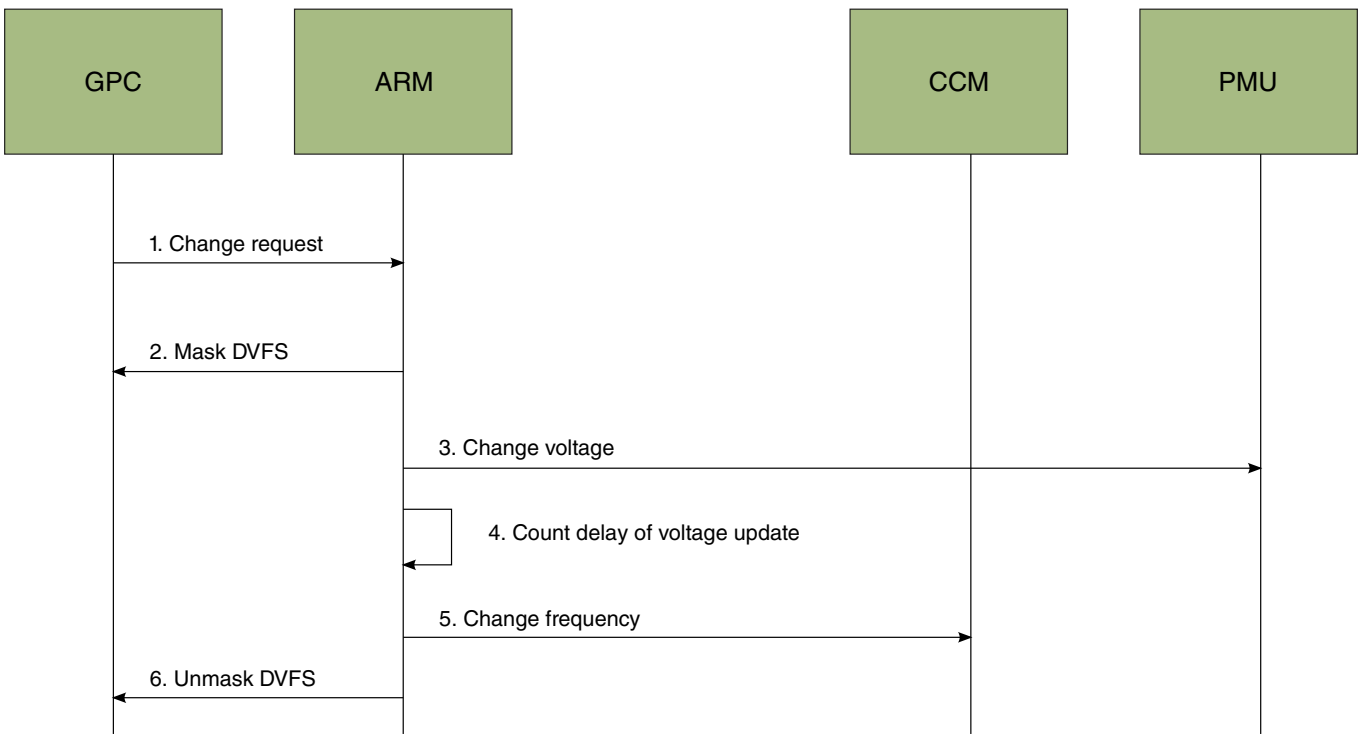


Figure 25-3. DVFS - frequency increase

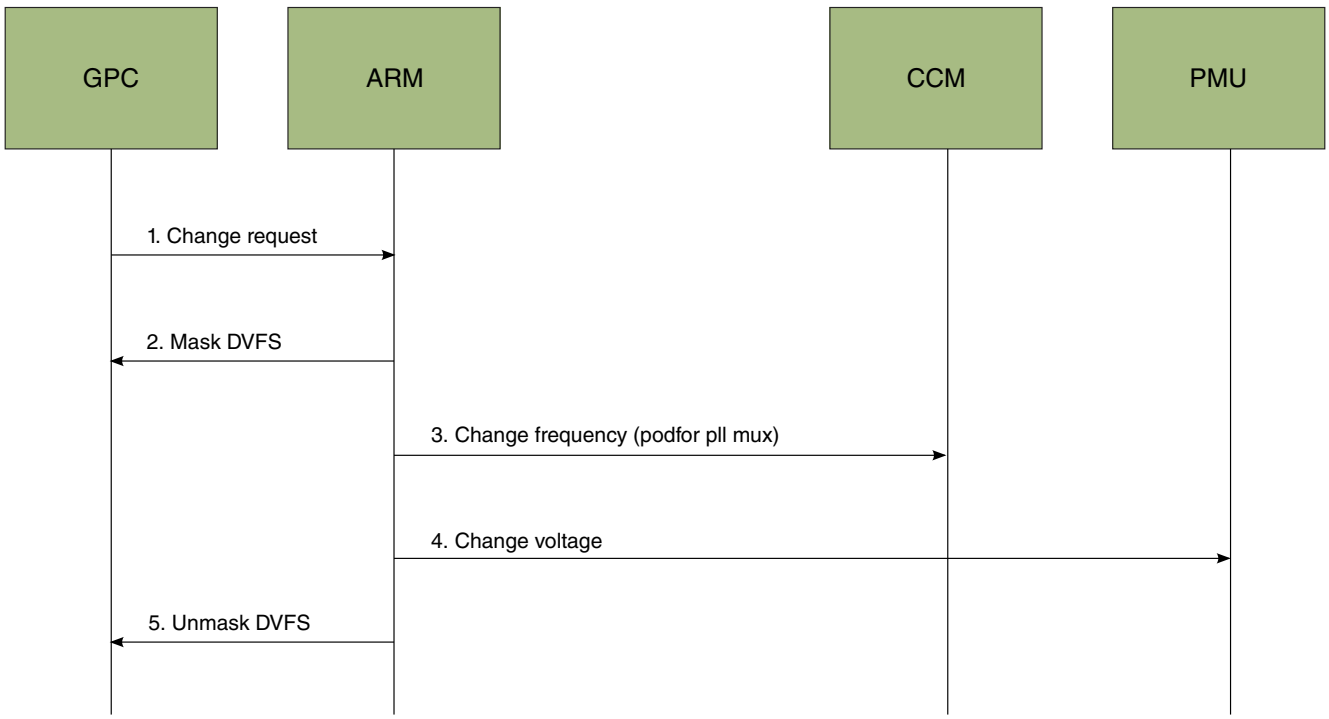


Figure 25-4. DVFS - frequency decrease

CPU DVFS frequency change can be performed in 2 ways:

- PLL inputs muxing update
- clock dividers update

25.5 Power Gating Control (PGC)

Power Gating (PGC) is applied to the ARM CPU only in STOP low power mode, after all essential CPU registers data are saved by ARM dormant procedure.

If any of the unmasked interrupts appears, CPU is powered up and clock restore request (exit from STOP mode) is sent to CCM.

PGC power down sequence:

- CCM sends power down request when the chip is about to enter stop mode. The user should define which modules will be powered down (PGCR registers of corresponding PGC module, bit 0).

PGC power up sequence:

- One of the power up irq is asserted.

- Power up request is asserted in GPC and in CCM.
- The Power Gated modules are powered up, according to PGC settings of appropriate module.

The Power Gated modules require reset after powering up. The next figure describes GPC-SRC handshake procedure for reset after power gating.

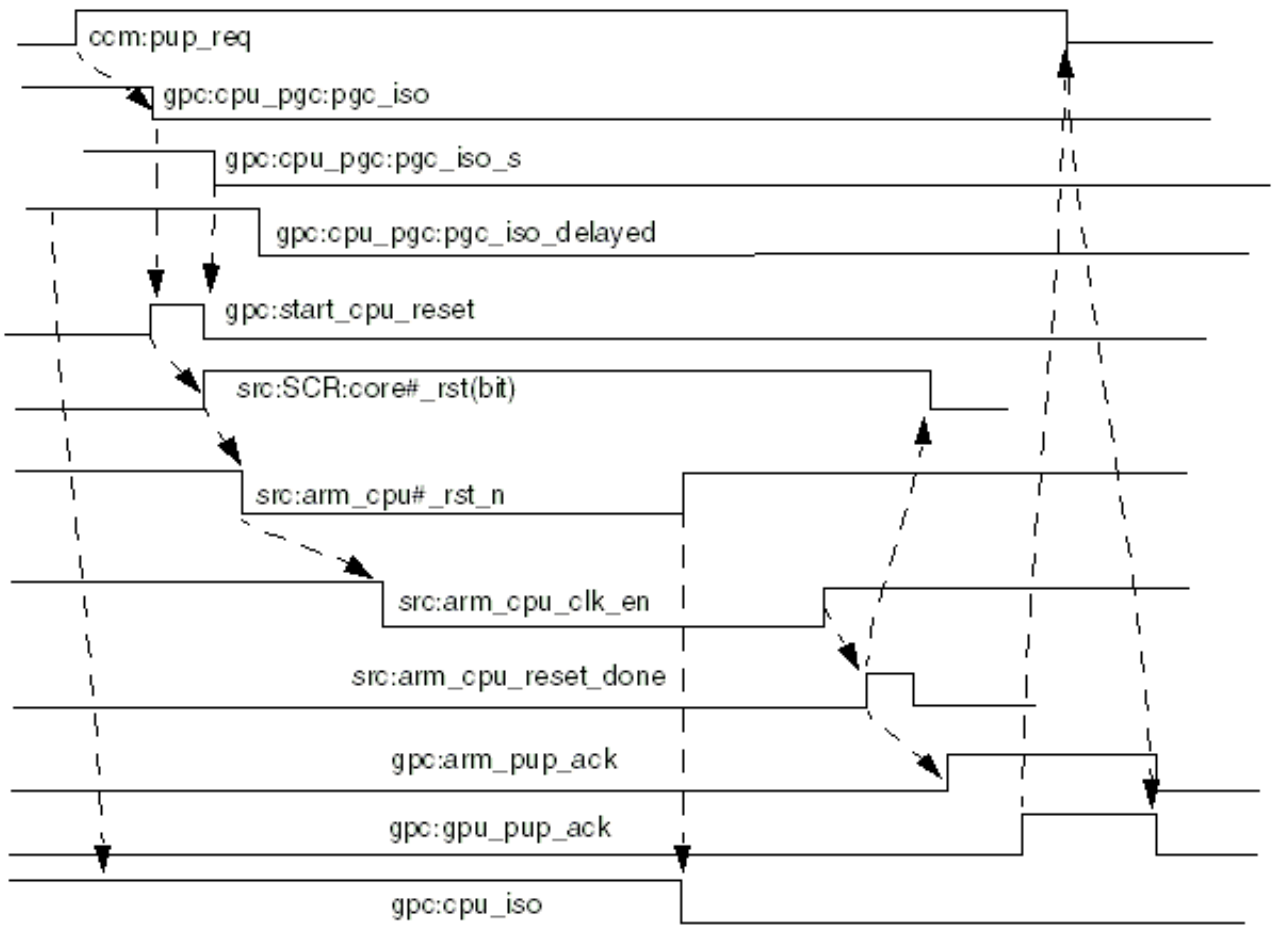


Figure 25-5. GPC-SRC handshake for reset after power gating

25.5.1 Overview

The Power Gating Controller (PGC) is a power management component that controls the power-down and power-up sequencing of individual subsystems.

The sequence timing is programmable using the PGC control registers. Figure 25-6 shows PGC as part of the SoC's overall power management scheme.

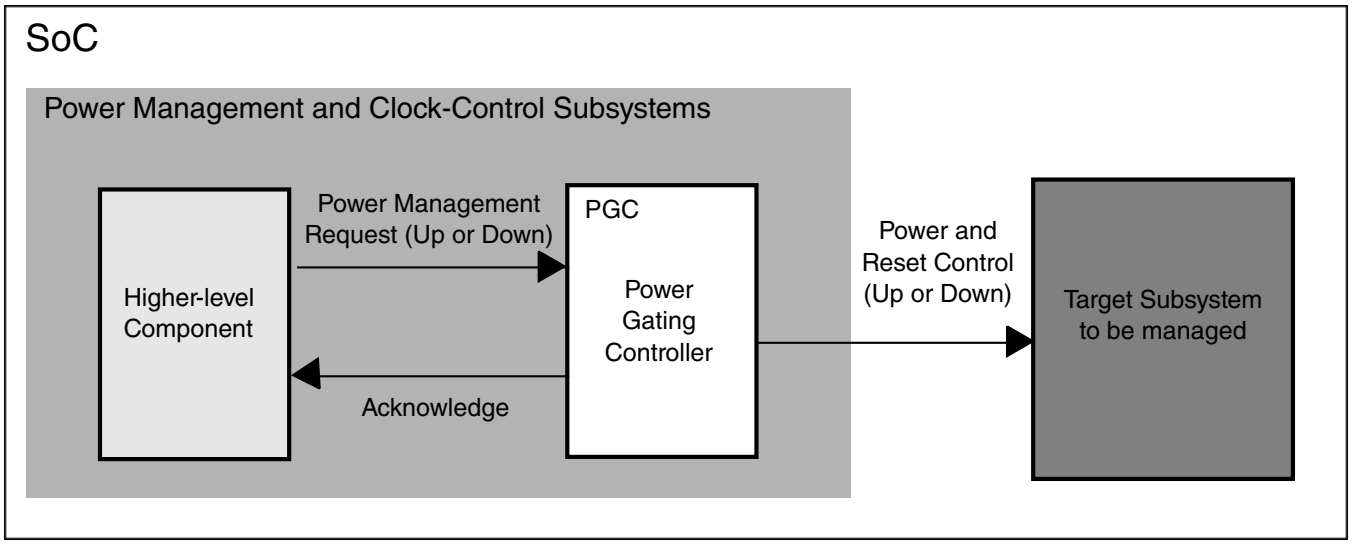


Figure 25-6. PGC Block Diagram

25.5.1.1 Features

Key features of the PGC include:

- Provides the ability to switch off power to a target subsystem.
- Generates power-up and power-down control sequences.
- Provides programmable registers to adjust the timing of the power control signals.

25.6 GPC Interrupt Controller (INTC)

The INTC (Interrupt Controller) detects an interrupt and generates the wakeup signal. It supports up to 128 interrupts.

25.6.1 Interrupt Controller features

The features of the GPC INTC are listed below.

Features:

- Supports up to 128 interrupts
- Provides an option to mask/unmask each interrupt
- Detects interrupts and generates the wake up signal
- 32-bits IP bus interface
- All registers are byte-accessible

25.7 GPC Memory Map/Register Definition

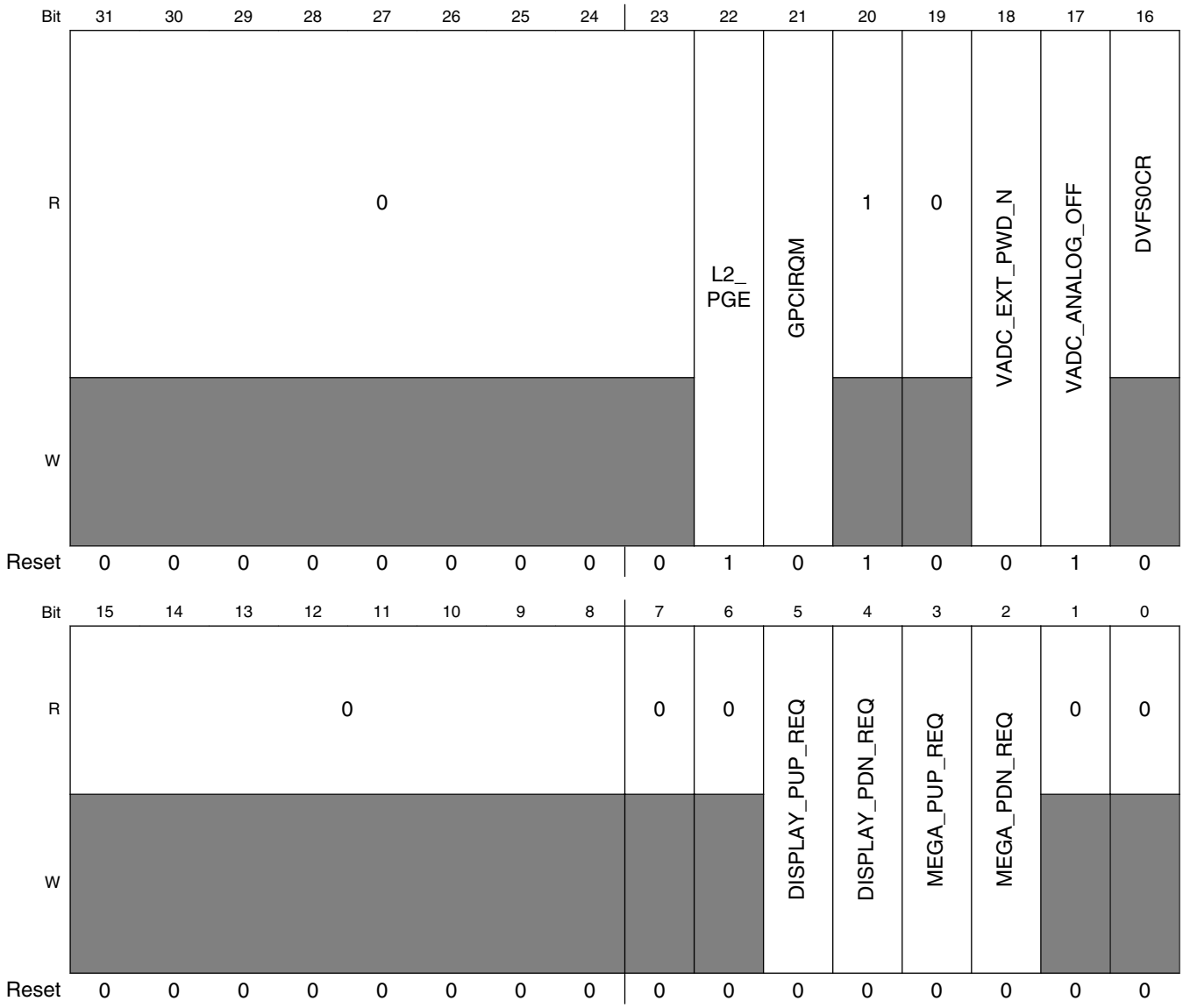
Detailed descriptions of each register can be found below.

GPC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20D_C000	GPC Interface control register (GPC_CNTR)	32	R/W	0052_0000h	25.7.1/1107
20D_C004	GPC Power Gating Register (GPC_PGR)	32	R/W	0000_0000h	25.7.2/1110
20D_C008	IRQ masking register 1 (GPC_IMR1)	32	R/W	0000_0000h	25.7.3/1110
20D_C00C	IRQ masking register 2 (GPC_IMR2)	32	R/W	0000_0000h	25.7.4/1111
20D_C010	IRQ masking register 3 (GPC_IMR3)	32	R/W	0000_0000h	25.7.5/1111
20D_C014	IRQ masking register 4 (GPC_IMR4)	32	R/W	0000_0000h	25.7.6/1112
20D_C018	IRQ status resister 1 (GPC_ISR1)	32	R	0000_0000h	25.7.7/1112
20D_C01C	IRQ status resister 2 (GPC_ISR2)	32	R	0000_0000h	25.7.8/1113
20D_C020	IRQ status resister 3 (GPC_ISR3)	32	R	0000_0000h	25.7.9/1113
20D_C024	IRQ status resister 4 (GPC_ISR4)	32	R	0000_0000h	25.7.10/ 1114

25.7.1 GPC Interface control register (GPC_CNTR)

Address: 20D_C000h base + 0h offset = 20D_C000h



GPC_CNTR field descriptions

Field	Description
31–23 Reserved	This read-only field is reserved and always has the value 0.
22 L2_PGE	L2 Cache Power Gate Enable 1 L2 cache power gate off request, L2 cache will be power down once when CPU core is power down and will be hardware invalidated automatically when CPU core is re-power up 0 L2 cache will keep power on even if CPU core is power down and will not be hardware invalidated when CPU core is re-power up the reset value is 1'b1
21 GPCIRQM	GPC interrupt/event masking 1 interrupt/event is masked 0 not masked

Table continues on the next page...

GPC_CNTR field descriptions (continued)

Field	Description
20 Reserved	This read-only field is reserved and always has the value 1.
19 Reserved	This read-only field is reserved and always has the value 0.
18 VADC_EXT_ PWD_N	VADC power down bit 0 — VADC power down 1 — VADC not power down
17 VADC_ ANALOG_OFF	Indication to VADC whether the analog power to VADC is available or not 0 — VADC analog power is on 1 — VADC analog power is off
16 DVFS0CR	DVFS0 (ARM) Change request (bit is read-only) 1 DVFS0 is requesting for frequency/voltage update 0 DVFS0 has no request
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 Reserved	This read-only field is reserved and always has the value 0.
6 Reserved	This read-only field is reserved and always has the value 0.
5 DISPLAY_PUP_ REQ	Display Power Up request. Self-cleared bit. **NOTE:** Software may directly control display power gate and utilize hardware control for reset sequence 1 Request Power Up sequence to start for Display 0 no request
4 DISPLAY_PDN_ REQ	Display Power Down request. Self-cleared bit. **NOTE:** Software may directly control display power gate and utilize hardware control for reset sequence 1 Request Power Down sequence to start for Display 0 no request
3 MEGA_PUP_ REQ	MEGA domain power up request. Self-clear bit. **NOTE:** Software may directly control display power gate and utilize hardware control for reset sequence. 0 — No Request 1 — Request power up sequence
2 MEGA_PDN_ REQ	MEGA domain power down request. Self-clear bit. **NOTE:** Software may directly control display power gate and utilize hardware control for reset sequence. Caution, MEGA domain is not allowed to power down when CPU is not powered down. 0 — No Request 1 — Request power down sequence
1 Reserved	This read-only field is reserved and always has the value 0.

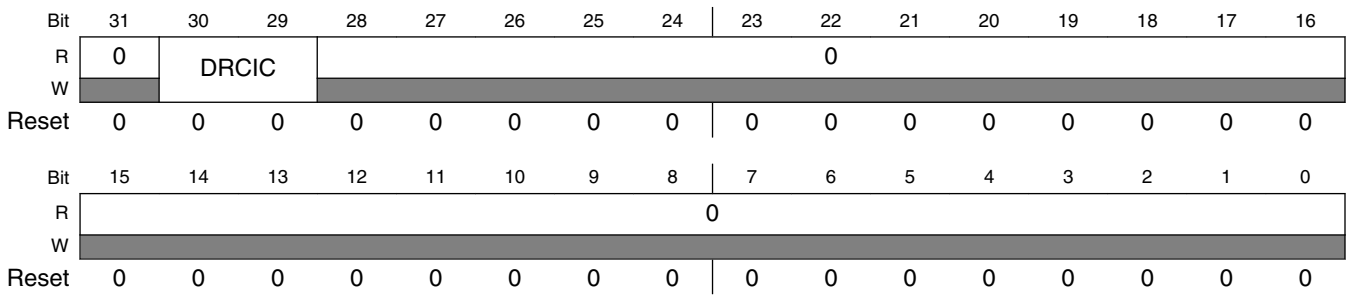
Table continues on the next page...

GPC_CNTR field descriptions (continued)

Field	Description
0 Reserved	This read-only field is reserved and always has the value 0.

25.7.2 GPC Power Gating Register (GPC_PGR)

Address: 20D_C000h base + 4h offset = 20D_C004h



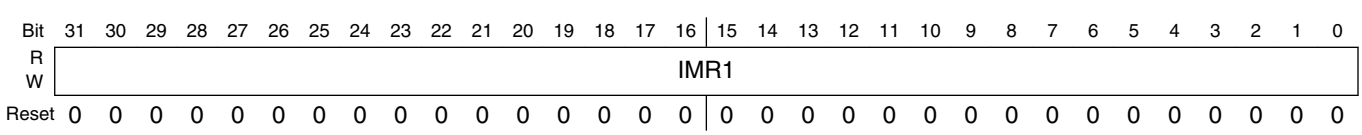
GPC_PGR field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–29 DRCIC	Debug ref cir in mux control 00 ccm_cosr_1_clk_in 01 ccm_cosr_2_clk_in 10 restricted 11 restricted
Reserved	This read-only field is reserved and always has the value 0.

25.7.3 IRQ masking register 1 (GPC_IMR1)

IMR1 Register - masking of irq[63:32].

Address: 20D_C000h base + 8h offset = 20D_C008h



GPC_IMR1 field descriptions

Field	Description
IMR1	IRQ[63:32] masking bits: 1-irq masked, 0-irq is not masked

25.7.4 IRQ masking register 2 (GPC_IMR2)

IMR2 Register - masking of irq[95:64].

Address: 20D_C000h base + Ch offset = 20D_C00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR2 field descriptions

Field	Description
IMR2	IRQ[95:64] masking bits: 1-irq masked, 0-irq is not masked

25.7.5 IRQ masking register 3 (GPC_IMR3)

IMR3 Register - masking of irq[127:96].

Address: 20D_C000h base + 10h offset = 20D_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

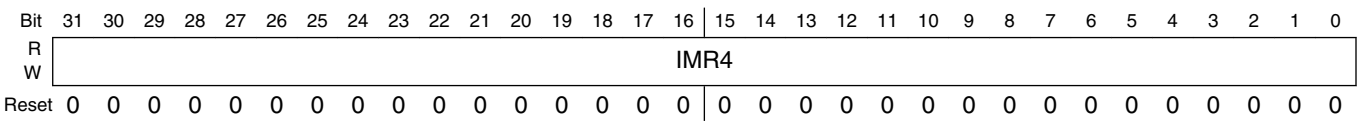
GPC_IMR3 field descriptions

Field	Description
IMR3	IRQ[127:96] masking bits: 1-irq masked, 0-irq is not masked

25.7.6 IRQ masking register 4 (GPC_IMR4)

IMR4 Register - masking of irq[159:128].

Address: 20D_C000h base + 14h offset = 20D_C014h



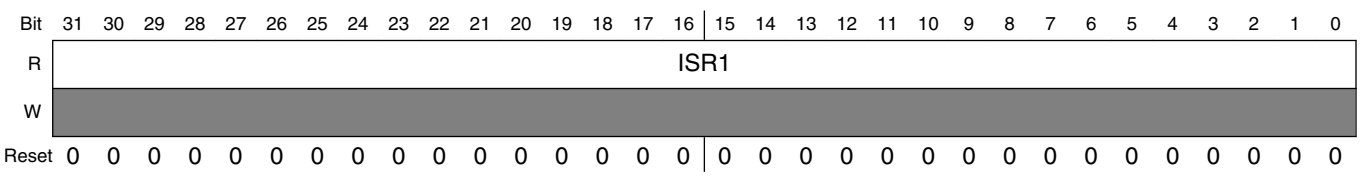
GPC_IMR4 field descriptions

Field	Description
IMR4	IRQ[159:128] masking bits: 1-irq masked, 0-irq is not masked

25.7.7 IRQ status resister 1 (GPC_ISR1)

ISR1 Register - status of irq [63:32].

Address: 20D_C000h base + 18h offset = 20D_C018h



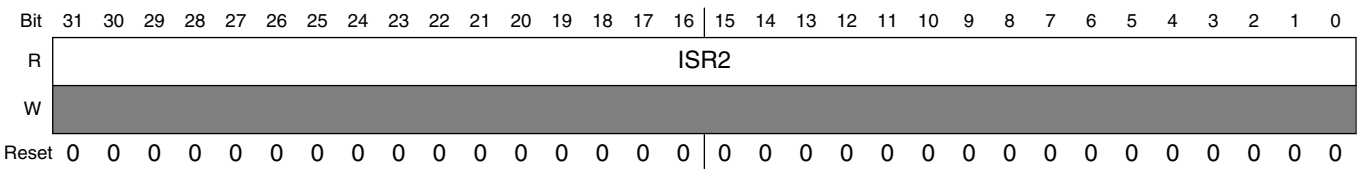
GPC_ISR1 field descriptions

Field	Description
ISR1	IRQ[63:32] status, read only

25.7.8 IRQ status resister 2 (GPC_ISR2)

ISR2 Register - status of irq [95:64].

Address: 20D_C000h base + 1Ch offset = 20D_C01Ch



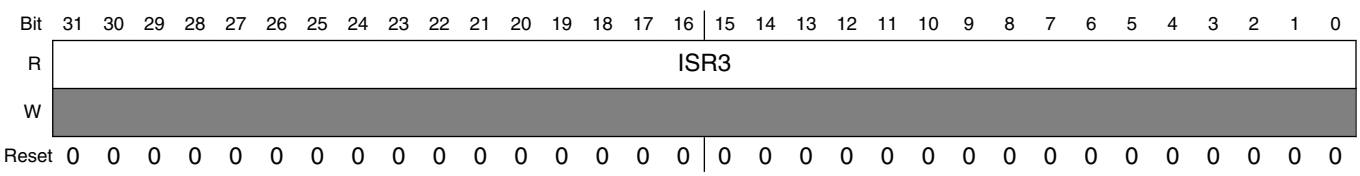
GPC_ISR2 field descriptions

Field	Description
ISR2	IRQ[95:64] status, read only

25.7.9 IRQ status resister 3 (GPC_ISR3)

ISR3 Register - status of irq [127:96].

Address: 20D_C000h base + 20h offset = 20D_C020h



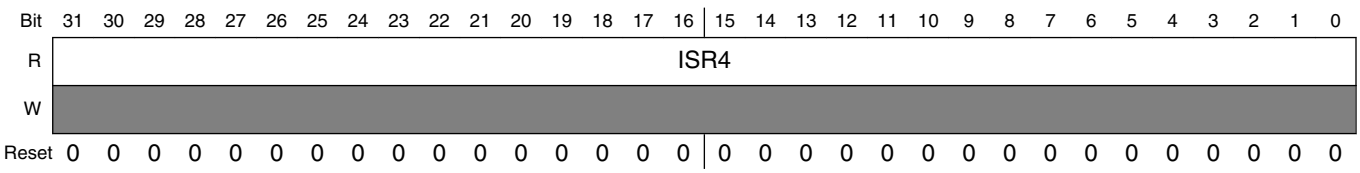
GPC_ISR3 field descriptions

Field	Description
ISR3	IRQ[127:96] status, read only

25.7.10 IRQ status resister 4 (GPC_ISR4)

ISR4 Register - status of irq [159:128].

Address: 20D_C000h base + 24h offset = 20D_C024h



GPC_ISR4 field descriptions

Field	Description
ISR4	IRQ[159:128] status, read only

25.8 PGC Memory Map/Register Definition

The PGC registers can be accessed only in supervisor mode.

Attempts to access registers when not in supervisor mode or attempts to access an unimplemented address location might trigger a bus transfer error. (The hardware asserts the signal ips_xfr_err if the PGC has been integrated with resp_sel tied low.) In this case, software should take appropriate action (such as ignore the error, log the error, or initiate a soft reset).

All PGC registers are byte-accessible.

NOTE

The base address of each PGC module instantiation is specified in the GPC module. Absolute address values will be calculated by [GPC base address] + [PGC CPU/GPU/DISPLAY Offset].

PGC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
220	PGC Control Register (PGC_MEGA_CTRL)	32	R/W	0000_0000h	25.8.1/1115

Table continues on the next page...

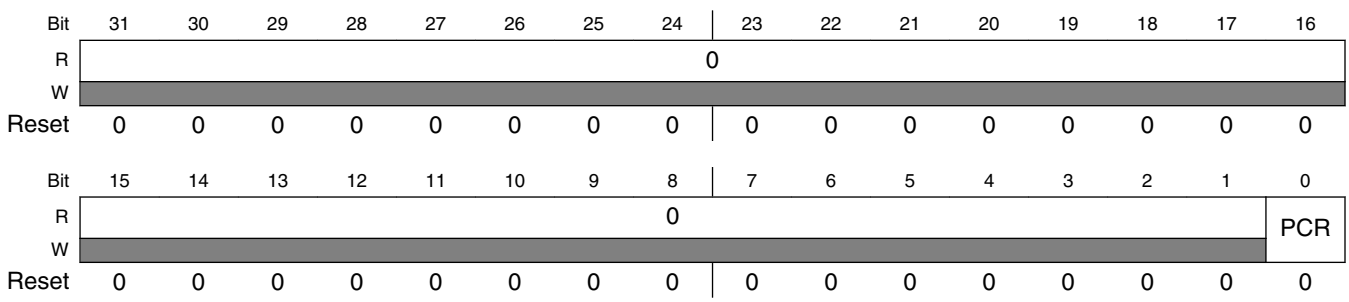
PGC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
224	Power Up Sequence Control Register (PGC_MEGA_PUPSCR)	32	R/W	0000_0F01h	25.8.2/1116
228	Pull Down Sequence Control Register (PGC_MEGA_PDNSCR)	32	R/W	0000_0101h	25.8.3/1116
22C	Power Gating Controller Status Register (PGC_MEGA_SR)	32	R/W	0000_0000h	25.8.4/1117
2A0	PGC Control Register (PGC_CPU_CTRL)	32	R/W	0000_0000h	25.8.5/1118
2A4	Power Up Sequence Control Register (PGC_CPU_PUPSCR)	32	R/W	0000_0F01h	25.8.6/1118
2A8	Pull Down Sequence Control Register (PGC_CPU_PDNSCR)	32	R/W	0000_0101h	25.8.7/1119
2AC	Power Gating Controller Status Register (PGC_CPU_SR)	32	R/W	0000_0000h	25.8.8/1120

25.8.1 PGC Control Register (PGC_MEGA_CTRL)

The PGCR enables the response to a power-down request.

Address: 0h base + 220h offset = 220h



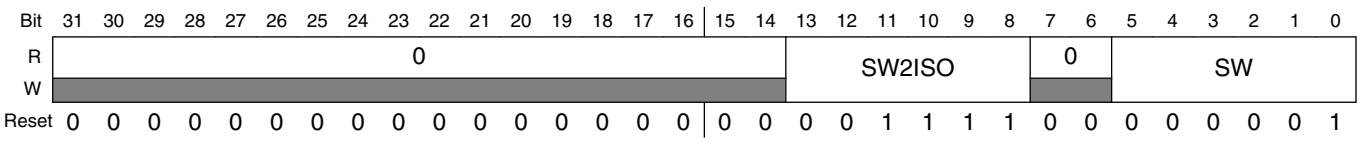
PGC_MEGA_CTRL field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PCR	Power Control **NOTE:** PCR must not change from power-down request (pdn_req) assertion until the target subsystem is completely powered up. 0 Do not switch off power even if pdn_req is asserted. 1 Switch off power when pdn_req is asserted.

25.8.2 Power Up Sequence Control Register (PGC_MEGA_PUPSCR)

The PUPSCR contains the power-up timing parameters.

Address: 0h base + 224h offset = 224h



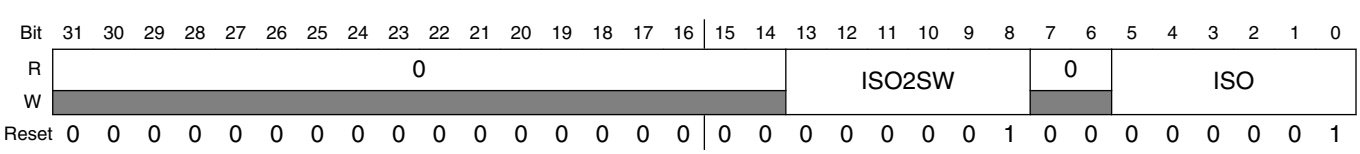
PGC_MEGA_PUPSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 SW2ISO	After asserting power toggle on/off signal (switch_b), the PGC waits a number of IPG clocks equal to the value of SW2ISO before negating isolation. **NOTE:** SW2ISO must not be programmed to zero.
7–6 Reserved	This read-only field is reserved and always has the value 0.
SW	After a power-up request (pup_req assertion), the PGC waits a number of IPG clocks equal to the value of SW before asserting power toggle on/off signal (switch_b). **NOTE:** SW must not be programmed to zero. **NOTE:** The PGC clock is generated from the IPG_CLK_ROOT. for frequency configuration of the IPG_CLK_ROOT. See Clock Controller Module (CCM).

25.8.3 Pull Down Sequence Control Register (PGC_MEGA_PDNSCR)

The PDNSCR contains the power-down timing parameters.

Address: 0h base + 228h offset = 228h



PGC_MEGA_PDNSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 ISO2SW	After asserting isolation, the PGC waits a number of IPG clocks equal to the value of ISO2SW before negating power toggle on/off signal (switch_b). **NOTE:** ISO2SW must not be programmed to zero.
7–6 Reserved	This read-only field is reserved and always has the value 0.
ISO	After a power-down request (pdn_req assertion), the PGC waits a number of IPG clocks equal to the value of ISO before asserting isolation. **NOTE:** ISO must not be programmed to zero.

25.8.4 Power Gating Controller Status Register (PGC_MEGA_SR)

The PDNSCR contains the power-down timing parameters.

Address: 0h base + 22Ch offset = 22Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

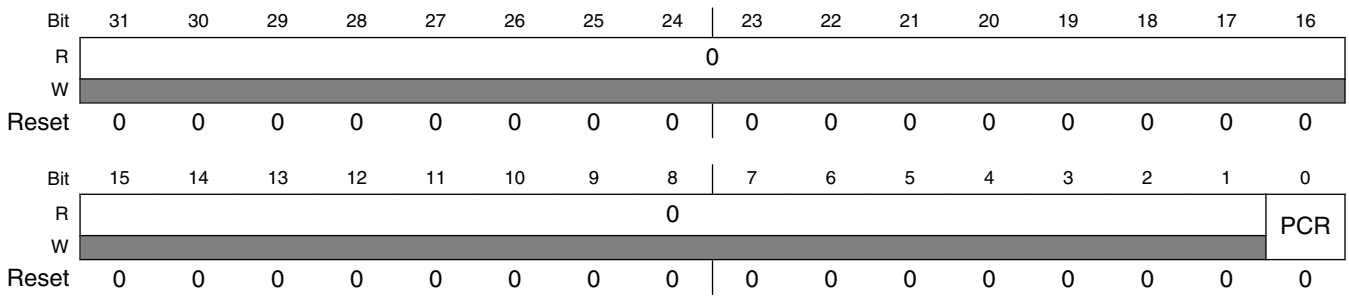
PGC_MEGA_SR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PSR	Power status. When in functional (or software-controlled debug) mode, PGC hardware sets PSR as soon as any of the power control output changes its state to one. Write one to clear this bit. Software should clear this bit after power up; otherwise, PSR continues to reflect the power status of the initial power down. 0 The target subsystem was not powered down for the previous power-down request. 1 The target subsystem was powered down for the previous power-down request.

25.8.5 PGC Control Register (PGC_CPU_CTRL)

The PGCR enables the response to a power-down request.

Address: 0h base + 2A0h offset = 2A0h



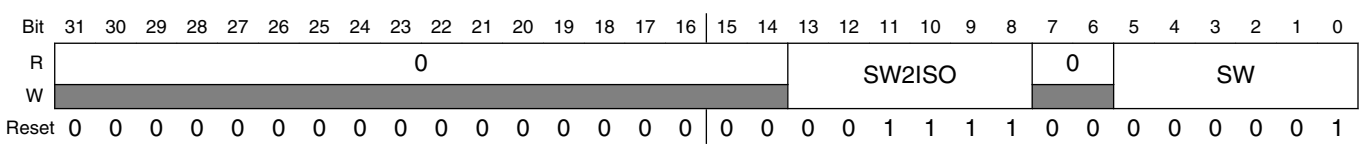
PGC_CPU_CTRL field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PCR	Power Control **NOTE:** PCR must not change from power-down request (pdn_req) assertion until the target subsystem is completely powered up. 0 Do not switch off power even if pdn_req is asserted. 1 Switch off power when pdn_req is asserted.

25.8.6 Power Up Sequence Control Register (PGC_CPU_PUPSCR)

The PUPSCR contains the power-up timing parameters.

Address: 0h base + 2A4h offset = 2A4h



PGC_CPU_PUPSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.

13–8
SW2ISO
	There are two different silicon revisions: 1.0 and 1.1 (defined by USB_ANALOG_DIGPROG).
	• For silicon revision 1.0: PGC waits for 2048*SW2ISO cycles of IPG_CLK before negating isolation.
	• For silicon revision 1.1:
	  • SW[5] = 1: PGC waits for 64*SW2ISO cycles of IPG_CLK before negating isolation.
	  • SW[5] = 0: PGC waits for 2048*SW2ISO cycles of IPG_CLK before negating isolation.
	
	NOTE: SW2ISO must not be programmed to zero. The SW2ISO value should be chosen such that the delay before negating isolation is greater than the LDO ramp-up time.

7–6 Reserved	This read-only field is reserved and always has the value 0.

SW	There are two different silicon revisions: 1.0 and 1.1 (defined by USB_ANALOG_DIGPROG). When IPG_CLK frequency is slow down in low power mode, it can set SW[5] to speed up ARM wakeup time.
	• For silicon revision 1.0: PGC waits for 2048*SW[5:0] cycles of IPG_CLK to switch on ARM power after pup_reg is asserted.
	• For silicon revision 1.1:
	  • SW[5] = 1: PGC waits for 64*SW[4:0] cycles of IPG_CLK to switch on ARM power after pup_req is asserted.
	  • SW[5] = 0: PGC waits for 2048*SW[4:0] cycles of IPG_CLK to switch on ARM power after pup_req is asserted.
	
	NOTE: SW[4:0] can be programmed to zero.

25.8.7 Pull Down Sequence Control Register (PGC_CPU_PDNSCR)

The PDNSCR contains the power-down timing parameters.

Address: 0h base + 2A8h offset = 2A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																ISO2SW								0		ISO					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

PGC_CPU_PDNSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 ISO2SW	After asserting isolation, the PGC waits a number of 32k clocks equal to the value of ISO2SW before negating .

Table continues on the next page...

PGC_CPU_PDNSCR field descriptions (continued)

Field	Description
	NOTE: ISO2SW must not be programmed to zero.
7–6 Reserved	This read-only field is reserved and always has the value 0.
ISO	After a power-down request (pdn_req assertion), the PGC waits a number of 32k clocks equal to the value of ISO before asserting isolation. **NOTE:** ISO must not be programmed to zero.

25.8.8 Power Gating Controller Status Register (PGC_CPU_SR)

The PDNSCR contains the power-down timing parameters.

Address: 0h base + 2ACh offset = 2ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

PGC_CPU_SR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PSR	Power status. When in functional (or software-controlled debug) mode, PGC hardware sets PSR as soon as any of the power control output changes its state to one. Write one to clear this bit. Software should clear this bit after power up; otherwise, PSR continues to reflect the power status of the initial power down. 0 The target subsystem was not powered down for the previous power-down request. 1 The target subsystem was powered down for the previous power-down request.

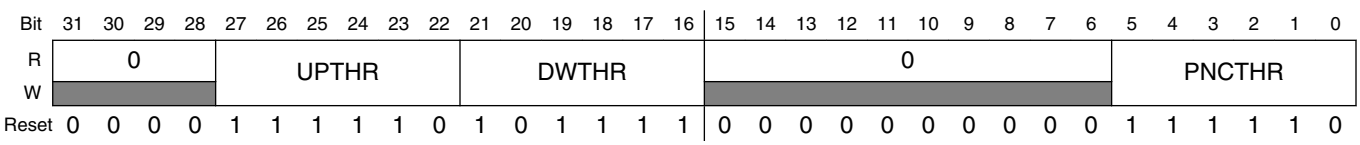
25.9 DVFSC Memory Map/Register Definition

DVFSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	DVFS Thresholds (DVFSC_THRS)	32	R/W	0FAF_003Eh	25.9.1/1121
4	DVFS Counters thresholds (DVFSC_COUN)	32	R/W	0007_0020h	25.9.2/1122
8	DVFS general purpose bits weight (DVFSC_SIG1)	32	R/W	0000_0000h	25.9.3/1122
C	DVFS general purpose bits weight (DVFSC_DVFSSIG0)	32	R/W	0000_0000h	25.9.4/1123
10	DVFS general purpose bit 0 weight counter (DVFSC_DVFSGPC0)	32	R/W	0000_0000h	25.9.5/1124
14	DVFS general purpose bit 1 weight counter (DVFSC_DVFSGPC1)	32	R/W	0000_0000h	25.9.6/1125
18	DVFS general purpose bits enables (DVFSC_DVFSGPBT)	32	R/W	0000_0000h	25.9.7/1126
1C	DVFS EMAC settings (DVFSC_DVFSEMAC)	32	R/W	0000_0004h	25.9.8/1128
20	DVFS Control (DVFSC_CNTR)	32	R/W	0900_000Eh	25.9.9/1129
24	DVFS Load Tracking Register 0, portion 0 (DVFSC_DVFSLTR0_0)	32	R	0000_0000h	25.9.10/ 1132
28	DVFS Load Tracking Register 0, portion 1 (DVFSC_DVFSLTR0_1)	32	R	0000_0000h	25.9.11/ 1133
2C	DVFS Load Tracking Register 1, portion 0 (DVFSC_DVFSLTR1_0)	32	R	0000_0000h	25.9.12/ 1133
30	DVFS Load Tracking Register 3, portion 1 (DVFSC_DVFSLTR1_1)	32	R	0000_0000h	25.9.13/ 1134
34	DVFS pattern 0 length (DVFSC_DVFSPT0)	32	R/W	0000_0010h	25.9.14/ 1135
38	DVFS pattern 1 length (DVFSC_DVFSPT1)	32	R/W	0000_0010h	25.9.15/ 1135
3C	DVFS pattern 2 length (DVFSC_DVFSPT2)	32	R/W	0000_0010h	25.9.16/ 1136
40	DVFS pattern 3 length (DVFSC_DVFSPT3)	32	R/W	0000_0010h	25.9.17/ 1137

25.9.1 DVFS Thresholds (DVFSC_THRS)

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



DVFSC_THRS field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

DVFSC_THRS field descriptions (continued)

Field	Description
27–22 UPTHR	Upper threshold for load tracking
21–16 DWTHR	Down threshold for load tracking
15–6 Reserved	This read-only field is reserved and always has the value 0.
PNCTHR	Panic threshold for load tracking

25.9.2 DVFS Counters thresholds (DVFSC_COUN)

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

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

DVFSC_COUN field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.
23–16 DN_CNT	Down counter threshold value
15–8 Reserved	This read-only field is reserved and always has the value 0.
UPCNT	UP counter threshold value

25.9.3 DVFS general purpose bits weight (DVFSC_SIG1)

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WSW15			WSW14			WSW13			WSW12			WSW11			WSW10		WSW9			WSW8			WSW7			WSW6			0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_SIG1 field descriptions

Field	Description
31–29 WSW15	General purpose load tracking signal weight dvfs_w_sig[15]

Table continues on the next page...

DVFSC_SIG1 field descriptions (continued)

Field	Description
28–26 WSW14	General purpose load tracking signal weight dvfs_w_sig[14]
25–23 WSW13	General purpose load tracking signal weight dvfs_w_sig[13]
22–20 WSW12	General purpose load tracking signal weight dvfs_w_sig[12]
19–17 WSW11	General purpose load tracking signal weight dvfs_w_sig[11]
16–14 WSW10	General purpose load tracking signal weight dvfs_w_sig[10]
13–11 WSW9	General purpose load tracking signal weight dvfs_w_sig[9]
10–8 WSW8	General purpose load tracking signal weight dvfs_w_sig[8]
7–5 WSW7	General purpose load tracking signal weight dvfs_w_sig[7]
4–2 WSW6	General purpose load tracking signal weight dvfs_w_sig[6]
Reserved	This read-only field is reserved and always has the value 0.

25.9.4 DVFS general purpose bits weight (DVFSC_DVFSSIG0)

Address: 0h base + Ch offset = Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSW5			WSW4			WSW3			WSW2			Reserved								WSW1						WSW0						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

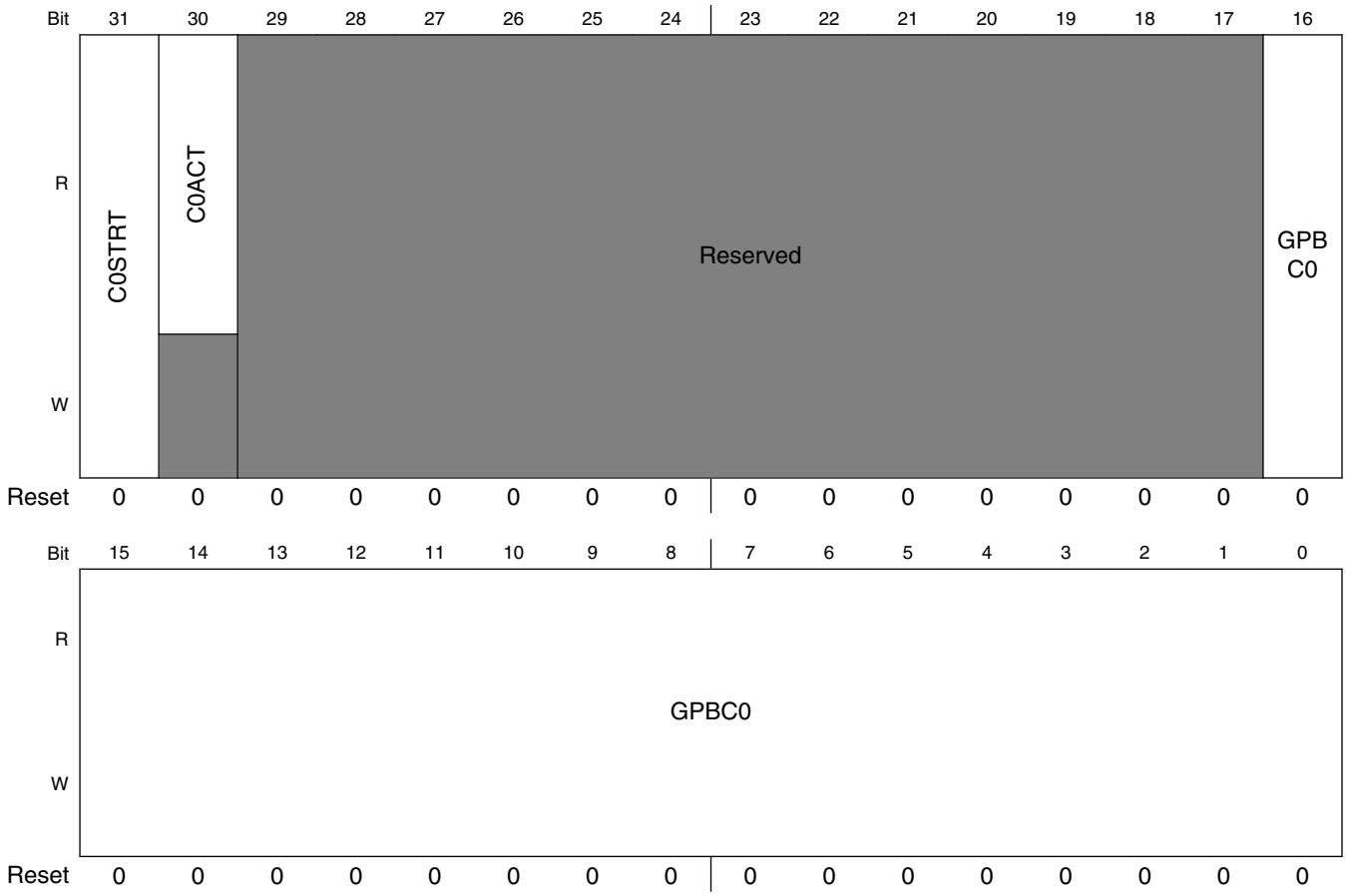
DVFSC_DVFSSIG0 field descriptions

Field	Description
31–29 WSW5	General purpose load tracking signal weight dvfs_w_sig[5]
28–26 WSW4	General purpose load tracking signal weight dvfs_w_sig[4]
25–23 WSW3	General purpose load tracking signal weight dvfs_w_sig[3]
22–20 WSW2	General purpose load tracking signal weight dvfs_w_sig[2]
19–12 -	This field is reserved. Reserved
11–6 WSW1	General purpose load tracking signal weight dvfs_w_sig[1]. This value is relevant during GPC1 counting period or when GPB1 is set.
WSW0	General purpose load tracking signal weight dvfs_w_sig[0]. This value is relevant during GPC0 counting period or when GPB0 is set.

25.9.5 DVFS general purpose bit 0 weight counter (DVFSC_DVFSGPC0)

DVFS general purpose bits weight counter.

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



DVFSC_DVFSGPC0 field descriptions

Field	Description
31 C0STRT	C0STRT - Counter 0 start Setting of this bit will initialize down counting of the GPC0 value. Bit is self-cleared next cycle after setting. Any setting of this bit will re-start GPC0 counter to the GPC0 value. GPB0 bit disables (overrides) GPC0 counter - WSW0 weight is applicable continuously
30 C0ACT	C0ACT - Counter 0 active indicator

Table continues on the next page...

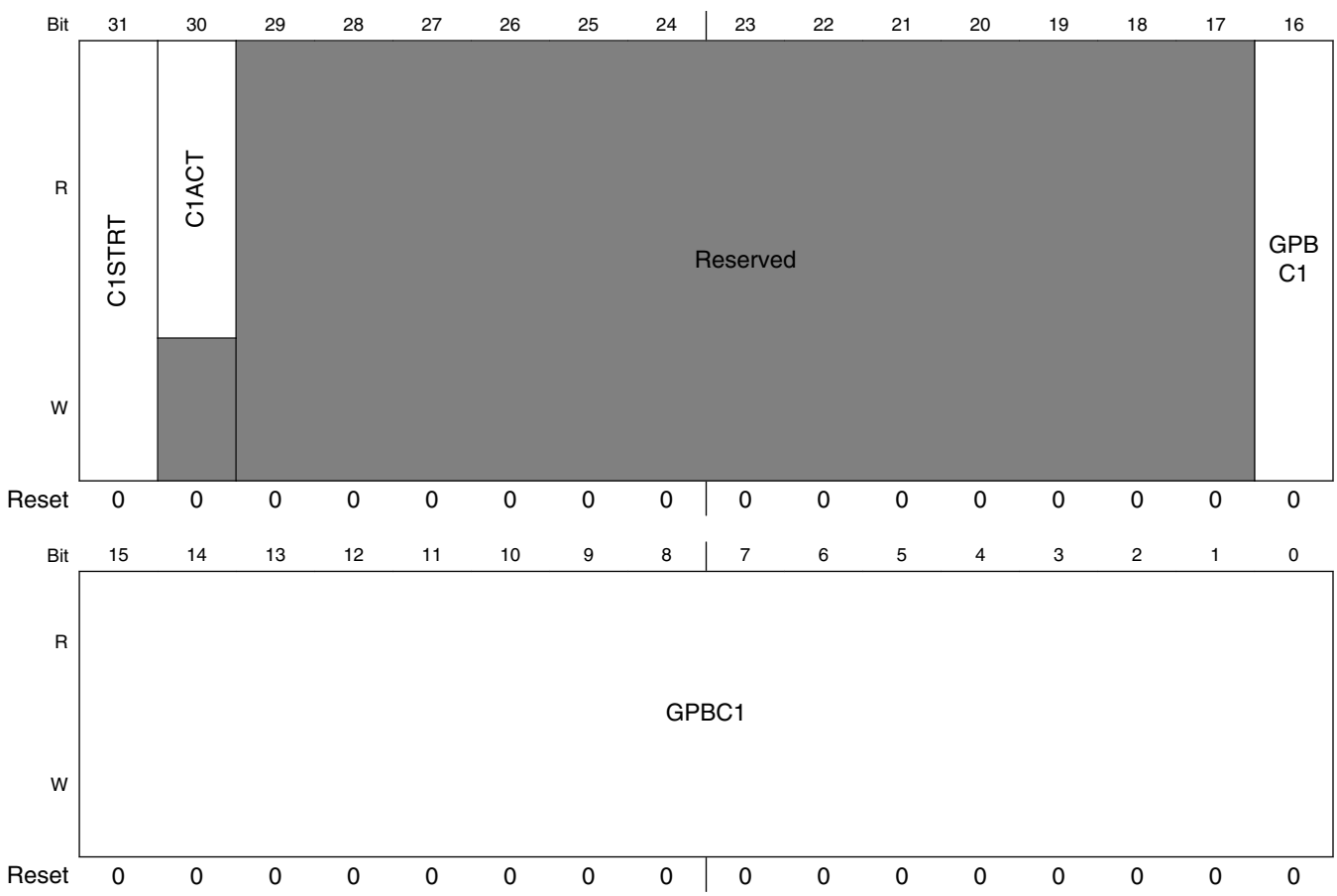
DVFSC_DVFSGPC0 field descriptions (continued)

Field	Description
	1 General Purpose bit0 counter didn't reach value of "0" - the WSW0 is provided to DVFS calculation 0 General Purpose bit0 counter reached value of "0" - the instead of WSW0, "0" (zero) is provided to DVFS calculation
29–17 -	This field is reserved. reserved
GPBC0	GPBC0 - General Purpose bits Counter 0 During period of this counter the GeP bit 0 will be set and WSW0 will be added to the calculations.

25.9.6 DVFS general purpose bit 1 weight counter (DVFSC_DVFSGPC1)

DVFS general purpose bits weight counter1.

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



DVFSC_DVFSGPC1 field descriptions

Field	Description
31 C1STRT	C1STRT - Counter 1start Setting of this bit will initialize down counting of the GPC1 value. Bit is self-cleared next cycle after setting. Any setting of this bit will re-start GPC1 counter to the GPC1 value. GPB1 bit disables (overrides) GPC1 counter - WSW1 weight is applicable continuously
30 C1ACT	C1ACT - Counter 1 active indicator 1 General Purpose bit1 counter didn't reach value of "0" - the WSW1 is provided to DVFS calculation 0 General Purpose bit1 counter reached value of "0" - the instead of WSW1, "0" (zero) is provided to DVFS calculation
29–17 -	This field is reserved. reserved
GPBC1	GPBC1 - General Purpose bits Counter 1 During period of this counter the GeP bit 1 will be set and WSW1 will be added to the calculations.

25.9.7 DVFS general purpose bits enables (DVFSC_DVFSGPBT)

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPB15	GPB14	GPB13	GPB12	GPB11	GPB10	GPB9	GPB8	GPB7	GPB6	GPB5	GPB4	GPB3	GPB2	GPB1	GPB0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_DVFSGPBT field descriptions

Field	Description
31–16 -	This field is reserved. reserved
15 GPB15	General purpose bit 15. Its weight is set by WSW15 value.
14 GPB14	General purpose bit 14. Its weight is set by WSW14 value.

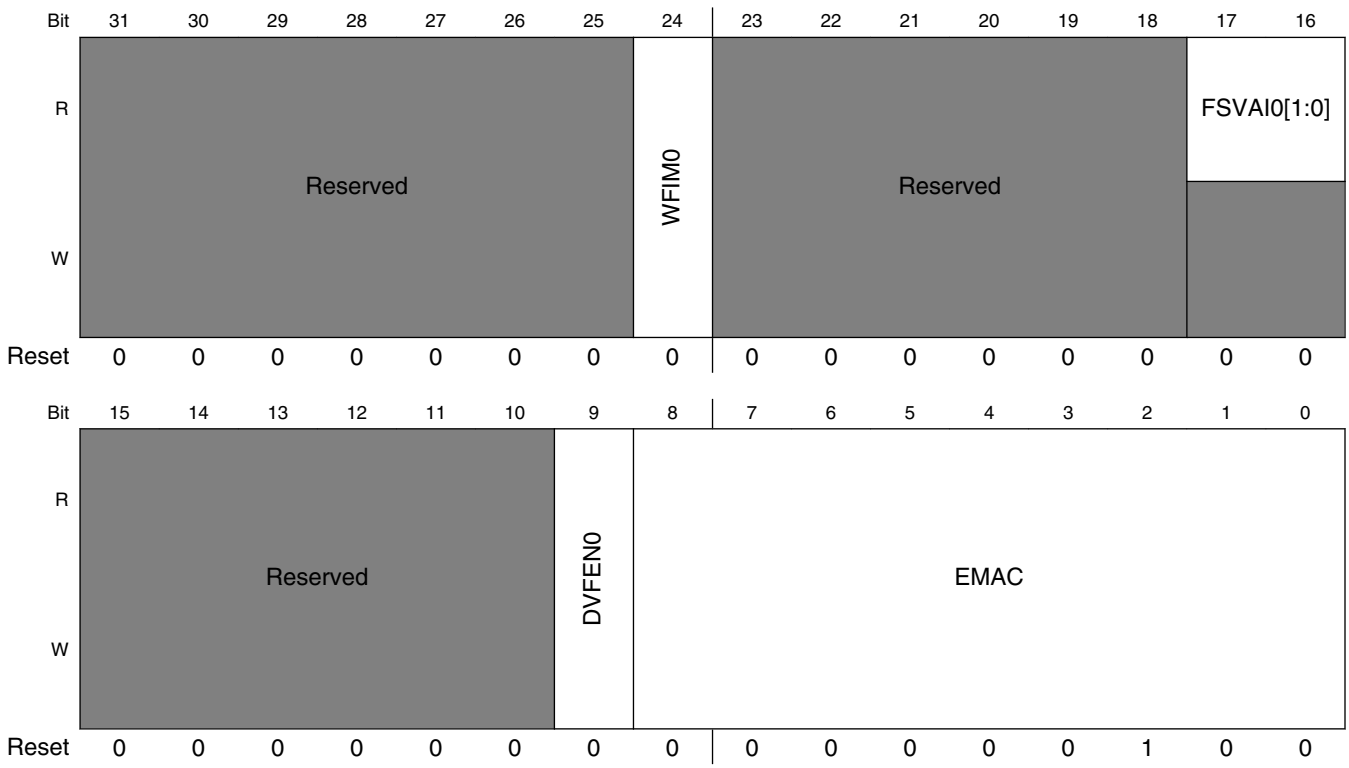
Table continues on the next page...

DVFSC_DVFSGPBT field descriptions (continued)

Field	Description
13 GPB13	General purpose bit 13. Its weight is set by WSW13 value.
12 GPB12	General purpose bit 12. Its weight is set by WSW12 value.
11 GPB11	General purpose bit 11. Its weight is set by WSW11 value.
10 GPB10	General purpose bit 10. Its weight is set by WSW10 value.
9 GPB9	General purpose bit 9. Its weight is set by WSW9 value.
8 GPB8	General purpose bit 8. Its weight is set by WSW8 value.
7 GPB7	General purpose bit 7. Its weight is set by WSW7 value.
6 GPB6	General purpose bit 6. Its weight is set by WSW6 value.
5 GPB5	General purpose bit 5. Its weight is set by WSW5 value.
4 GPB4	General purpose bit 4. Its weight is set by WSW4 value.
3 GPB3	General purpose bit 3. Its weight is set by WSW3 value.
2 GPB2	General purpose bit 2. Its weight is set by WSW2 value.
1 GPB1	General purpose bit 1. Its weight is set by WSW1 value. IF set (1), the GPBC1 operation is disregarded, WSW1 value is applied continuously.
0 GPB0	General purpose bit 0. Its weight is set by WSW0 value. IF set (1), the GPBC0 operation is disregarded, WSW0 value is applied continuously.

25.9.8 DVFS EMAC settings (DVFSC_DVFSEMAC)

Address: 0h base + 1Ch offset = 1Ch



DVFSC_DVFSEMAC field descriptions

Field	Description
31–25 -	This field is reserved. Reserved
24 WFIM0	DVFS Wait for Interrupt of core 0 mask bit 0 Wait for interrupt of core 0 isn't masked 1 Wait for interrupt of core 0 is masked.
23–18 -	This field is reserved. Reserved
17–16 FSVAI0[1:0]	DVFS Frequency adjustment status of core 0. These status bits indicate that frequency should be changed, following load of core 0. 00 no change 01 frequency should be increased. Low priority source for interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MAXF = 1 (highest frequency). 10 frequency should be decreased. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MINF= 1 (lowest frequency). 11 frequency should be increased immediately. High priority source of interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MAXF = 1 (highest frequency).

Table continues on the next page...

DVFSC_DVFSEMAC field descriptions (continued)

Field	Description
15–10 -	This field is reserved. Reserved
9 DVFEN0	DVFS tracking for core0 enable. 1 DVFS enabled. 0 DVFS disabled.
EMAC	EMAC - EMA control value

25.9.9 DVFS Control (DVFSC_CNTR)

Table 25-2. DIV3CK division

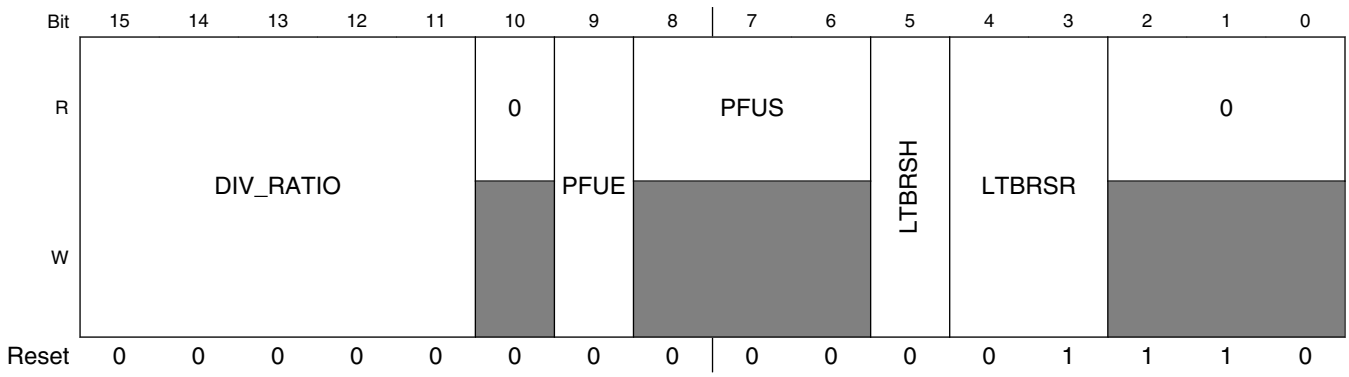
DIV3CK setting	dividing ratio	sum_3 passing bits	div_1_clk cumulative divider
00	1	4-0	1*512=512
001	4	6-2	4*512=2048
010	16	8-4	16*512=8192
011	64	10-6	64*512=32768
100	256	12-8	256*512=131072
101	1024	16-10	1024*512=524288

Table 25-3. Preliminary Divider definition

DIV_RATIO value	ARM clk division ratio
000000	1
000001	2
000010	3
...	...

Address: 0h base + 20h offset = 20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DIV3CK			DVFEV	LBMI	LBFL1	LBFL0	DVFIS	PIRQS	FSVAIM	FSVAI[1:0]		0	MAXF	MINF	DIV_RATIO
W																
Reset	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0



DVFSC_CNTR field descriptions

Field	Description
31–29 DIV3CK	DIV3CK - div_3_clk division ratio inside the DVFS module. According to the Table 25-2
28 DVFEV	Always give a DVFS event. 0 Do not give an event always. 1 Always give event.
27 LBMI	Load buffer full mask interrupt. This bit masks the generation of this interrupt. Load buffer full interrupt is masked (LBFL0 and LBFL1 bits still will be updated, but interrupt won't be generated) Load buffer full interrupt is enabled.
26 LBFL1	Load buffer 1 - full status bit. This bit indicates that log buffer registers are full. The bit is set to 1 automatically. An interrupt will be generated if LBMI bit is set to "0" Write '1' to clear. (write '0' leaves bit unchanged) 1 Load buffer0 is full. 0 Load buffer0 is not full.
25 LBFL0	Load buffer 0 - full status bit. This bit indicates that log buffer registers are full. The bit is set to 1 automatically. An interrupt will be generated if LBMI bit is set to "0" Write '1' to clear. (write '0' leaves bit unchanged) 1 Load buffer1 is full. 0 Load buffer1 is not full.
24 DVFIS	DVFS Interrupt select. These bits define destination of DVFS interrupts. 1 MCU interrupt will be generated for DVFS events. 0 SDMA interrupt will be generated for DVFS events.
23 PIRQS	PIRQS - Pattern IRQ Source * write '1' to clear. Writing '1' will clear interrupt if interrupt was from pattern 1 DVFS IRQ source was from pattern 0 DVFS IRQ source was not from pattern

Table continues on the next page...

DVFSC_CNTR field descriptions (continued)

Field	Description
22 FSVAIM	DVFS Frequency adjustment interrupt mask. This bit masks the DVFS frequency adjustment interrupt. FSVAI status bits will be still asserted in relevant cases. 1 interrupt is masked. 0 interrupt is enabled.
21–20 FSVAI[1:0]	FSVAI DVFS Frequency adjustment interrupt. These status bits indicate that the system frequency should be changed. 00 no interrupt 01 frequency should be increased. Low priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MAXF = 1 (highest frequency). 10 frequency should be decreased. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MINF= 1 (lowest frequency). 11 frequency should be increased immediately. High priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MAXF = 1 (highest frequency).
19 Reserved	This read-only field is reserved and always has the value 0.
18 MAXF	Maximum frequency reached. Interrupt will not be created in maximum frequency reached and frequency increase required. 1 max frequency reached 0 max frequency not reached
17 MINF	Minimum frequency reached. Interrupt will not be created in minimum frequency reached and frequency decrease required. 1 min frequency reached 0 min frequency not reached
16–11 DIV_RATIO	DIV_RATIO - Divider value. Divider divides the input ARM clock, following the table Table 25-3
10 Reserved	This read-only field is reserved and always has the value 0.
9 PFUE	PFUE - Period Frequency Update Enable 1 enabled 0 disabled
8–6 PFUS	PFUS - Periodic Frequency Update Status 000 no update 100 DVFSPT0 period, previous finished(can be performance level decrease) 101 DVFSPT1 period, previous finished(can be EMA-detected performance level) 110 DVFSPT2 period, previous finished(can be performance level increase) 111 DVFSPT3 period, previous finished (can be EMA-detected performance level)
5 LTBRSH	LTBRSH - Load Tracking Buffer Register Shift: 0 values of [5:2] of the selected input are saving in Load Tracking Buffer 1 values of [4:1] of the selected input are saving in Load Tracking Buffer
4–3 LTBRSR	LTBRSR - Load Tracking Buffer Register Source:

Table continues on the next page...

DVFSC_CNTR field descriptions (continued)

Field	Description
	00 pre_ld_add 01 ld_add 10 ema_ld 11 reserved
Reserved	This read-only field is reserved and always has the value 0.

25.9.10 DVFS Load Tracking Register 0, portion 0 (DVFSC_DVFSLTR0_0)

Address: 0h base + 24h offset = 24h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS0_7				LTS0_6				LTS0_5				LTS0_4				LTS0_3				LTS0_2				LTS0_1				LTS0_0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_DVFSLTR0_0 field descriptions

Field	Description
31–28 LTS0_7	core 0 Load Tracking Sample 7
27–24 LTS0_6	core 0 Load Tracking Sample 6
23–20 LTS0_5	core 0 Load Tracking Sample 5
19–16 LTS0_4	core 0 Load Tracking Sample 4
15–12 LTS0_3	core 0 Load Tracking Sample 3
11–8 LTS0_2	core 0 Load Tracking Sample 2
7–4 LTS0_1	core 0 Load Tracking Sample 1
LTS0_0	core 0 Load Tracking Sample 0

25.9.11 DVFS Load Tracking Register 0, portion 1 (DVFSC_DVFSLTR0_1)

Address: 0h base + 28h offset = 28h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS0_15				LTS0_14				LTS0_13				LTS0_12				LTS0_11				LTS0_10				LTS0_9				LTS0_8			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_DVFSLTR0_1 field descriptions

Field	Description
31–28 LTS0_15	core 0 Load Tracking Sample 15
27–24 LTS0_14	core 0 Load Tracking Sample 14
23–20 LTS0_13	core 0 Load Tracking Sample 13
19–16 LTS0_12	core 0 Load Tracking Sample 12
15–12 LTS0_11	core 0 Load Tracking Sample 11
11–8 LTS0_10	core 0 Load Tracking Sample 10
7–4 LTS0_9	core 0 Load Tracking Sample 9
LTS0_8	core 0 Load Tracking Sample 8

25.9.12 DVFS Load Tracking Register 1, portion 0 (DVFSC_DVFSLTR1_0)

Address: 0h base + 2Ch offset = 2Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS1_7				LTS1_6				LTS1_5				LTS1_4				LTS1_3				LTS1_2				LTS1_1				LTS1_0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_DVFSLTR1_0 field descriptions

Field	Description
31–28 LTS1_7	core 0 Load Tracking Sample 7

Table continues on the next page...

DVFSC_DVFSLTR1_0 field descriptions (continued)

Field	Description
27–24 LTS1_6	core 0 Load Tracking Sample 6
23–20 LTS1_5	core 0 Load Tracking Sample 5
19–16 LTS1_4	core 0 Load Tracking Sample 4
15–12 LTS1_3	core 0 Load Tracking Sample 3
11–8 LTS1_2	core 0 Load Tracking Sample 2
7–4 LTS1_1	core 0 Load Tracking Sample 1
LTS1_0	core 0 Load Tracking Sample 0

25.9.13 DVFS Load Tracking Register 3, portion 1 (DVFSC_DVFSLTR1_1)

Address: 0h base + 30h offset = 30h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS1_15				LTS1_14				LTS1_13				LTS1_12				LTS1_11				LTS1_10				LTS1_9				LTS1_8			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_DVFSLTR1_1 field descriptions

Field	Description
31–28 LTS1_15	core 0 Load Tracking Sample 15
27–24 LTS1_14	core 0 Load Tracking Sample 14
23–20 LTS1_13	core 0 Load Tracking Sample 13
19–16 LTS1_12	core 0 Load Tracking Sample 12
15–12 LTS1_11	core 0 Load Tracking Sample 11
11–8 LTS1_10	core 0 Load Tracking Sample 10
7–4 LTS1_9	core 0 Load Tracking Sample 9
LTS1_8	core 0 Load Tracking Sample 8

25.9.14 DVFS pattern 0 length (DVFSC_DVFSPT0)

Address: 0h base + 34h offset = 34h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														PT0A	FPTN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FPTN0															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

DVFSC_DVFSPT0 field descriptions

Field	Description
31–18 -	This field is reserved. reserved
17 PT0A	PT0A - Pattern 0 currently active (read-only) 1 active 0 non-active
FPTN0	FPTN0 - Frequency pattern 0 counter During period of this counter the frequency will be reduced from the EMA-detected level.

25.9.15 DVFS pattern 1 length (DVFSC_DVFSPT1)

Address: 0h base + 38h offset = 38h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														PT1A	FPTN1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FPTN1															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

DVFSC_DVFSPT1 field descriptions

Field	Description
31–18 -	This field is reserved. reserved
17 PT1A	PT1A - Pattern 1 currently active (read-only) 1 active 0 non-active
FPTN1	FPTN1 - Frequency pattern 1 counter During period of this counter the frequency will be set to the EMA-detected level.

25.9.16 DVFS pattern 2 length (DVFSC_DVFSPT2)

Address: 0h base + 3Ch offset = 3Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	P2THR						Reserved								PT2A	FPTN2
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FPTN2															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

DVFSC_DVFSPT2 field descriptions

Field	Description
31–26 P2THR	P2THR - Pattern 2 Threshold Threshold of current DVFS load (after EMA), for generating interrupts with PFUS indicators 110, 111. If the current performance is greater than the P2THR value, the interrupts will be generated. Otherwise, pattern delay will be counted, but without interrupt generation.
25–18 -	This field is reserved. reserved
17 PT2A	PT2A - Pattern 2 currently active (read-only) 1 active 0 non-active
FPTN2	FPTN2 - Frequency pattern 2 counter During period of this counter the frequency will be increased to higher, than detected by the EMA-detected level.

25.9.17 DVFS pattern 3 length (DVFSC_DVFSPT3)

Address: 0h base + 40h offset = 40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved														PT3A	FPTN3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FPTN3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

DVFSC_DVFSPT3 field descriptions

Field	Description
31–18 -	This field is reserved. reserved
17 PT3A	PT3A - Pattern 3 currently active (read-only) 1 active 0 non-active
FPTN3	FPTN3 - Frequency pattern 3 counter During period of this counter the frequency will be set to the EMA-detected level.

Chapter 26
General Purpose Input/Output (GPIO)

26.1 Overview

The GPIO general-purpose input/output peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs.

When configured as an output, it is possible to write to an internal register to control the state driven on the output pin. When configured as an input, it is possible to detect the state of the input by reading the state of an internal register. In addition, the GPIO peripheral can produce CORE interrupts.

The GPIO is one of the blocks controlling the IOMUX of the chip.

Figure 26-1 shows the chip multiplexing scheme.

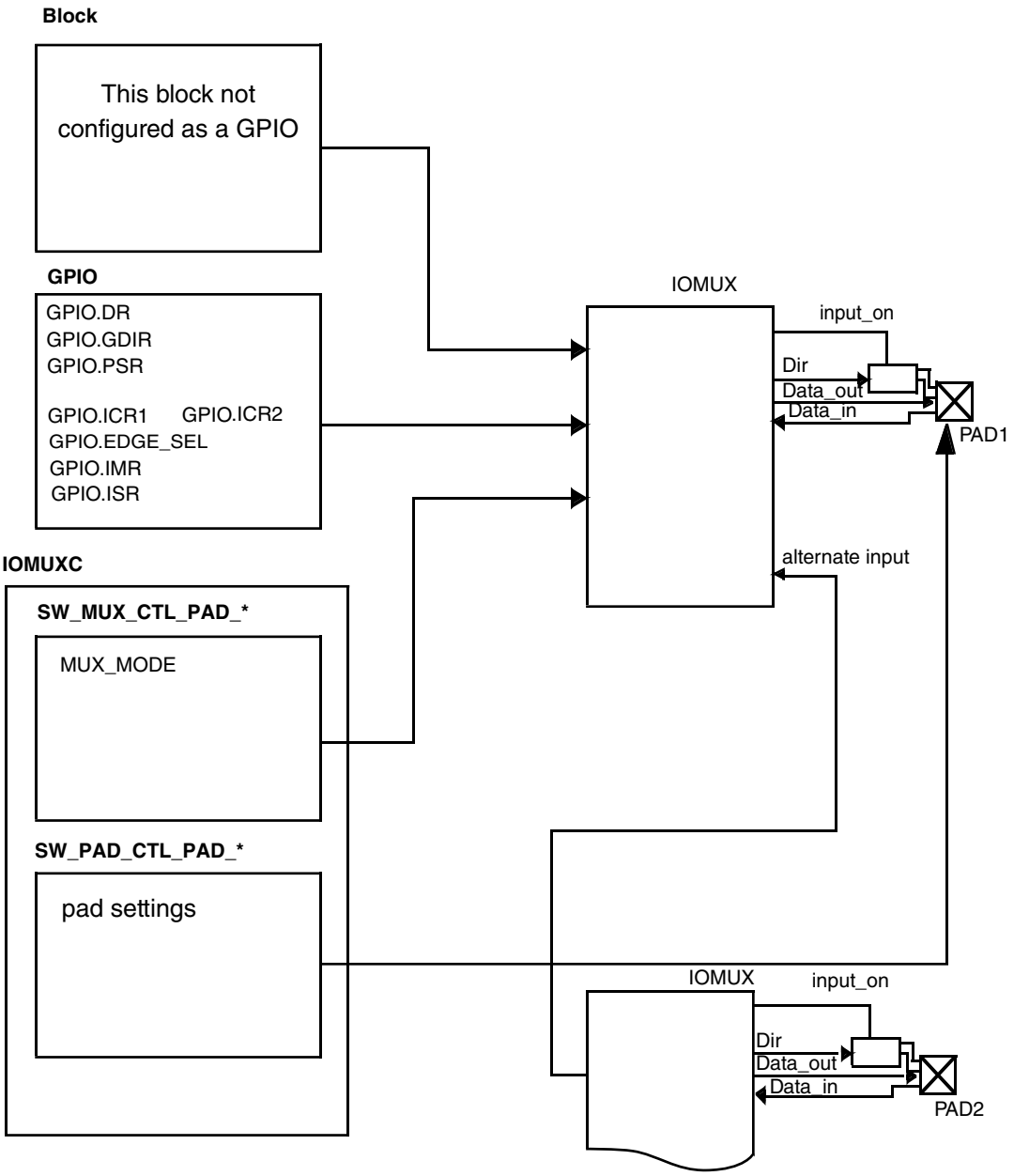


Figure 26-1. Chip IOMUX Scheme

The GPIO functionality is provided through eight registers, an edge-detect circuit, and interrupt generation logic.

The eight registers are:

- Data register (GPIO_DR)
- GPIO direction register (GPIO_GDIR)
- Pad sample register (GPIO_PSR)
- Interrupt control registers (GPIO_ICR1, GPIO_ICR2)

- Edge select register (GPIO_EDGE_SEL)
- Interrupt mask register (GPIO_IMR)
- Interrupt status register (GPIO_ISR)

These registers are described in detail in GPIO Memory Map/Register Definition.

Each GPIO input has a dedicated edge-detect circuit which can be configured through software to detect rising edges, falling edges, logic low-levels or logic high-levels on the input signals. The outputs of the edge detect circuits are optionally masked by setting the corresponding bit in the interrupt mask register (GPIO_IMR). These qualified outputs are OR'ed together to generate two one-bit interrupt lines:

- Combined interrupt indication for GPIOx signals 0 - 15
- Combined interrupt indication for GPIOx signals 16 - 31

In addition, GPIO1 provides visibility to each of its 8 low order interrupt sources (i.e. GPIO1 interrupt n, for n = 0 – 7). However, individual interrupt indications from other GPIOx are not available.

The GPIO edge detection is described further in Interrupt Control Unit.

The GPIO's overall functionality is described further in GPIO Functional Description.

26.1.1 Block Diagram

The GPIO subsystem contains 8 GPIO blocks which can generate and control up to 32 signals for general purpose.

A block diagram of the GPIO is shown in Figure 26-2 .

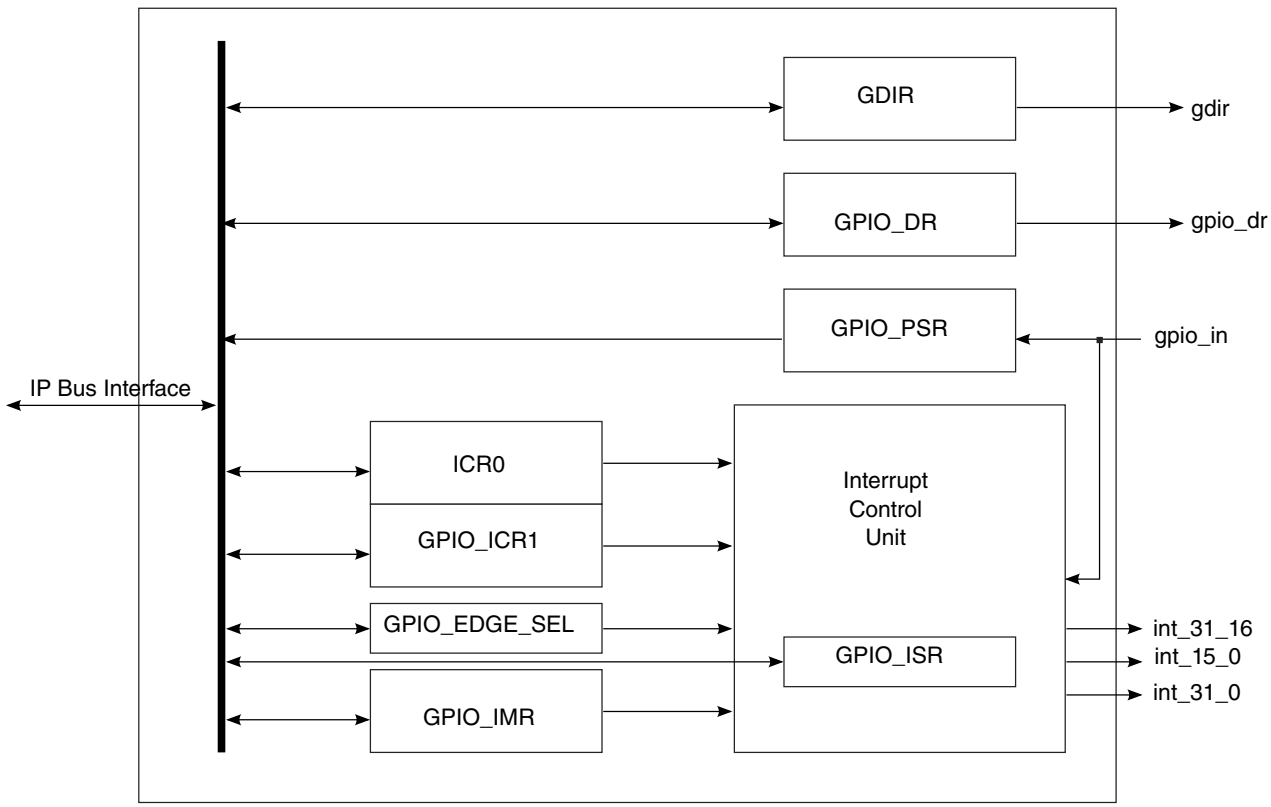


Figure 26-2. GPIO Block Diagram

26.1.2 Features

The GPIO includes the following features:

- General purpose input/output logic capabilities:
  - Drives specific data to output using the data register (GPIO_DR)
  - Controls the direction of the signal using the GPIO direction register (GPIO_GDIR)
  - Enables the core to sample the status of the corresponding inputs by reading the pad sample register (GPIO_PSR).
- GPIO interrupt capabilities:
  - Supports up to 32 interrupts

- Identifies interrupt edges
- Generates three active-high interrupts to the SoC interrupt controller

26.2 External Signals

The tables found here describe the external signals of GPIO.

Table 26-1. GPIO External Signals

Instance	Signal	Pad	Mode	Direction
GPIO1	IO0	GPIO1_IO00	ALT5	I/O
	IO1	GPIO1_IO01	ALT5	
	IO2	GPIO1_IO02	ALT5	
	IO3	GPIO1_IO03	ALT5	
	IO4	GPIO1_IO04	ALT5	
	IO5	GPIO1_IO05	ALT5	
	IO6	GPIO1_IO06	ALT5	
	IO7	GPIO1_IO07	ALT5	
	IO8	GPIO1_IO08	ALT5	
	IO9	GPIO1_IO09	ALT5	
	IO10	JTAG_MOD	ALT5	
	IO11	JTAG_TMS	ALT5	
	IO12	JTAG_TDO	ALT5	
	IO13	JTAG_TDI	ALT5	
	IO14	JTAG_TCK	ALT5	
	IO15	JTAG_TRST_B	ALT5	
	IO16	UART1_TX_DATA	ALT5	
	IO17	UART1_RX_DATA	ALT5	
	IO18	UART1_CTS_B	ALT5	
	IO19	UART1_RTS_B	ALT5	
	IO20	UART2_TX_DATA	ALT5	
	IO21	UART2_RX_DATA	ALT5	
	IO22	UART2_CTS_B	ALT5	
	IO23	UART2_RTS_B	ALT5	
	IO24	UART3_TX_DATA	ALT5	
	IO25	UART3_RX_DATA	ALT5	
	IO26	UART3_CTS_B	ALT5	
	IO27	UART3_RTS_B	ALT5	
	IO28	UART4_TX_DATA	ALT5	
	IO29	UART4_RX_DATA	ALT5	
	IO30	UART5_TX_DATA	ALT5	

Table continues on the next page...

Table 26-1. GPIO External Signals (continued)

Instance	Signal	Pad	Mode	Direction
	IO31	UART5_RX_DATA	ALT5	
GPIO2	IO0	ENET1_RX_DATA0	ALT5	I/O
	IO1	ENET1_RX_DATA1	ALT5	
	IO2	ENET1_RX_EN	ALT5	
	IO3	ENET1_TX_DATA0	ALT5	
	IO4	ENET1_TX_DATA1	ALT5	
	IO5	ENET1_TX_EN	ALT5	
	IO6	ENET1_TX_CLK	ALT5	
	IO7	ENET1_RX_ER	ALT5	
	IO8	ENET2_RX_DATA0	ALT5	
	IO9	ENET2_RX_DATA1	ALT5	
	IO10	ENET2_RX_EN	ALT5	
	IO11	ENET2_TX_DATA0	ALT5	
	IO12	ENET2_TX_DATA1	ALT5	
	IO13	ENET2_TX_EN	ALT5	
	IO14	ENET2_TX_CLK	ALT5	
	IO15	ENET2_RX_ER	ALT5	
	IO16	SD1_CMD	ALT5	
	IO17	SD1_CLK	ALT5	
	IO18	SD1_DATA0	ALT5	
	IO19	SD1_DATA1	ALT5	
	IO20	SD1_DATA2	ALT5	
	IO21	SD1_DATA3	ALT5	
GPIO3	IO0	LCD_CLK	ALT5	I/O
	IO1	LCD_ENABLE	ALT5	
	IO2	LCD_HSYNC	ALT5	
	IO3	LCD_VSYNC	ALT5	
	IO4	LCD_RESET	ALT5	
	IO5	LCD_DATA00	ALT5	
	IO6	LCD_DATA01	ALT5	
	IO7	LCD_DATA02	ALT5	
	IO8	LCD_DATA03	ALT5	
	IO9	LCD_DATA04	ALT5	
	IO10	LCD_DATA05	ALT5	
	IO11	LCD_DATA06	ALT5	
	IO12	LCD_DATA07	ALT5	
	IO13	LCD_DATA08	ALT5	
	IO14	LCD_DATA09	ALT5	
	IO15	LCD_DATA10	ALT5	

Table continues on the next page...

Table 26-1. GPIO External Signals (continued)

Instance	Signal	Pad	Mode	Direction
	IO16	LCD_DATA11	ALT5	
	IO17	LCD_DATA12	ALT5	
	IO18	LCD_DATA13	ALT5	
	IO19	LCD_DATA14	ALT5	
	IO20	LCD_DATA15	ALT5	
	IO21	LCD_DATA16	ALT5	
	IO22	LCD_DATA17	ALT5	
	IO23	LCD_DATA18	ALT5	
	IO24	LCD_DATA19	ALT5	
	IO25	LCD_DATA20	ALT5	
	IO26	LCD_DATA21	ALT5	
	IO27	LCD_DATA22	ALT5	
	IO28	LCD_DATA23	ALT5	
GPIO4	IO0	NAND_RE_B	ALT5	I/O
	IO1	NAND_WE_B	ALT5	
	IO2	NAND_DATA00	ALT5	
	IO3	NAND_DATA01	ALT5	
	IO4	NAND_DATA02	ALT5	
	IO5	NAND_DATA03	ALT5	
	IO6	NAND_DATA04	ALT5	
	IO7	NAND_DATA05	ALT5	
	IO8	NAND_DATA06	ALT5	
	IO9	NAND_DATA07	ALT5	
	IO10	NAND_ALE	ALT5	
	IO11	NAND_WP_B	ALT5	
	IO12	NAND_READY_B	ALT5	
	IO13	NAND_CE0_B	ALT5	
	IO14	NAND_CE1_B	ALT5	
	IO15	NAND_CLE	ALT5	
	IO16	NAND_DQS	ALT5	
	IO17	CSI_MCLK	ALT5	
	IO18	CSI_PIXCLK	ALT5	
	IO19	CSI_VSYNC	ALT5	
	IO20	CSI_HSYNC	ALT5	
	IO21	CSI_DATA00	ALT5	
	IO22	CSI_DATA01	ALT5	
	IO23	CSI_DATA02	ALT5	
	IO24	CSI_DATA03	ALT5	
	IO25	CSI_DATA04	ALT5	

Table continues on the next page...

Table 26-1. GPIO External Signals (continued)

Instance	Signal	Pad	Mode	Direction
	IO26	CSI_DATA05	ALT5	
	IO27	CSI_DATA06	ALT5	
	IO28	CSI_DATA07	ALT5	
GPIO5	IO0	SNVS_TAMPER0	No Muxing (ALT5)	I/O
	IO1	SNVS_TAMPER1	No Muxing (ALT5)	
	IO2	SNVS_TAMPER2	No Muxing (ALT5)	
	IO3	SNVS_TAMPER3	No Muxing (ALT5)	
	IO4	SNVS_TAMPER4	No Muxing (ALT5)	
	IO5	SNVS_TAMPER5	No Muxing (ALT5)	
	IO6	SNVS_TAMPER6	No Muxing (ALT5)	
	IO7	SNVS_TAMPER7	No Muxing (ALT5)	
	IO8	SNVS_TAMPER8	No Muxing (ALT5)	
	IO9	SNVS_TAMPER9	No Muxing (ALT5)	
	IO10	BOOT_MODE0	No Muxing (ALT5)	
	IO11	BOOT_MODE1	No Muxing (ALT5)	

26.3 Clocks

The table found here describes the clock sources for GPIO.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 26-2. GPIO Clocks

Clock name	Clock Root	Description
ipg_clk_s	ipg_clk_root	Peripheral access clock

26.4 GPIO Functional Description

This section provides a complete functional description of the block.

26.4.1 GPIO Function

A GPIO signal can operate as a general-purpose input/output when the IOMUX is set to GPIO mode. Each GPIO signal may be independently configured as either an input or an output using the GPIO direction register (GPIO_GDIR).

When configured as an output (GPIO_GDIR bit = 1), the value in the data bit in the GPIO data register (GPIO_DR) is driven on the corresponding GPIO line. When a signal is configured as an input (GPIO_GDIR bit = 0), the state of the input can be read from the corresponding GPIO_PSR bit.

26.4.2 GPIO pad structure

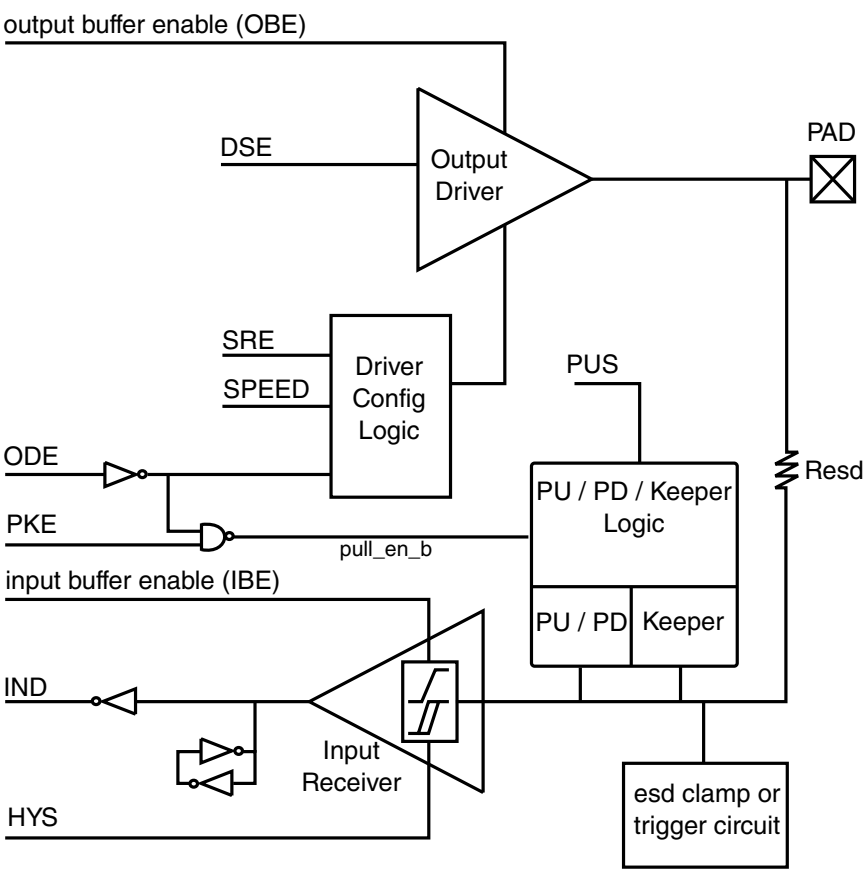


Figure 26-3. GPIO pad functional diagram

26.4.2.1 Input Driver

Input driver characteristics

- Selectable Schmitt trigger or CMOS input mode
- Keeper structure with buffer at the input receiver output to Core
- Receiver is tri-stated when I/O supply (OVDD) is powered down. (Keeper at receiver output keeps its previous state).

26.4.2.1.1 Schmitt trigger

The anti-jamming functionality of the Schmitt trigger is illustrated below.

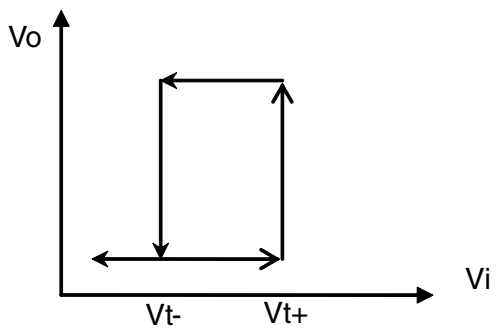


Figure 26-4. Schmitt trigger transfer characteristic

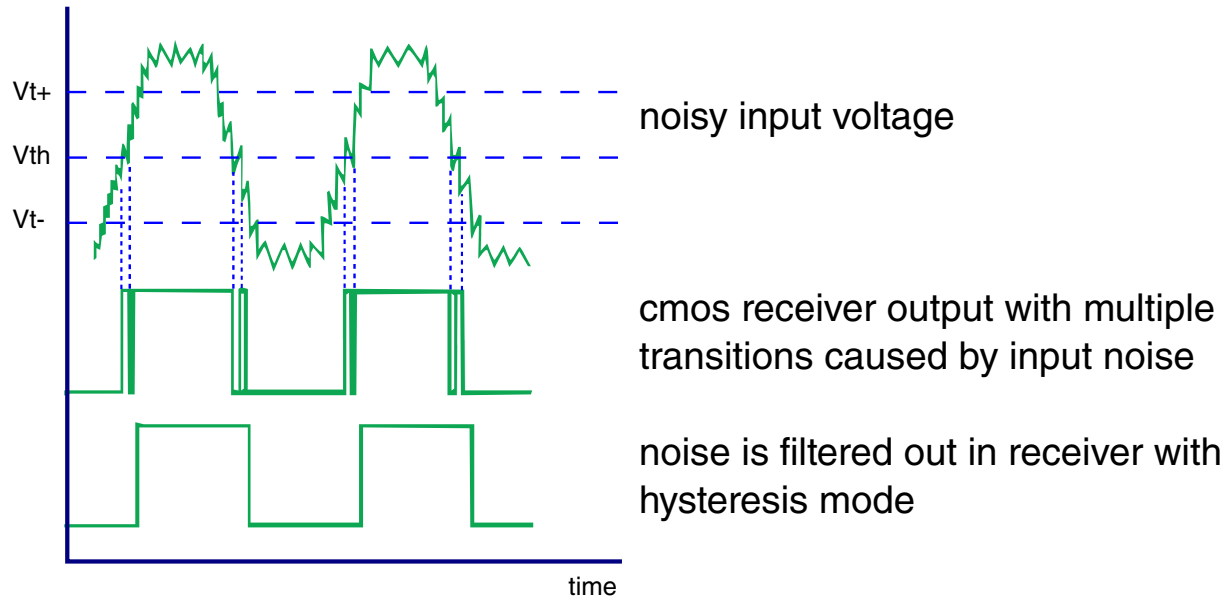


Figure 26-5. Receiver output in CMOS and hysteresis

26.4.2.1.2 Input keeper

A simple latch to hold the input value when OVDD is powered down, or the first inverter is tri-stated. Input buffer's keeper is always enabled for all the pads.

26.4.2.2 Output Driver

Output driver characteristics

- Selectable CMOS or open-drain output type
- Selectable pull-keeper enable signal to enable/disable the pull-up/down and output keeper
- Selectable pull-up resistors of 22K, 47K, 100K and a pull-down resistor of 100KOhm. Unsilicided P+ poly resistor is used to limit resistance variation to within +/- 20%.
- Pull-up, pull-down, and pad keeper are disabled in output mode.
- Seven drive strengths in each operating mode
- Additional 2-bit slew rate control to select between 50, 100, and 200 MHz IO cell operation range with reduced switching noise

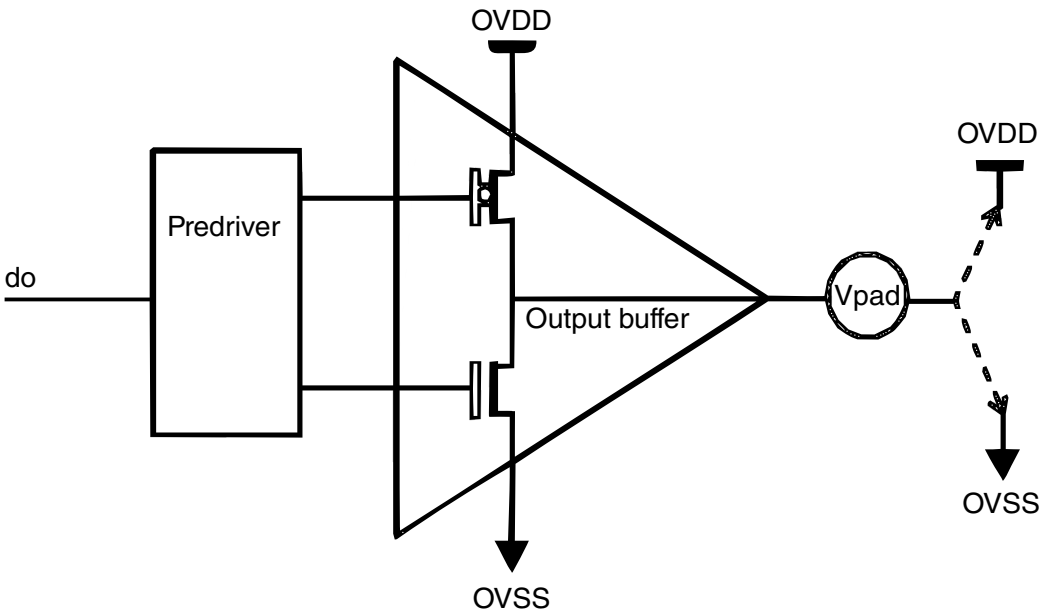


Figure 26-6. Output Driver Functional Diagram

26.4.2.2.1 Drive strength

Drive strength selection can be use to make the impedance matched and get better signal integrity.

26.4.2.2.2 Output keeper

A simple latch to hold the input value.

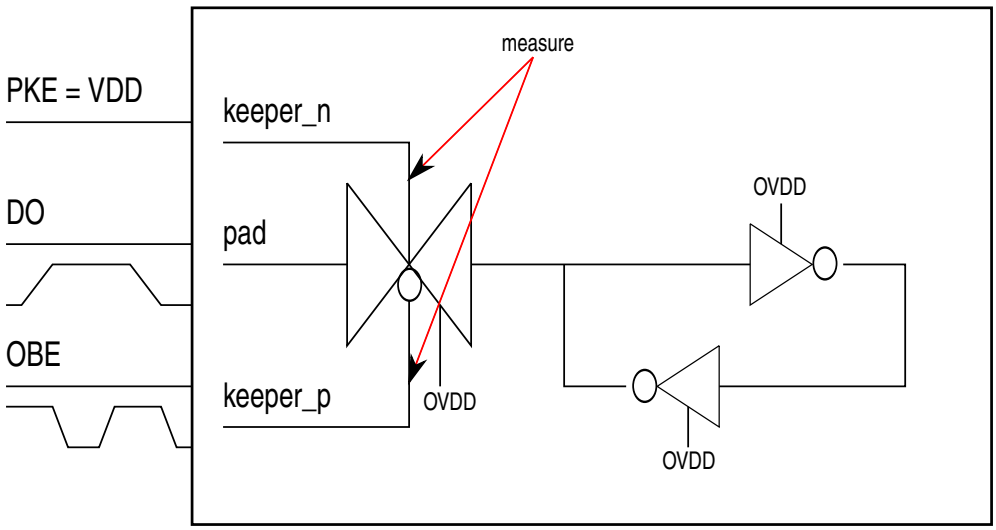


Figure 26-7. Keeper functional diagram

26.4.2.2.3 PU / PD / Keeper Logic

When Keeper is enabled, the pull-up and pull-down are disabled, and the output value of the pad depends on the Keeper. The output keeper is powered by OVDD. When the core VDD is powered down or the first inverter is tri-stated, the pad's state can be kept. Keeper and Pull can't be enabled together.

26.4.2.2.4 Open drain

Open drain is a circuit technique which allows multiple devices to communication over a single wire bi-directionally. Open drain drivers usually operate with an external or internal pull-up resistor that holds the signal line high until a device sinks enough current to pull the line low, usually used for a bus with multiple devices.

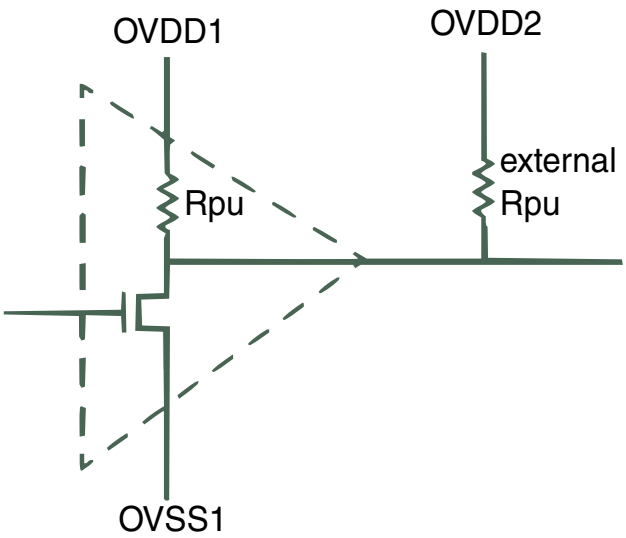


Figure 26-8. Output buffer in open drain mode

26.4.3 GPIO Programming

26.4.3.1 GPIO Read Mode

The programming sequence for reading input signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUX Controller (IOMUXC)).
2. Configure GPIO direction register to input (GPIO_GDIR[GDIR] set to 0b).
3. Read value from data reg*ister*/pad status register.

A pseudocode description to read [input3:input0] values is as follows:

```
// SET INPUTS TO GPIO MODE.
write sw_mux_ctl_<input0>_<input1>_<input2>_<input3>, 32'h00000000
// SET GDIR TO INPUT.
write GDIR[31:4,input3_bit, input2_bit, input1_bit, input0_bit,] 32'hxxxxxxx0
// READ INPUT VALUE FROM DR.
read DR
// READ INPUT VALUE FROM PSR.
read PSR
```

NOTE

While the GPIO direction is set to input (GPIO_GDIR = 0), a read access to GPIO_DR does not return GPIO_DR data. Instead, it returns the GPIO_PSR data, which is the corresponding input signal value.

26.4.3.2 GPIO Write Mode

The programming sequence for driving output signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUXC), also enable SION if need to read loopback pad value through PSR
2. Configure GPIO direction register to output (GPIO_GDIR[GDIR] set to 1b).
3. Write value to data register (GPIO_DR).

A pseudocode description to drive 4'b0101 on [output3:output0] is as follows:

```
// SET PADS TO GPIO MODE VIA IOMUX.
write sw_mux_ctl_pad_<output[0-3]>.mux_mode, <GPIO_MUX_MODE>
// Enable loopback so we can capture pad value into PSR in output mode
write sw_mux_ctl_pad_<output[0-3]>.sion, 1
// SET GDIR=1 TO OUTPUT BITS.
write GDIR[31:4,output3_bit,output2_bit, output1_bit, output0_bit,] 32'hxxxxxxxF
// WRITE OUTPUT VALUE=4'b0101 TO DR.
write DR, 32'hxxxxxxx5
// READ OUTPUT VALUE FROM PSR ONLY.
read_cmp PSR, 32'hxxxxxxx5
```

26.4.4 Interrupt Control Unit

In addition to the general-purpose input/output function, the edge-detect logic in the GPIO peripheral reflects whether a transition has occurred on a given GPIO signal that is configured as an input (GDIR bit = 0). The interrupt control registers (GPIO_ICR1 and GPIO_ICR2) may be used to independently configure the interrupt condition of each input signal (low-to-high transition, high-to-low transition, low, or high). For information about GPIO_ICR1 and GPIO_ICR2 settings, see GPIO Memory Map/Register Definition.

The interrupt control unit is built of 32 interrupt control subunits, where each subunit handles a single interrupt line.

26.5 GPIO Memory Map/Register Definition

There are eight 32-bit GPIO registers. All registers are accessible from the IP interface. Only 32-bit access is supported.

The GPIO memory map is shown in the following table.

GPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
209_C000	GPIO data register (GPIO1_DR)	32	R/W	0000_0000h	26.5.1/1155
209_C004	GPIO direction register (GPIO1_GDIR)	32	R/W	0000_0000h	26.5.2/1156
209_C008	GPIO pad status register (GPIO1_PSR)	32	R	0000_0000h	26.5.3/1156
209_C00C	GPIO interrupt configuration register1 (GPIO1_ICR1)	32	R/W	0000_0000h	26.5.4/1157
209_C010	GPIO interrupt configuration register2 (GPIO1_ICR2)	32	R/W	0000_0000h	26.5.5/1161
209_C014	GPIO interrupt mask register (GPIO1_IMR)	32	R/W	0000_0000h	26.5.6/1164
209_C018	GPIO interrupt status register (GPIO1_ISR)	32	w1c	0000_0000h	26.5.7/1165
209_C01C	GPIO edge select register (GPIO1_EDGE_SEL)	32	R/W	0000_0000h	26.5.8/1166
20A_0000	GPIO data register (GPIO2_DR)	32	R/W	0000_0000h	26.5.1/1155
20A_0004	GPIO direction register (GPIO2_GDIR)	32	R/W	0000_0000h	26.5.2/1156
20A_0008	GPIO pad status register (GPIO2_PSR)	32	R	0000_0000h	26.5.3/1156
20A_000C	GPIO interrupt configuration register1 (GPIO2_ICR1)	32	R/W	0000_0000h	26.5.4/1157
20A_0010	GPIO interrupt configuration register2 (GPIO2_ICR2)	32	R/W	0000_0000h	26.5.5/1161
20A_0014	GPIO interrupt mask register (GPIO2_IMR)	32	R/W	0000_0000h	26.5.6/1164
20A_0018	GPIO interrupt status register (GPIO2_ISR)	32	w1c	0000_0000h	26.5.7/1165
20A_001C	GPIO edge select register (GPIO2_EDGE_SEL)	32	R/W	0000_0000h	26.5.8/1166
20A_4000	GPIO data register (GPIO3_DR)	32	R/W	0000_0000h	26.5.1/1155
20A_4004	GPIO direction register (GPIO3_GDIR)	32	R/W	0000_0000h	26.5.2/1156
20A_4008	GPIO pad status register (GPIO3_PSR)	32	R	0000_0000h	26.5.3/1156
20A_400C	GPIO interrupt configuration register1 (GPIO3_ICR1)	32	R/W	0000_0000h	26.5.4/1157
20A_4010	GPIO interrupt configuration register2 (GPIO3_ICR2)	32	R/W	0000_0000h	26.5.5/1161
20A_4014	GPIO interrupt mask register (GPIO3_IMR)	32	R/W	0000_0000h	26.5.6/1164
20A_4018	GPIO interrupt status register (GPIO3_ISR)	32	w1c	0000_0000h	26.5.7/1165
20A_401C	GPIO edge select register (GPIO3_EDGE_SEL)	32	R/W	0000_0000h	26.5.8/1166
20A_8000	GPIO data register (GPIO4_DR)	32	R/W	0000_0000h	26.5.1/1155
20A_8004	GPIO direction register (GPIO4_GDIR)	32	R/W	0000_0000h	26.5.2/1156
20A_8008	GPIO pad status register (GPIO4_PSR)	32	R	0000_0000h	26.5.3/1156
20A_800C	GPIO interrupt configuration register1 (GPIO4_ICR1)	32	R/W	0000_0000h	26.5.4/1157
20A_8010	GPIO interrupt configuration register2 (GPIO4_ICR2)	32	R/W	0000_0000h	26.5.5/1161
20A_8014	GPIO interrupt mask register (GPIO4_IMR)	32	R/W	0000_0000h	26.5.6/1164
20A_8018	GPIO interrupt status register (GPIO4_ISR)	32	w1c	0000_0000h	26.5.7/1165
20A_801C	GPIO edge select register (GPIO4_EDGE_SEL)	32	R/W	0000_0000h	26.5.8/1166
20A_C000	GPIO data register (GPIO5_DR)	32	R/W	0000_0000h	26.5.1/1155
20A_C004	GPIO direction register (GPIO5_GDIR)	32	R/W	0000_0000h	26.5.2/1156
20A_C008	GPIO pad status register (GPIO5_PSR)	32	R	0000_0000h	26.5.3/1156
20A_C00C	GPIO interrupt configuration register1 (GPIO5_ICR1)	32	R/W	0000_0000h	26.5.4/1157
20A_C010	GPIO interrupt configuration register2 (GPIO5_ICR2)	32	R/W	0000_0000h	26.5.5/1161
20A_C014	GPIO interrupt mask register (GPIO5_IMR)	32	R/W	0000_0000h	26.5.6/1164

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20A_C018	GPIO interrupt status register (GPIO5_ISR)	32	w1c	0000_0000h	26.5.7/1165
20A_C01C	GPIO edge select register (GPIO5_EDGE_SEL)	32	R/W	0000_0000h	26.5.8/1166

26.5.1 GPIO data register (GPIOx_DR)

The 32-bit GPIO_DR register stores data that is ready to be driven to the output lines. If the IOMUXC is in GPIO mode and a given GPIO direction bit is set, then the corresponding DR bit is driven to the output. If a given GPIO direction bit is cleared, then a read of GPIO_DR reflects the value of the corresponding signal.Two wait states are required in read access for synchronization.

The results of a read of a DR bit depends on the IOMUXC input mode settings and the corresponding GDIR bit as follows:

- If GDIR[*n*] is set and IOMUXC input mode is GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUXC input mode is GPIO, then reading DR[*n*] returns the corresponding input signal's value.
- If GDIR[*n*] is set and IOMUXC input mode is not GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUXC input mode is not GPIO, then reading DR[*n*] always returns zero.

Address: Base address + 0h offset

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

GPIOx_DR field descriptions

Field	Description
DR	Data bits. This register defines the value of the GPIO output when the signal is configured as an output (GDIR[n]=1). Writes to this register are stored in a register. Reading GPIO_DR returns the value stored in the register if the signal is configured as an output (GDIR[n]=1), or the input signal's value if configured as an input (GDIR[n]=0). **NOTE:** The I/O multiplexer must be configured to GPIO mode for the GPIO_DR value to connect with the signal. Reading the data register with the input path disabled always returns a zero value.

26.5.2 GPIO direction register (GPIOx_GDIR)

GPIO_GDIR functions as direction control when the IOMUXC is in GPIO mode. Each bit specifies the direction of a one-bit signal. The mapping of each DIR bit to a corresponding SoC signal is determined by the SoC's pin assignment and the IOMUX table. For more details consult the IOMUXC chapter.

Address: Base address + 4h offset

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

GPIOx_GDIR field descriptions

Field	Description
GDIR	GPIO direction bits. Bit n of this register defines the direction of the GPIO[n] signal. **NOTE:** GPIO_GDIR affects only the direction of the I/O signal when the corresponding bit in the I/O MUX is configured for GPIO. 0 **INPUT** — GPIO is configured as input. 1 **OUTPUT** — GPIO is configured as output.

26.5.3 GPIO pad status register (GPIOx_PSR)

GPIO_PSR is a read-only register. Each bit stores the value of the corresponding input signal (as configured in the IOMUX). This register is clocked with the ipg_clk_s clock, meaning that the input signal is sampled only when accessing this location. Two wait states are required any time this register is accessed for synchronization.

Address: Base address + 8h offset

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

GPIOx_PSR field descriptions

Field	Description
PSR	GPIO pad status bits (status bits). Reading GPIO_PSR returns the state of the corresponding input signal.

GPIOx_PSR field descriptions (continued)

Field	Description
	Settings: **NOTE:** The IOMUXC must be configured to GPIO mode for GPIO_PSR to reflect the state of the corresponding signal.

26.5.4 GPIO interrupt configuration register1 (GPIOx_ICR1)

GPIO_ICR1 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR15		ICR14		ICR13		ICR12		ICR11		ICR10		ICR9		ICR8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR7		ICR6		ICR5		ICR4		ICR3		ICR2		ICR1		ICR0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_ICR1 field descriptions

Field	Description
31–30 ICR15	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 15. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
29–28 ICR14	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 14. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
27–26 ICR13	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 13.

Table continues on the next page...

GPIOx_ICR1 field descriptions (continued)

Field	Description
	Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
25–24 ICR12	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 12. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
23–22 ICR11	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 11. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
21–20 ICR10	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 10. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
19–18 ICR9	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 9. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
17–16 ICR8	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 8. Settings:

Table continues on the next page...

GPIOx_ICR1 field descriptions (continued)

Field	Description
	Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
15–14 ICR7	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 7. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
13–12 ICR6	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 6. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
11–10 ICR5	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 5. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
9–8 ICR4	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 4. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
7–6 ICR3	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 3. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows:

Table continues on the next page...

GPIOx_ICR1 field descriptions (continued)

Field	Description
	00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
5–4 ICR2	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 2. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
3–2 ICR1	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 1. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
ICR0	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 0. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

26.5.5 GPIO interrupt configuration register2 (GPIOx_ICR2)

GPIO_ICR2 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR31		ICR30		ICR29		ICR28		ICR27		ICR26		ICR25		ICR24	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR23		ICR22		ICR21		ICR20		ICR19		ICR18		ICR17		ICR16	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_ICR2 field descriptions

Field	Description
31–30 ICR31	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 31. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
29–28 ICR30	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 30. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
27–26 ICR29	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 29. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR2 field descriptions (continued)

Field	Description
25–24 ICR28	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 28. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
23–22 ICR27	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 27. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
21–20 ICR26	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 26. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
19–18 ICR25	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 25. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
17–16 ICR24	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 24. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR2 field descriptions (continued)

Field	Description
15–14 ICR23	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 23. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
13–12 ICR22	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 22. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
11–10 ICR21	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 21. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
9–8 ICR20	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 20. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
7–6 ICR19	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 19. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR2 field descriptions (continued)

Field	Description
5–4 ICR18	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 18. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
3–2 ICR17	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 17. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
ICR16	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 16. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

26.5.6 GPIO interrupt mask register (GPIOx_IMR)

GPIO_IMR contains masking bits for each interrupt line.

Address: Base address + 14h offset

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

GPIOx_IMR field descriptions

Field	Description
IMR	Interrupt Mask bits. This register is used to enable or disable the interrupt function on each of the 32 GPIO signals.

GPIOx_IMR field descriptions (continued)

Field	Description
	Settings: Bit IMR[n] (n=0...31) controls interrupt n as follows: 0 **UNMASKED** — Interrupt n is disabled. 1 **MASKED** — Interrupt n is enabled.

26.5.7 GPIO interrupt status register (GPIOx_ISR)

The GPIO_ISR functions as an interrupt status indicator. Each bit indicates whether an interrupt condition has been met for the corresponding input signal. When an interrupt condition is met (as determined by the corresponding interrupt condition register field), the corresponding bit in this register is set. Two wait states are required in read access for synchronization. One wait state is required for reset.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_ISR field descriptions

Field	Description
ISR	Interrupt status bits - Bit n of this register is asserted (active high) when the active condition (as determined by the corresponding ICR bit) is detected on the GPIO input and is waiting for service. The value of this register is independent of the value in GPIO_IMR. When the active condition has been detected, the corresponding bit remains set until cleared by software. Status flags are cleared by writing a 1 to the corresponding bit position.

26.5.8 GPIO edge select register (GPIOx_EDGE_SEL)

GPIO_EDGE_SEL may be used to override the ICR registers' configuration. If the GPIO_EDGE_SEL bit is set, then a rising edge or falling edge in the corresponding signal generates an interrupt. This register provides backward compatibility. On reset all bits are cleared (ICR is not overridden).

Address: Base address + 1Ch offset

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

GPIOx_EDGE_SEL field descriptions

Field	Description
GPIO_EDGE_SEL	Edge select. When GPIO_EDGE_SEL[*n*] is set, the GPIO disregards the ICR[*n*] setting, and detects any edge on the corresponding input signal.

Chapter 27
General Purpose Media Interface (GPMI)

27.1 Overview

The GPMI controller is a flexible interface to supporting up to four NAND flash chip selects.

- ONFI 2.2, DDR Mode, Samsung / Toshiba Toggle NAND protocol compatible.
- Fully configurable address and command behavior, providing support for future devices not yet specified.

The GPMI resides on the APBH. The GPMI also provides an interface to the BCH module to allow direct parity processing.

Registers are clocked on the HCLK domain. The I/O and pin timing are clocked on a dedicated GPMICLK domain. GPMICLK can be set to maximize I/O performance.

The following figure shows a block diagram of the GPMI controller.

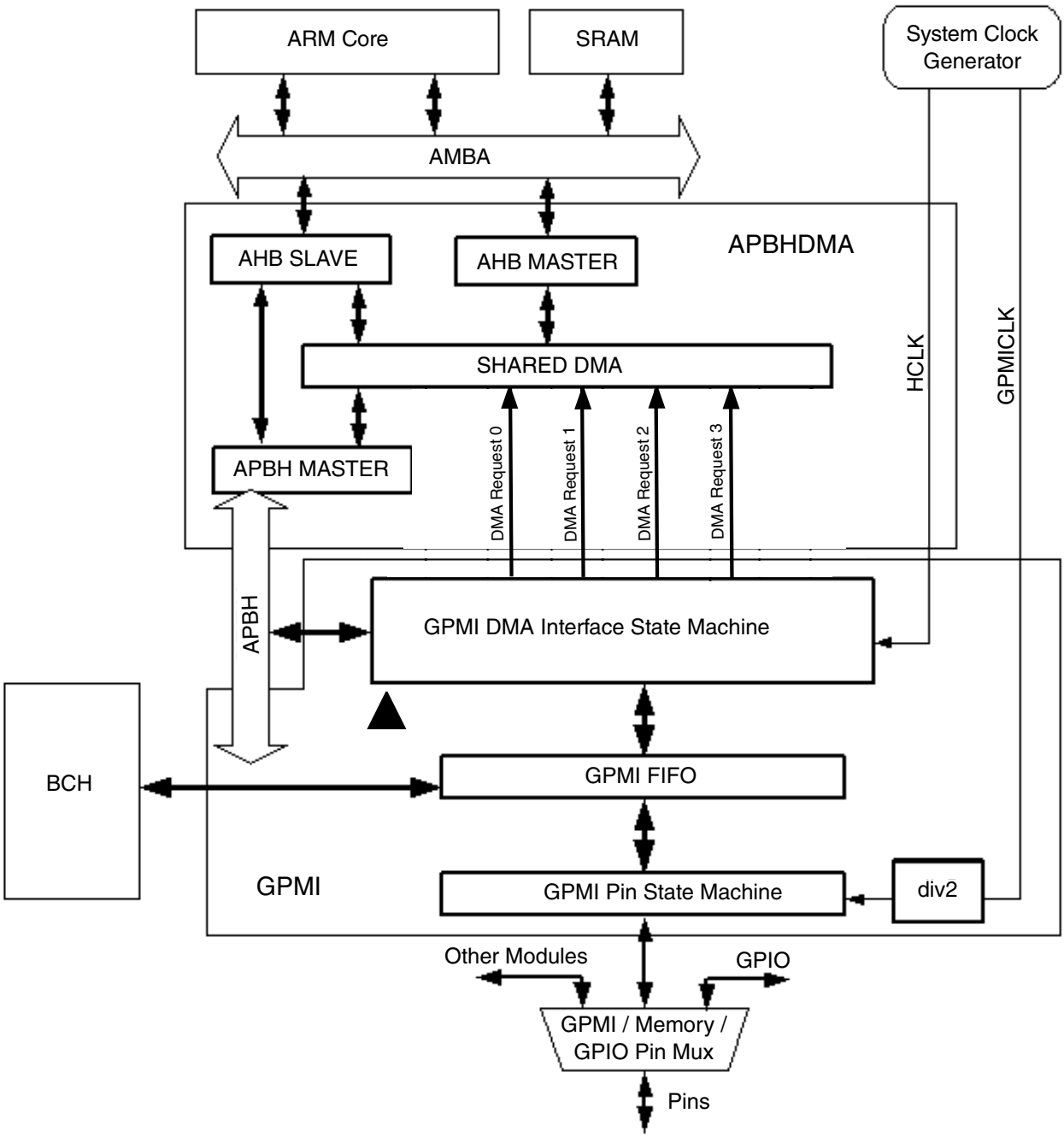


Figure 27-1. General-Purpose Media Interface Controller Block Diagram

27.2 External Signals

The table found here describes the external signals of GPMI.

Table 27-1. GPMI External Signals

Signal	Description	Pad	Mode	Direction
NAND_ALE	Address latch enable signal	NAND_ALE	ALT0	O
NAND_CE0_B	Chip enable signal	NAND_CE0_B	ALT0	O
NAND_CE1_B	Chip enable signal	NAND_CE1_B	ALT0	O
NAND_CE2_B	Chip enable signal	CSI_MCLK	ALT2	O
NAND_CE3_B	Chip enable signal	CSI_PIXCLK	ALT2	O
NAND_CLE	Command latch enable signal	NAND_CLE	ALT0	O
NAND_DATA00	Data signal	NAND_DATA00	ALT0	IO
NAND_DATA01	Data signal	NAND_DATA01	ALT0	IO
NAND_DATA02	Data signal	NAND_DATA02	ALT0	IO
NAND_DATA03	Data signal	NAND_DATA03	ALT0	IO
NAND_DATA04	Data signal	NAND_DATA04	ALT0	IO
NAND_DATA05	Data signal	NAND_DATA05	ALT0	IO
NAND_DATA06	Data signal	NAND_DATA06	ALT0	IO
NAND_DATA07	Data signal	NAND_DATA07	ALT0	IO
NAND_DQS	DQS signal	NAND_DQS	ALT0	IO
NAND_READY_B	Ready signal	NAND_READY_B	ALT0	IO
NAND_RE_B	Read enable signal	NAND_RE_B	ALT0	O
NAND_WE_B	Write enable signal	NAND_WE_B	ALT0	O
NAND_WP_B	Wait polarity signal	NAND_WP_B	ALT0	O

27.3 Clocks

The table found here describes the clock sources for GPMI.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 27-2. GPMI Clocks

Clock name	Clock Root	Description
bch_input_apb_clk		BCH to APBH input clock
gpmi_bch_input_bch_clk		BCH input clock
gpmi_bch_input_gpmi_io_clk		GPMI IO input clock
gpmi_input_apb_clk		GPMI to APBH clock

27.4 GPMI NAND Mode

The general-purpose media interface has several features to efficiently support NAND:

- Individual chip select pins and ganged ready/busy pin for up to four NANDs.
- Individual state machine and DMA channel for each chip select.
- Special command modes work with DMA controller to perform all normal NAND functions without CPU intervention.
- Configurable timing based on a dedicated clock allows optimal balance of high NAND performance and low system power.

GPMI and DMA have been designed to handle complex multi-page operations without CPU intervention. The DMA uses a linked descriptor function with branching capability to automatically handle all of the operations needed to read/write multiple pages:

- **Data/Register Read/Write**-The GPMI can be programmed to read or write multiple cycles to the NAND address, command or data registers.
- **Wait for NAND Ready**-The GPMI's Wait-for-Ready mode can monitor the ready/busy signal of a single NAND flash and signal the DMA when the device has become ready. It also has a time-out counter and can indicate to the DMA that a time-out error has occurred. The DMAs can conditionally branch to a different descriptor in the case of an error.
- **Check Status**-The Read-and-Compare mode allows the GPMI to check NAND status against a reference. If an error is found, the GPMI can instruct the DMA to branch to an alternate descriptor, which attempts to fix the problem or asserts a CPU IRQ.

27.4.1 Multiple NAND Support

The GPMI supports up to four NAND chip selects, with ganged ready/busy pins.

Since they share a data bus and control lines, the GPMI can only actively communicate with a single NAND at a time. However, all NANDs can concurrently perform internal read, write, or erase operations. With fast NAND flash and software support for concurrent NAND operations, this architecture allows the total throughput to approach the data bus speed, which can be as high as 50 MB/s (8-bit bus running at 50 MHz single clock edge) in asynchronous mode and 200MB/s (8-bit bus running at 100MHz both clock edges) in Source Synchronous mode.

There are two options for controlling the four NAND chip selects via the DMA interface. The first option is the one to one mapping, where the each DMA channel is tied to its own NAND chip select. For example DMA channel 'n' accesses only NAND attached to chip select 'n'. The second option is the decoupled mode where a DMA channel can access any or all NAND chip selects connected to the GPMI. A DMA channel will signify the NAND chip select it wants to access by writing its chip select value in the GPMI_CTRL0[CS] field and setting the GPMI_CTRL1[DECOUPLE_CS] to 1. This option is useful if software chooses to use only one DMA channel to access all the attached NAND devices.

27.4.2 GPMI NAND Timing and Clocking

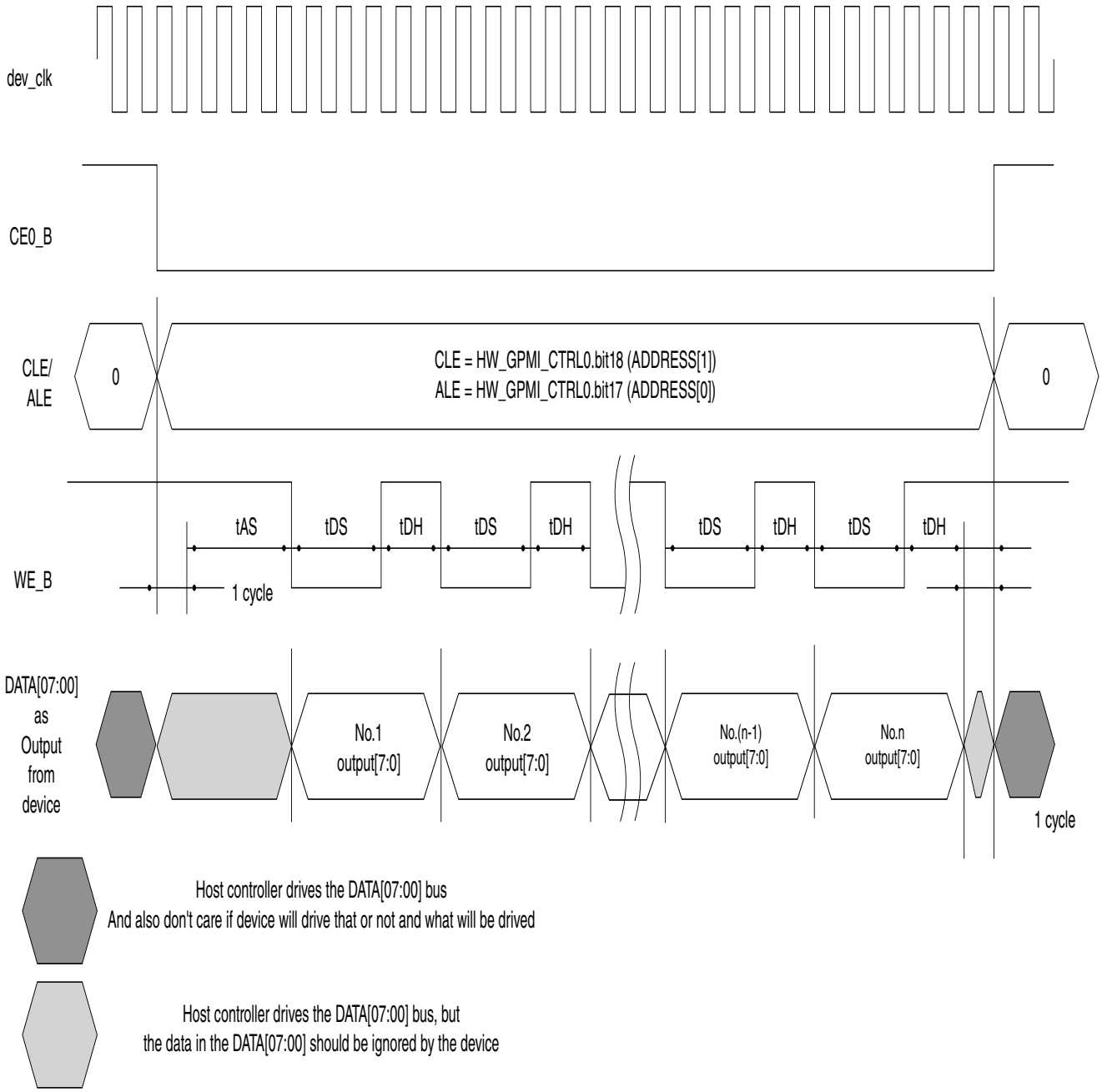
The dedicated clock, GPMICLK, is used as a timing reference for NAND flash I/O. Since various NANDs have different timing requirements, GPMICLK may need to be adjusted for each application.

While the actual pin timings are limited by the NAND chips used and the I/O pad configuration, the GPMI can support data bus speeds of up to 200 MHz x 8 bits. The actual read/write strobe timing parameters are adjusted as indicated in the register descriptions in Memory Map.

27.4.3 Basic NAND Timing

27.4.3.1 NAND Asynchronous Timing

Figure 27-3 and illustrates the operation of the output (from host to device) timing parameters in NAND ONFI asynchronous mode.



- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup: in this example, Address_Setup = 4, tAS is equal to 4 dev_clk cycles.
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup; in this example, Data_Setup = 3, tDS is equal to 3 dev_clk cycles
- tDH is configuarble by programming HW_GPMI_TIMING0 Data_Hold: in this example, Data_Hold = 2, tDH is equal to 2 dev_clk cycles
- tAS/tDS/tDH will extend, if the output data is not ready in device fifo.

Figure 27-2. Asynchronous Mode Basic Write Timing Diagram (command write, address write, or data write

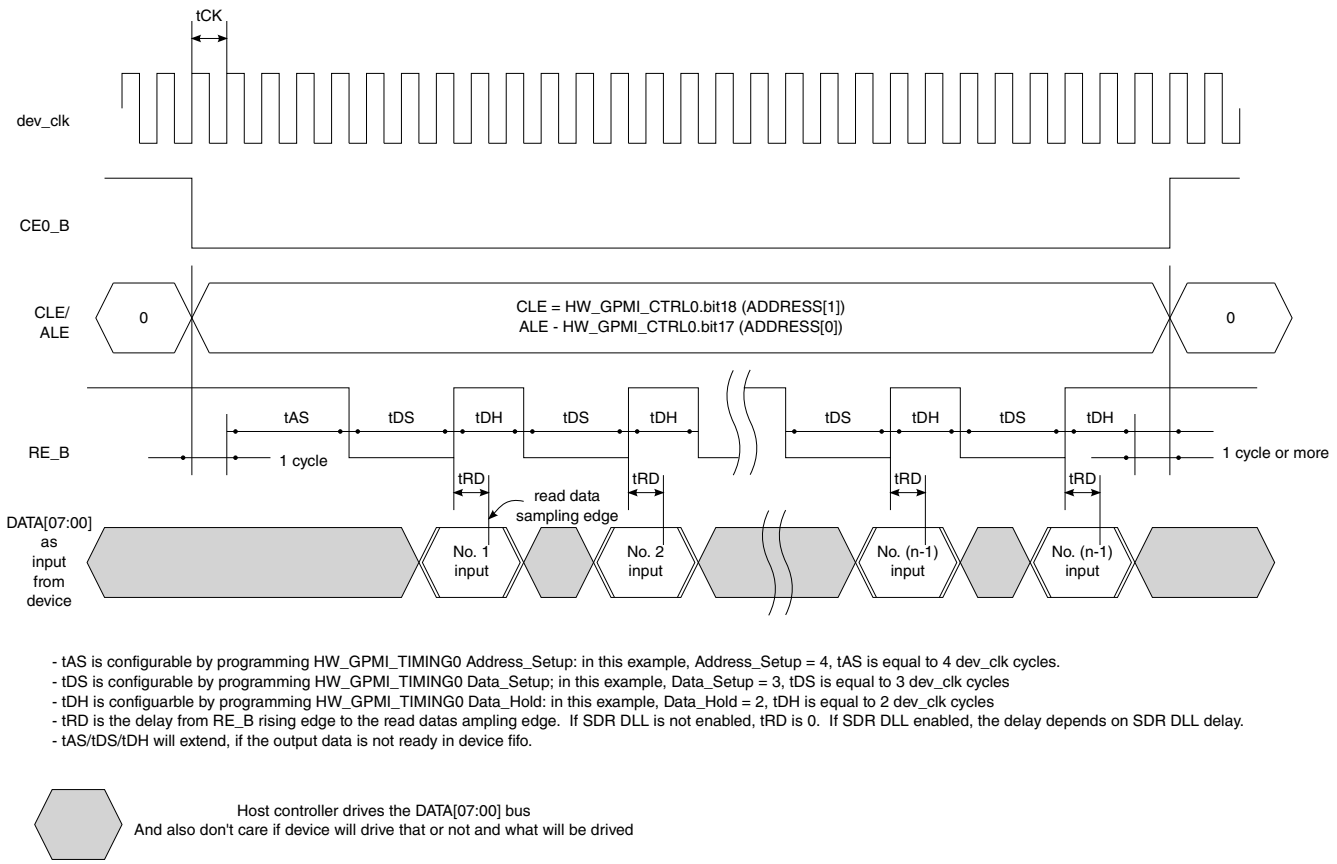


Figure 27-3. ONFI Asynchronous Mode Basic Read Timing Diagram (data read)

27.4.3.2 NAND Asynchronous EDO Mode Timing

In high-speed NANDS, the read data may not be valid until after the read strobe (NAND_RE_B) deasserts. This is the case when the minimum tDS is programmed to achieve higher bandwidth.

The GPMI implements a feedback read strobe to sample the read data. The feedback read strobe can be delayed to support fast nand EDO (Extended Data Out) timing where the read strobe may deassert before the read data is valid, and read data is valid for some time after read strobe.

Nand EDO timings is applied typically for read cycle frequency above 33 MHz. See Figure 27-4.

The GPMI provides control over the amount of delay applied to the feedback read strobe. This delay depends on the maximum read access time (tREA) of the nand and the read pulse width (tRP) used to access the nand. tRP is specified by

GPMI_TIMING0[DATA_SETUP] register. When (tREA + 4ns) is less than tRP, no delay is required to sample to nand read data. (The 4ns provides adequate data setup time for the GPMI.) In this case set GPMI_CTRL1[HALF_PERIOD] = 0; GPMI_CTRL1[RDN_DELAY] = 0; GPMI_CTRL1[DLL_ENABLE] = 0.

When (tREA + 4ns) is greater than or equal to tRP, a delay of the feedback read strobe is required to sample to nand read data. This delay is equal to the difference between these two timings:

DELAY = tREA + 4ns - tRP.

Since the GPMI delay chain is limited to ns maximum, if DELAY > ns then increase tRP by increasing the value of GPMI_TIMING0[DATA_SETUP] until DELAY is less than or equal to ns.

The GPMI programming for this DELAY depends on the GPMICLK period. The GPMI DLL will not function properly if the GPMICLK period is greater than ns: disable the DLL if this is the case. If the GPMICLK period is greater than ns (and not greater than ns), set the GPMI_CTRL1[HALF_PERIOD]=1; This will cause the DLL reference period (RP) to be one-half of the GPMICLK period. If the GPMICLK period is ns or less then set the GPMI_CTRL1[HALF_PERIOD]=0; This will cause the DLL reference period (RP) to be equal to the GPMICLK period. DELAY is a multiple (0 to 1.875) of RP.

The GPMI_CTRL1[RDN_DELAY] is encoded as a 1-bit integer and 3-bit fraction delay factor. DELAY is a multiple of the delay factor and the reference period. See table below for details.

Table 27-3. RDN DELAY

HW_GPMI_CTRL1[RDN_DELAY]	Delay Factor
0	0.000
1	0.125
2	0.250
3	0.375
4	0.500
5	0.625
6	0.750
7	0.875
8	1.000
9	1.125
10	1.250
11	1.375
12	1.500

Table continues on the next page...

Table 27-3. RDN DELAY (continued)

HW_GPMI_CTRL1[RDN_DELAY]	Delay Factor
13	1.625
14	1.750
15	1.875

DELAY = DelayFactor x RP or DELAY = GPMI_CTRL1[RDN_DELAY] x 0.125 x RP.

Use this equation to calculate the value for GPMI_CTRL1[RDN_DELAY]. Then set GPMI_CTRL1[DLL_ENABLE]=1.

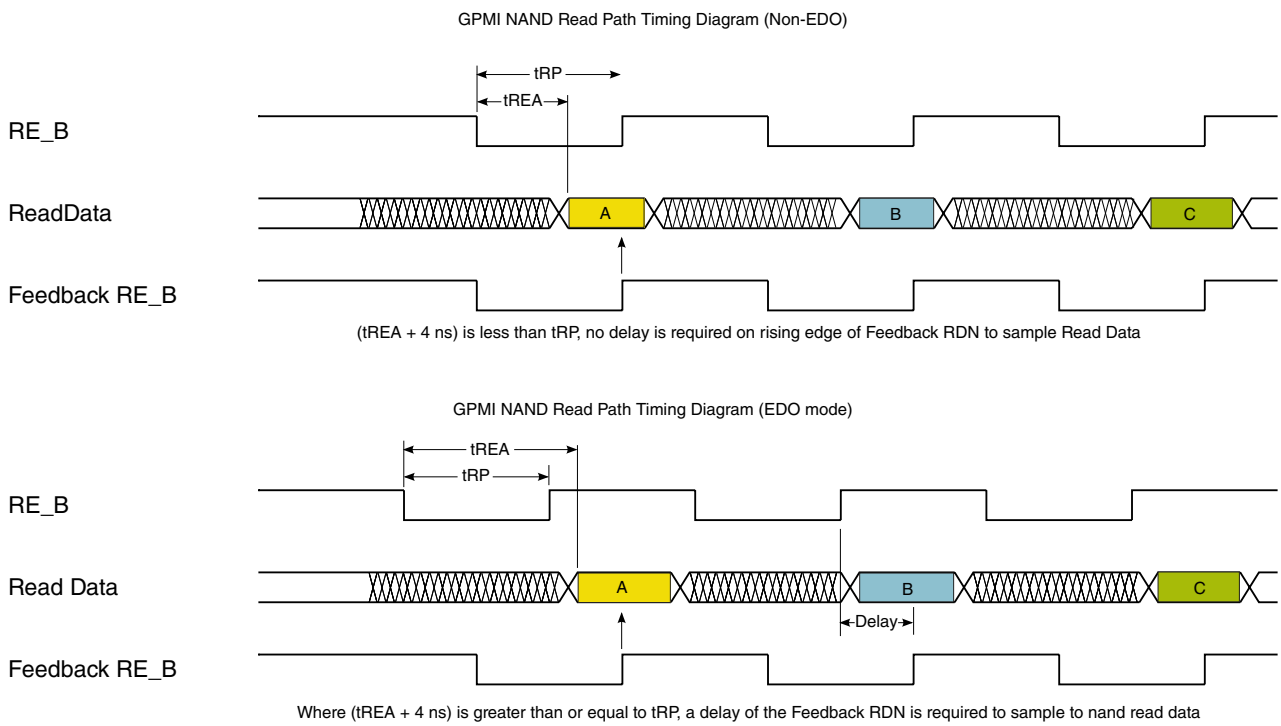


Figure 27-4. NAND Read Path Timing

For example, a NAND with tREAmax = 20 ns, tRPmin = 12 ns, and tRCmin = 25 ns (read cycle time) may be programmed as follows:

- GPMICLK clock frequency: Consider 480/6 = 80 MHz which is 12.5 ns clock period. This is too close to the minimum NAND spec if we program the data setup and hold to 1 GPMICLK cycle. Consider 480/7=68.57 MHz which is 14.58 ns clock period. With data setup and hold set to 1, we have a tRP of 14.58ns and a tRC of 29.16 ns (good margins).

- Since (tREA +4ns) is greater than tRP, required DELAY = tREA + 4ns - tRP = 20 + 4 - 14.58ns = 9.42 ns.
- GPMI_CTRL1[HALF_PERIOD] =, since GPMICLK period is ns. So RP=GPMICLK period = ns.
- DELAY = GPMI_CTRL1[RDN_DELAY] x 0.125 x RP. 9.42 ns = GPMI_CTRL1[RDN_DELAY] x 0.125 x ns. GPMI_CTRL1[RDN_DELAY] =

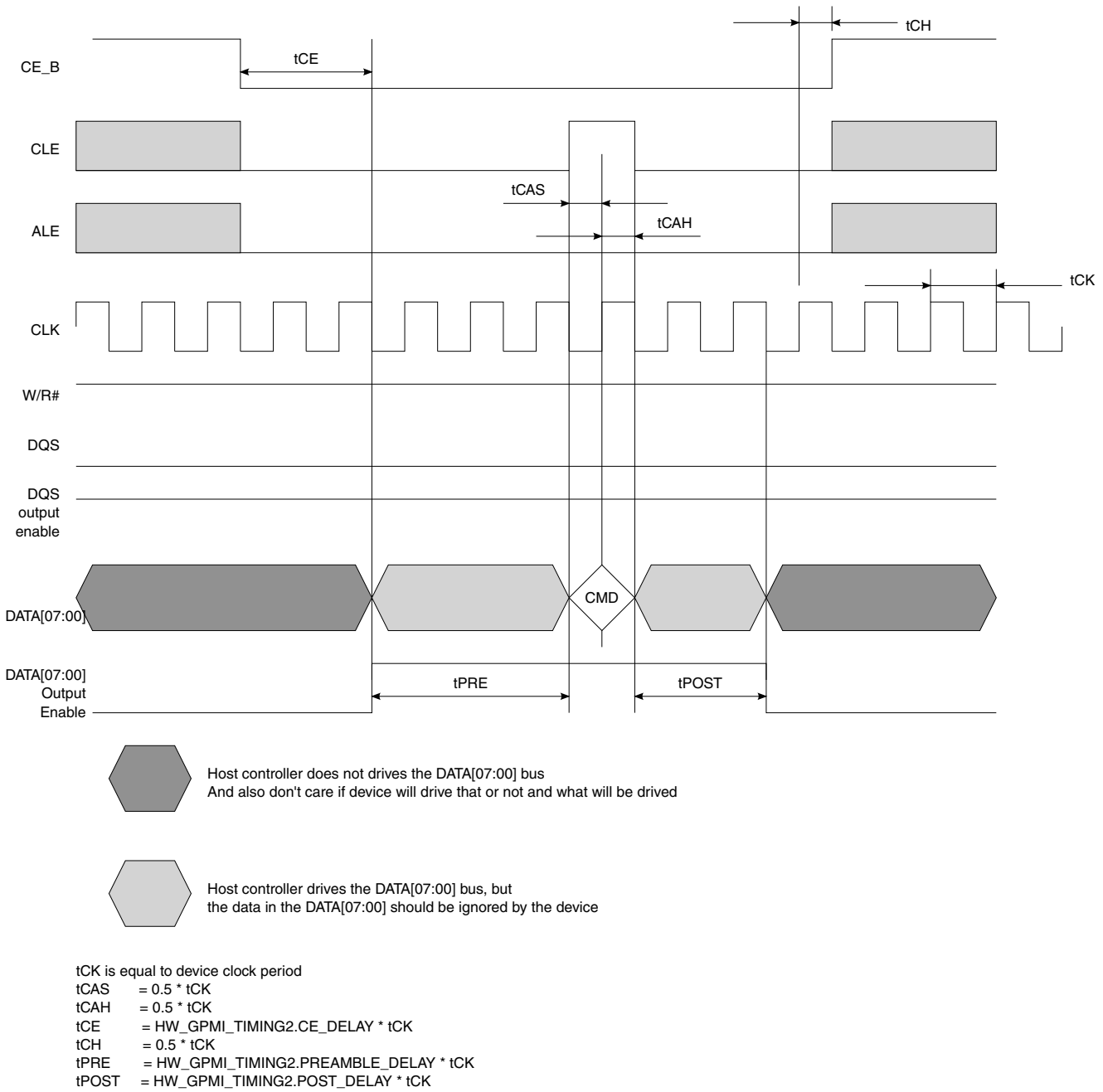
NOTE

It is recommended that the drive strength of NAND_RE_B and NAND_WE_B output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND interface read and write cycle times are below 30 ns. The other GPMI pins may remain at 4 mA, since their frequency is only up to half that of NAND_RE_B and NAND_WE_B.

27.4.3.3 NAND ONFI Source Synchronous Mode Timing

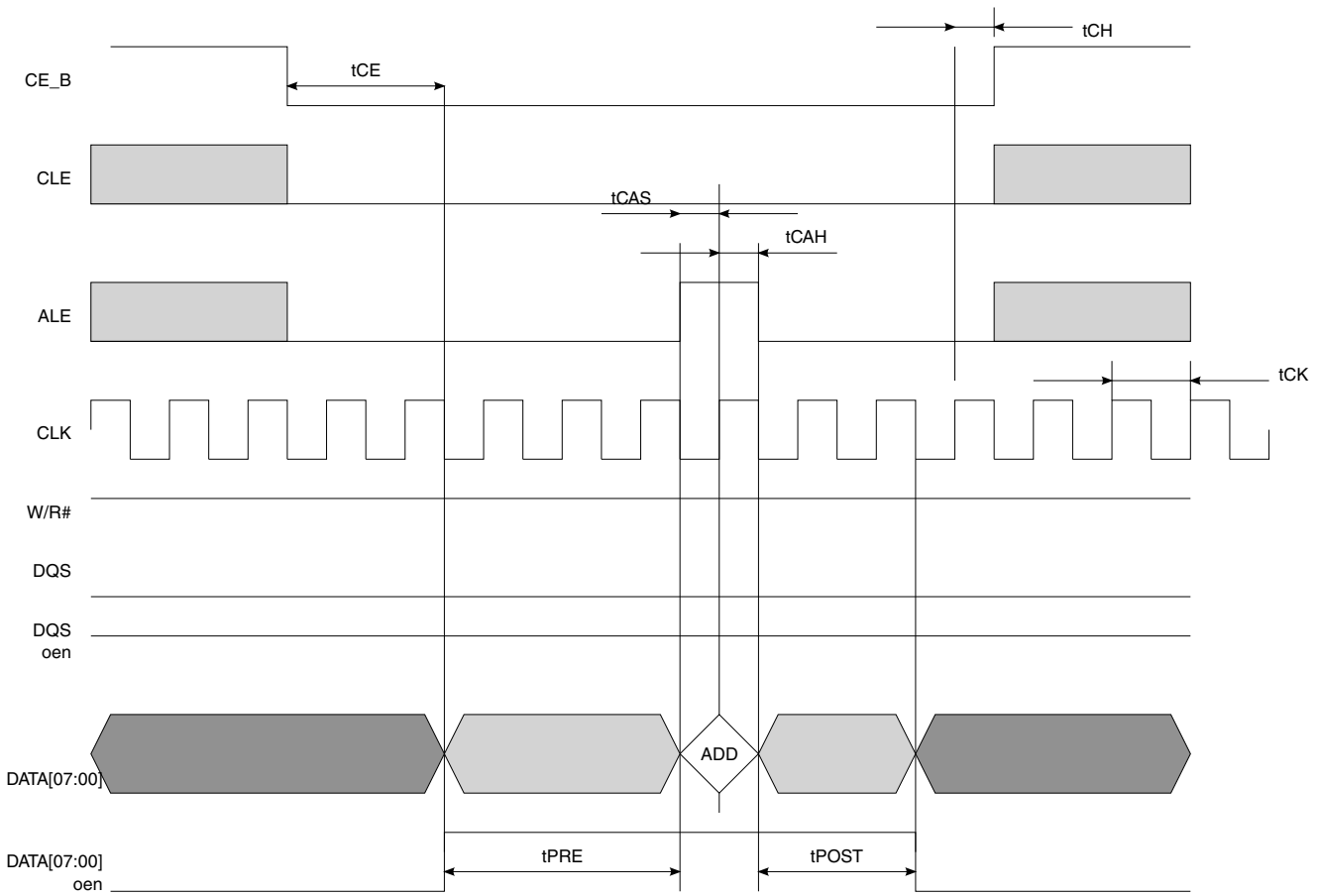
NOTE

In the following figures, CLK shares the same pin as WE_B in Async Mode. And W/R# shares the same pin as RE_B in Async Mode.



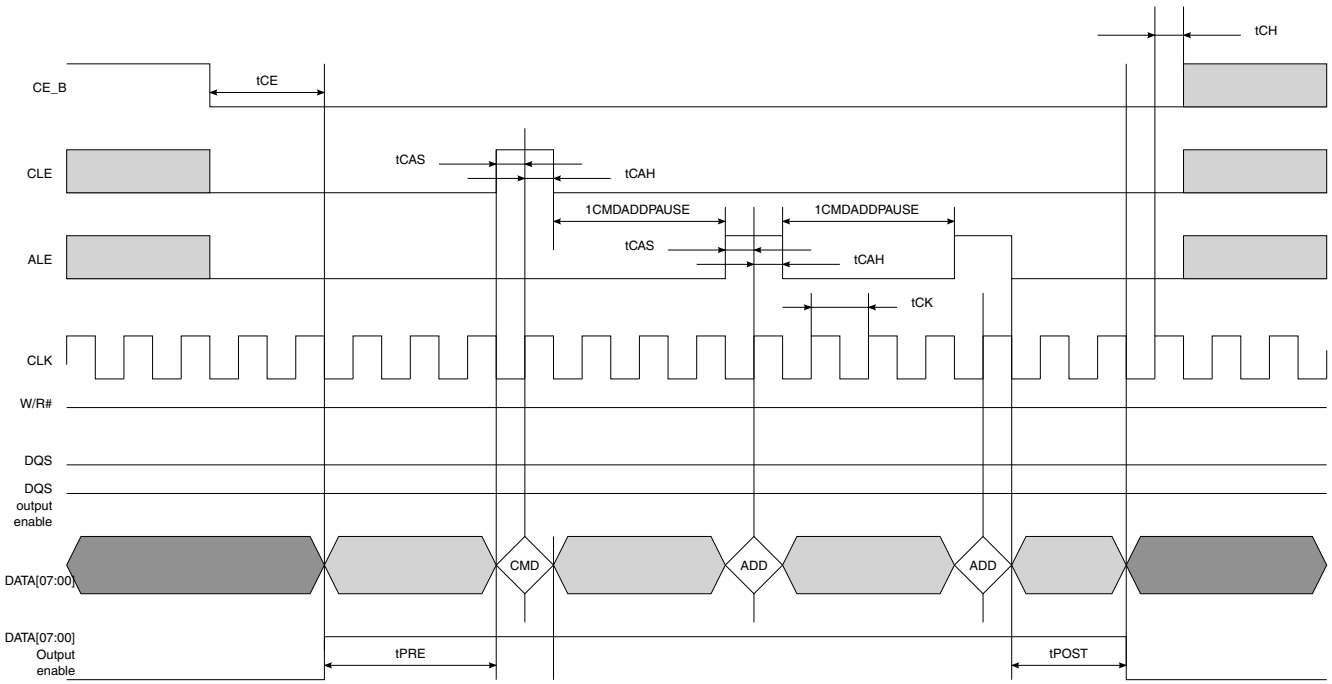
tCK is equal to device clock period
tCAS = 0.5 * tCK
tCAH = 0.5 * tCK
tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK

Figure 27-5. ONFI Source Synchronous Mode Basic Command Write Timing Diagram



tCK is equal to device clock period
tCAS = 0.5 * tCK
tCAH = 0.5 * tCK
tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK

Figure 27-6. ONFI Source Synchronous Mode Basic Address Write Timing Diagram

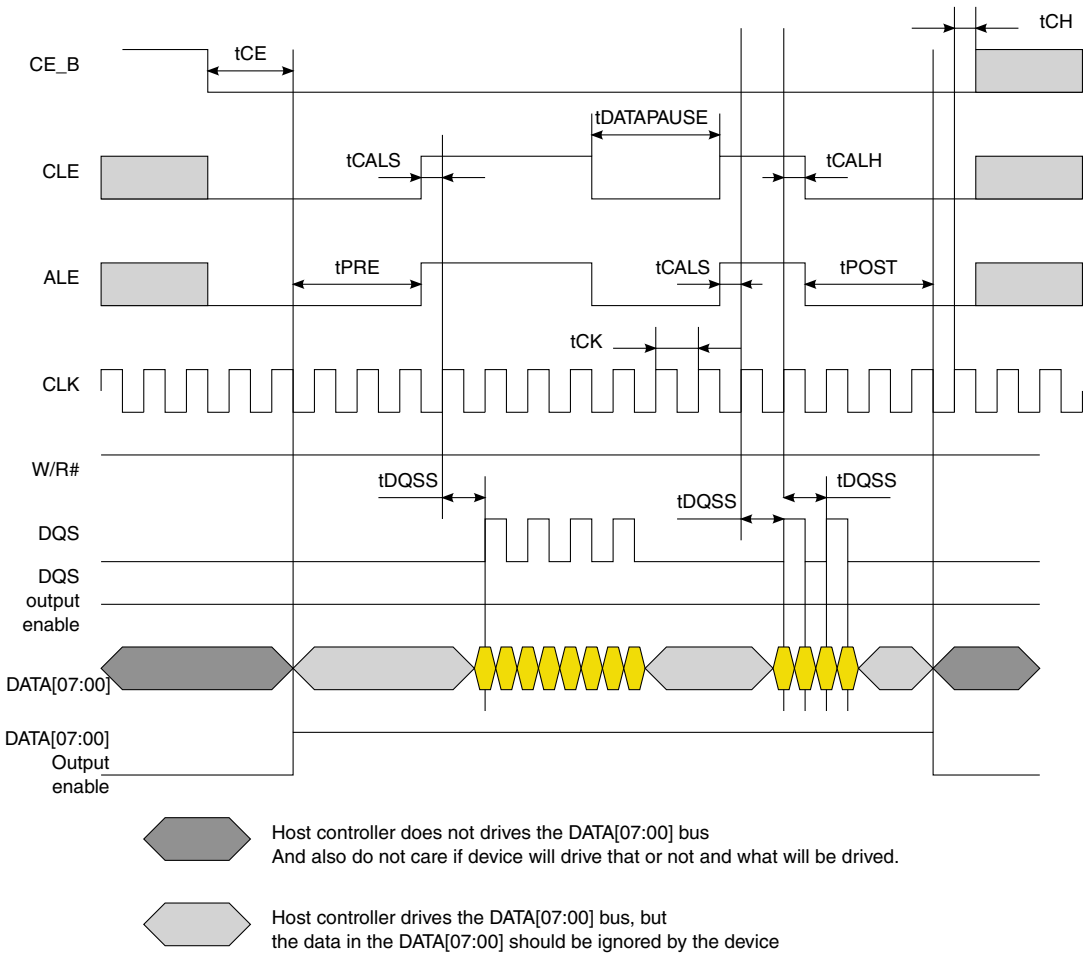


Host controller does not drives the DATA[07:00] bus
And also do not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

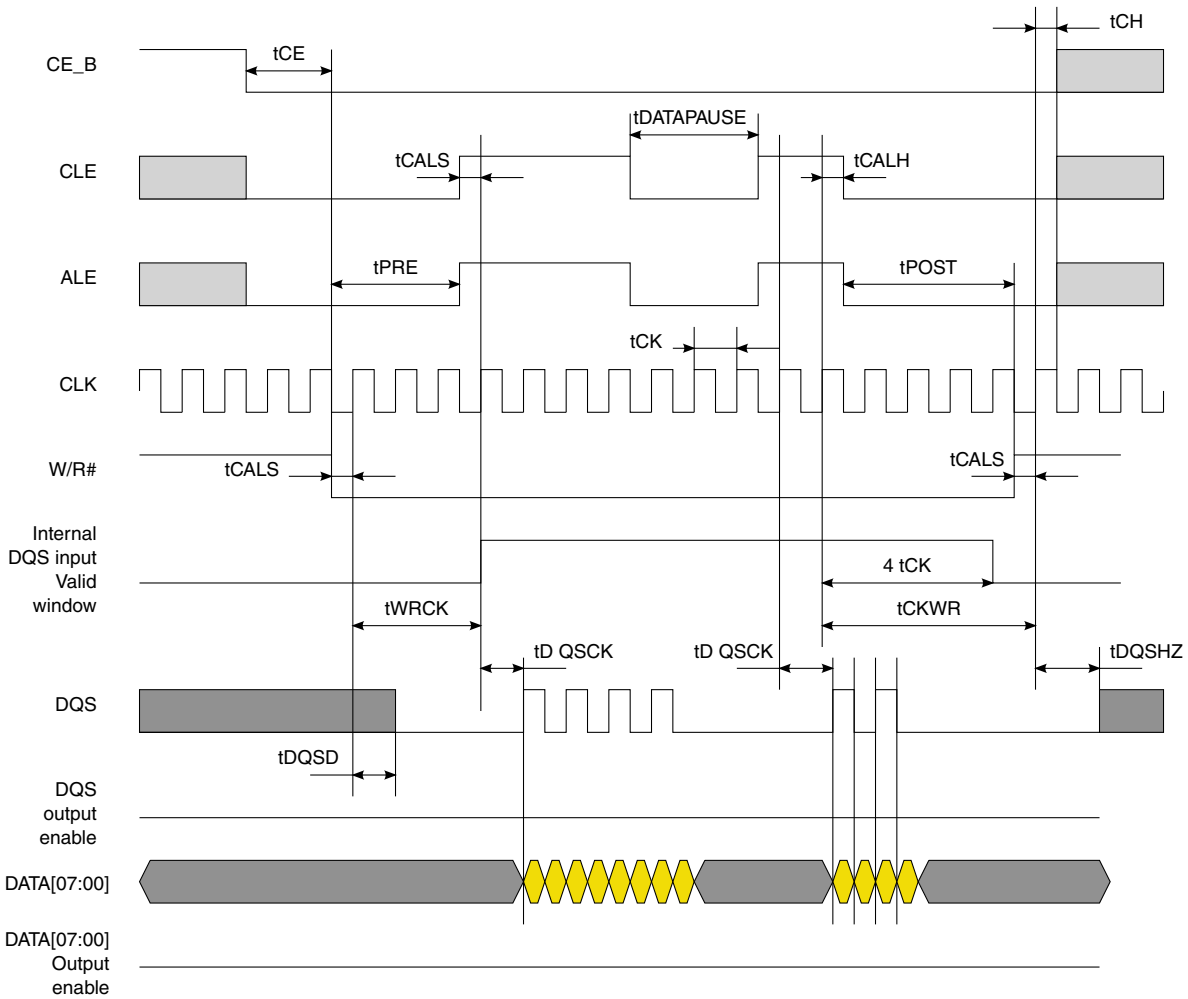
tCK is equal to device clock period
tCAS = 0.5 * tCK
tCAH = 0.5 *tCK
tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tCMDADDPAUSE = HW_GPMI_TIMING2.CMDADD_PAUSE * tCK

Figure 27-7. ONFI Source Synchronous Mode Command + Address Write Timing Diagram



tCK is equal to device clock period
tCE = HW_GPI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tCALS = 0.5 * tCK
tCALH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY *tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tDATAPAUSE = HW_GPMI_TIMING2.DATA_PAUSE * tCK
tDQSS = tCK

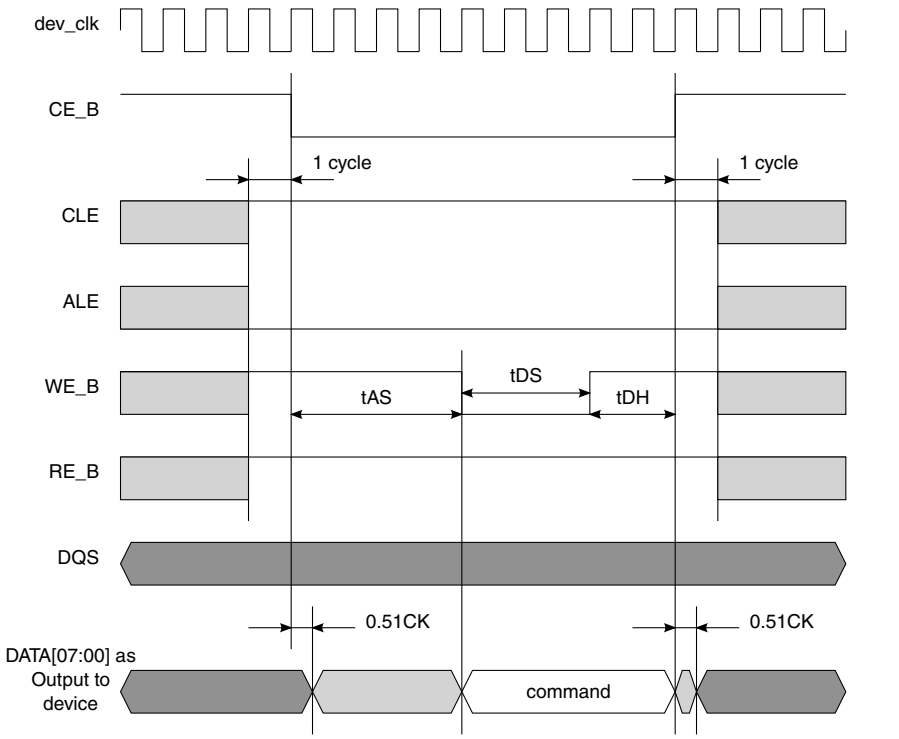
Figure 27-8. ONFI Source Synchronous Mode Data Write Timing Diagram



tCK is equal to device clock period
tCE = HW_GPI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tCALS = 0.5 * tCK
tCALH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY *tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tDATAPAUSE = HW_GPMI_TIMING2.DATA_PAUSE * tCK
tWRCK/tCKWR/tDQSD/tDASCK/tDASHZ are device parameters

Figure 27-9. ONFI Source Synchronous Mode Data Read Timing Diagram

27.4.3.4 NAND Toggle Mode Timing



Host controller does not drive the DATA[07:00] bus
And also does not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup;
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup;
- tDH is configurable br programming HW_GPMI_TIMING Data_Hold;
- tAS/tDS/tDH will extend, if the output data is not ready in device fifo.

Figure 27-10. Samsung Toggle Mode Basic Command Write Timing Diagram

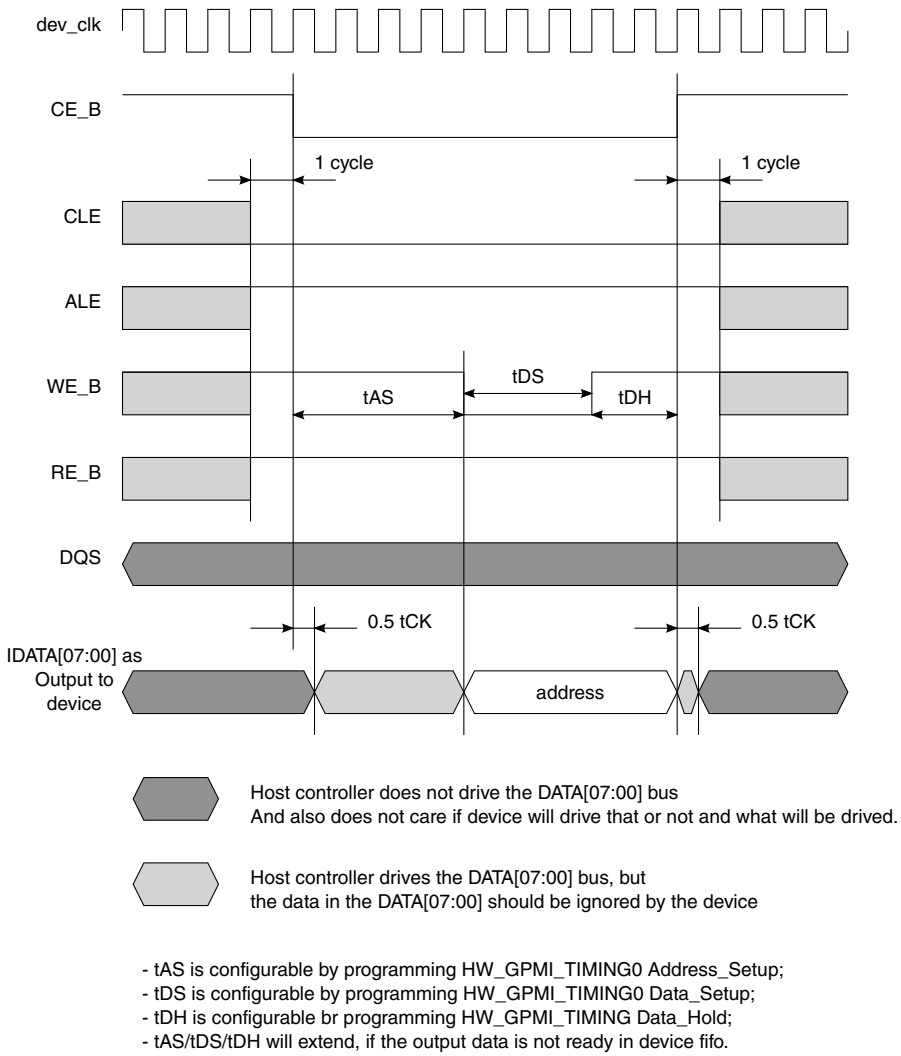


Figure 27-11. Samsung Toggle Mode Basic Address Write Timing Diagram

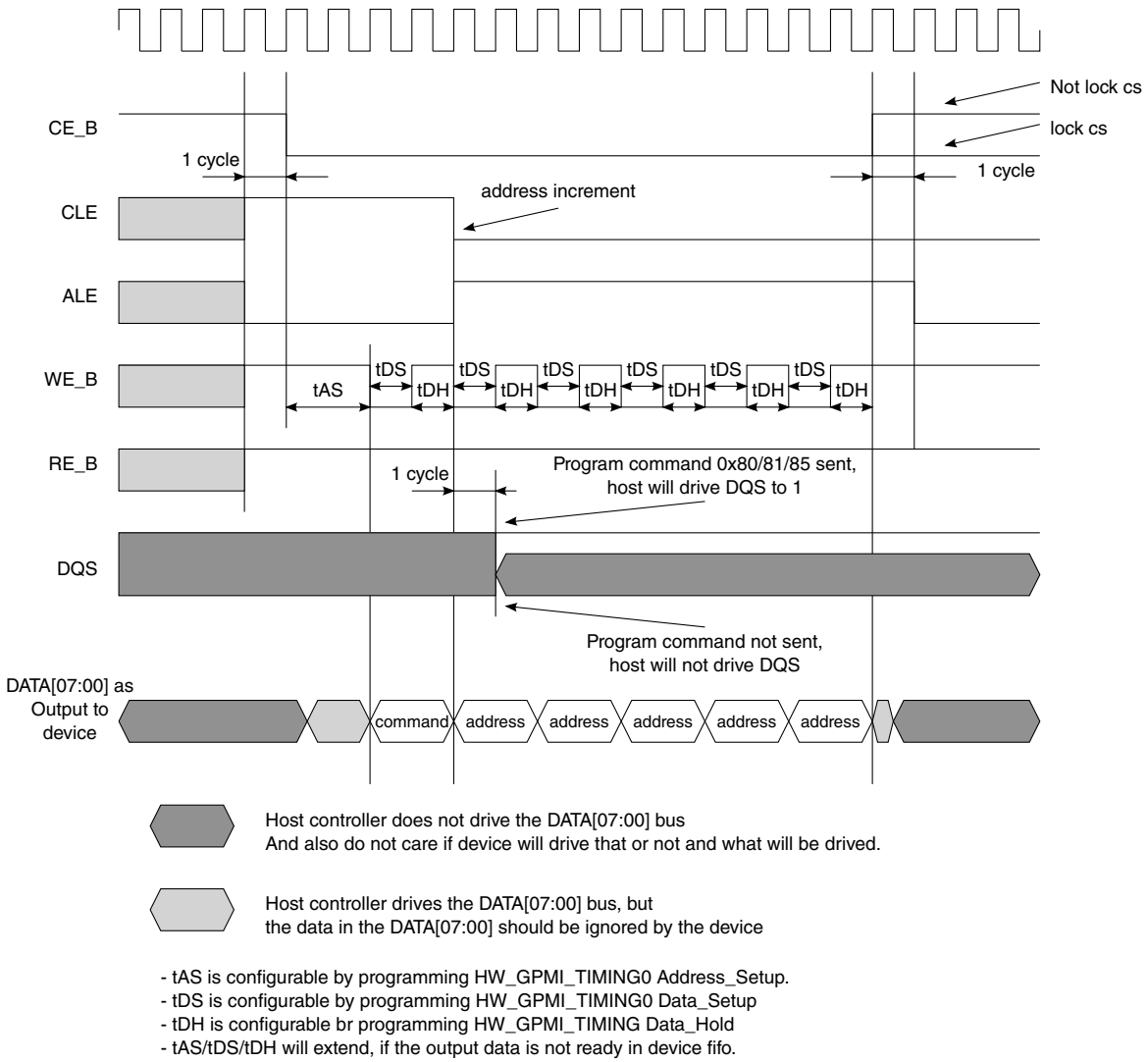


Figure 27-12. Samsung Toggle Mode Basic Command + Address Timing Diagram

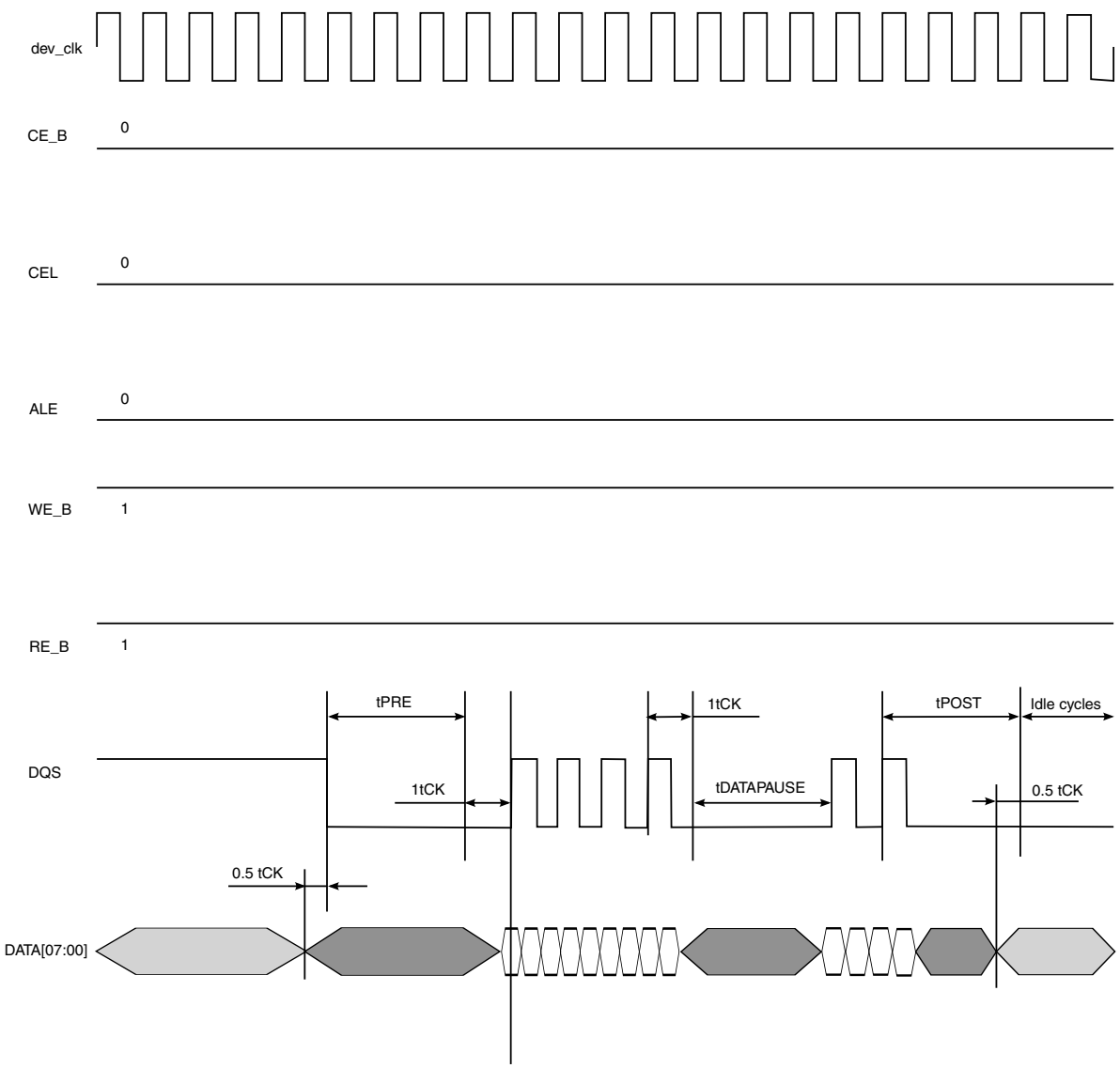
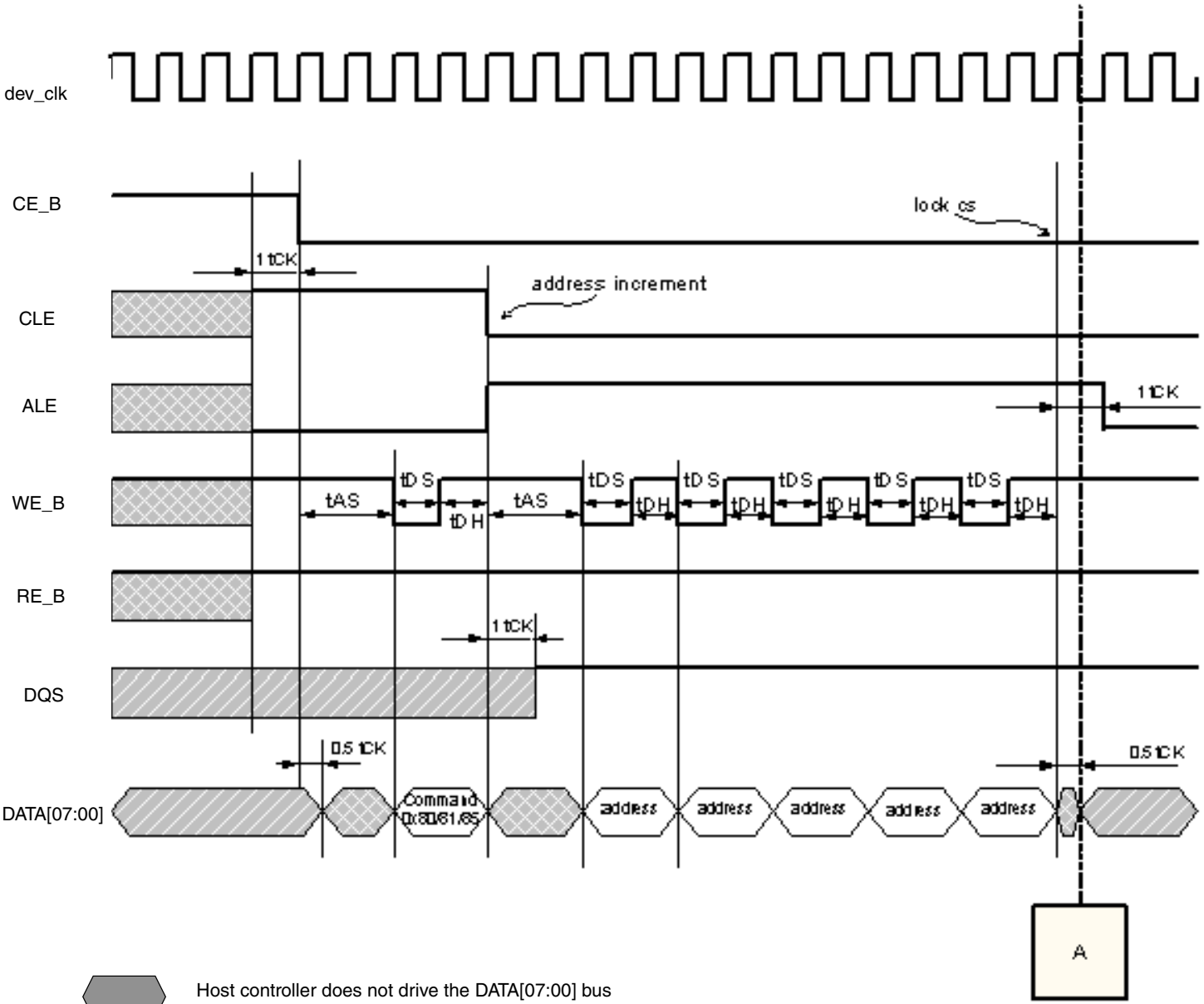


Figure 27-13. Toggle Mode Data Write Timing Diagram

Figure 27-14. Toggle Mode Data Read Timing Diagram

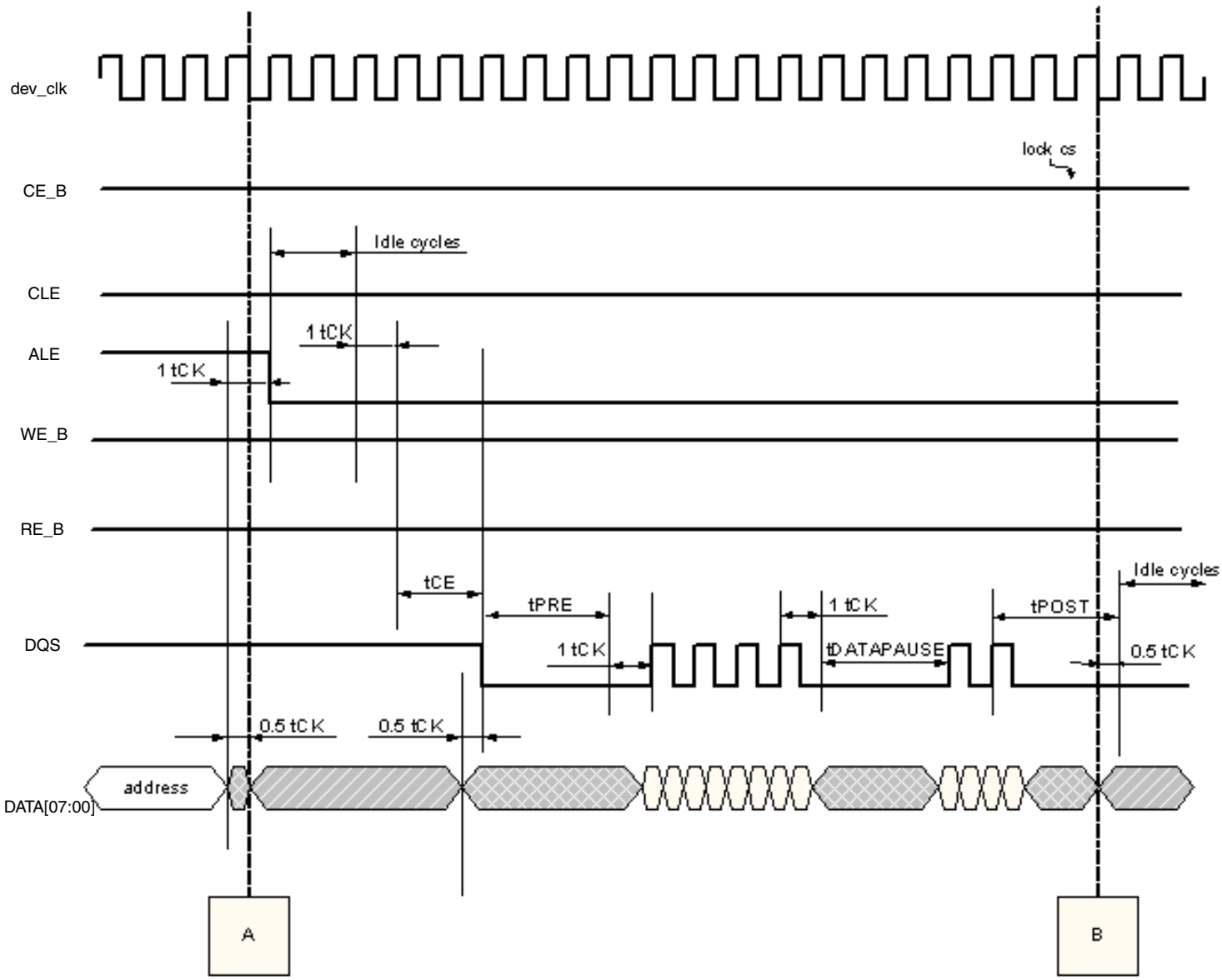


Host controller does not drive the DATA[07:00] bus
And also does not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup:;
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup;
- tDH is configuarble by programming HW_GPMI_TIMING0 Data_Hold;
- tAS/tDS/tDH will extend, if the output data is not ready in device fifo.

Figure 27-15. Toggle Mode Program Timing Diagram (A)



Host controller does not drive the DATA[07:00] bus
And also does not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

- tCK is equal to device clock period
- tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
- tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
- tPOST = HW_GPMI_TIMING2.POSTAMBLE_DELAY * tCK
- tDATAPULSE = HW_GPMI_TIMING2.DATA_PULSE * tCK

Figure 27-16. Toggle Mode Program Timing Diagram (B)

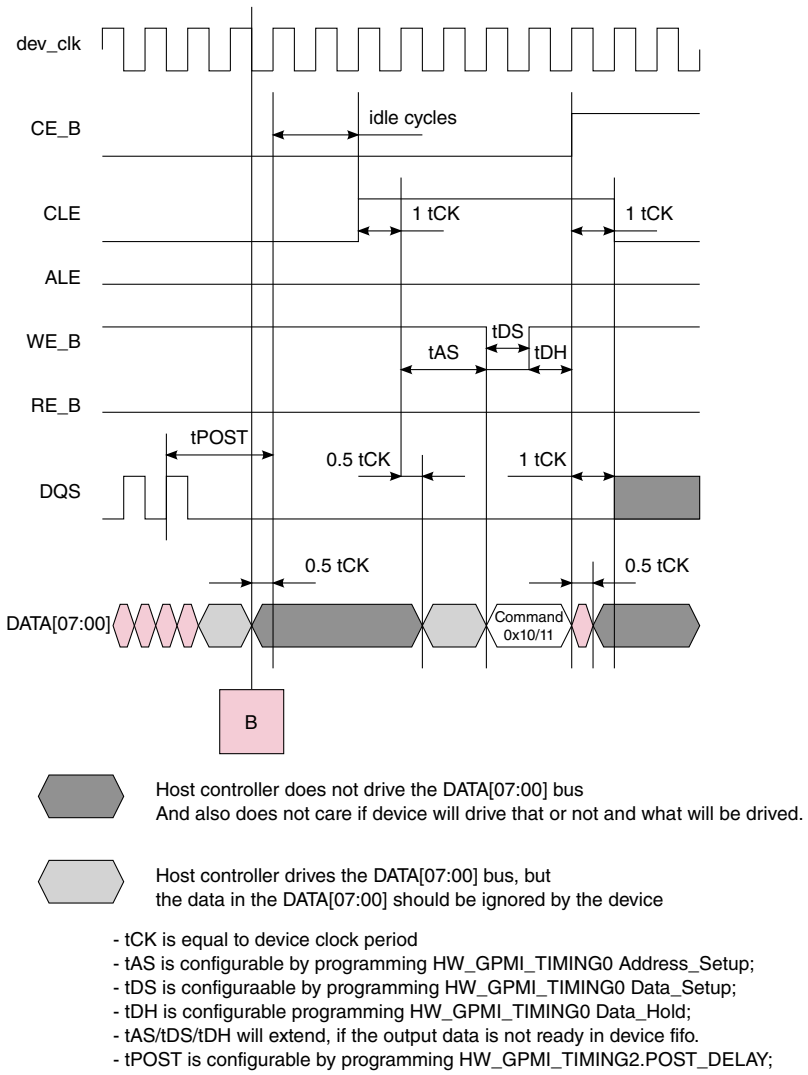


Figure 27-17. Toggle Mode Program Timing Diagram (C)

27.4.4 Hardware BCH Interface

The GPMI provides an interface to the BCH module. This reduces the SOC bus traffic and the software involvement.

.

When BCH ECC is enable, parity information is inserted on-the-fly during writes to 8-bit NAND devices. The BCH will supply payload and parity to the GPMI to write to the NAND. During NAND reads, parity is checked and ECC processing is performed after each read block. In this case the GPMI reads the NAND device and redirects the data and parity to the BCH module for ECC processing.

To program the BCH for NAND writes, remove the soft reset and clock gates from BCH_CTRL[SFTRST] and BCH_CTRL[CLKGATE]. The bulk of BCH programming is actually applied to the GPMI via PIO operations embedded in its DMA command structures. This has a subtle implication when writing to the GPMI ECC registers: access to the these registers must be written in progressive register order. Thus, to write to the GPMI_ECCCOUNT register, write first (in order) to registers GPMI_CTRL0, GPMI_COMPARE, and GPMI_ECCCTRL before writing to GPMI_ECCCOUNT. These additional register writes need to be accounted for in the CMDWORDS field of the respective DMA channel command register.

Note that the GPMI_PAYLOAD and GPMI_AUXILIARY pointers need to be word-aligned for proper ECC operation. If those pointers are non-word-aligned, then the BCH engine will not operate properly and could possibly corrupt system memory in the adjoining memory regions.

27.5 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

27.6 GPMI Memory Map/Register Definition

The following registers provide control for programmable elements of the GPMI module.

NOTE

All ATA or UDMA features are not supported for the chip.

GPMI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_6000	GPMI Control Register 0 Description (GPMI_CTRL0)	32	R/W	C000_0000h	27.6.1/1191
180_6004	GPMI Control Register 0 Description (GPMI_CTRL0_SET)	32	R/W	C000_0000h	27.6.1/1191
180_6008	GPMI Control Register 0 Description (GPMI_CTRL0_CLR)	32	R/W	C000_0000h	27.6.1/1191
180_600C	GPMI Control Register 0 Description (GPMI_CTRL0_TOG)	32	R/W	C000_0000h	27.6.1/1191
180_6010	GPMI Compare Register Description (GPMI_COMPARE)	32	R/W	0000_0000h	27.6.2/1193
180_6020	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL)	32	R/W	0000_0000h	27.6.3/1194

Table continues on the next page...

GPMI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180_6024	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL_SET)	32	R/W	0000_0000h	27.6.3/1194
180_6028	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL_CLR)	32	R/W	0000_0000h	27.6.3/1194
180_602C	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL_TOG)	32	R/W	0000_0000h	27.6.3/1194
180_6030	GPMI Integrated ECC Transfer Count Register Description (GPMI_ECCCOUNT)	32	R/W	0000_0000h	27.6.4/1195
180_6040	GPMI Payload Address Register Description (GPMI_PAYLOAD)	32	R/W	0000_0000h	27.6.5/1195
180_6050	GPMI Auxiliary Address Register Description (GPMI_AUXILIARY)	32	R/W	0000_0000h	27.6.6/1196
180_6060	GPMI Control Register 1 Description (GPMI_CTRL1)	32	R/W	0004_0004h	27.6.7/1197
180_6064	GPMI Control Register 1 Description (GPMI_CTRL1_SET)	32	R/W	0004_0004h	27.6.7/1197
180_6068	GPMI Control Register 1 Description (GPMI_CTRL1_CLR)	32	R/W	0004_0004h	27.6.7/1197
180_606C	GPMI Control Register 1 Description (GPMI_CTRL1_TOG)	32	R/W	0004_0004h	27.6.7/1197
180_6070	GPMI Timing Register 0 Description (GPMI_TIMING0)	32	R/W	0001_0203h	27.6.8/1199
180_6080	GPMI Timing Register 1 Description (GPMI_TIMING1)	32	R/W	0000_0000h	27.6.9/1200
180_6090	GPMI Timing Register 2 Description (GPMI_TIMING2)	32	R/W	0302_3336h	27.6.10/ 1201
180_60A0	GPMI DMA Data Transfer Register Description (GPMI_DATA)	32	R/W	0000_0000h	27.6.11/ 1202
180_60B0	GPMI Status Register Description (GPMI_STAT)	32	R	0000_0005h	27.6.12/ 1202
180_60C0	GPMI Debug Information Register Description (GPMI_DEBUG)	32	R	0000_0000h	27.6.13/ 1205
180_60D0	GPMI Version Register Description (GPMI_VERSION)	32	R	0502_0000h	27.6.14/ 1205
180_60E0	GPMI Debug2 Information Register Description (GPMI_DEBUG2)	32	R/W	0000_F100h	27.6.15/ 1206
180_60F0	GPMI Debug3 Information Register Description (GPMI_DEBUG3)	32	R	0000_0000h	27.6.16/ 1209
180_6100	GPMI Double Rate Read DLL Control Register Description (GPMI_READ_DDR_DLL_CTRL)	32	R/W	0000_0038h	27.6.17/ 1209
180_6110	GPMI Double Rate Write DLL Control Register Description (GPMI_WRITE_DDR_DLL_CTRL)	32	R/W	0000_0038h	27.6.18/ 1211
180_6120	GPMI Double Rate Read DLL Status Register Description (GPMI_READ_DDR_DLL_STS)	32	R	0000_0000h	27.6.19/ 1213
180_6130	GPMI Double Rate Write DLL Status Register Description (GPMI_WRITE_DDR_DLL_STS)	32	R	0000_0000h	27.6.20/ 1214

27.6.1 GPMI Control Register 0 Description (GPMI_CTRL0*n*)

The GPMI control register 0 specifies the GPMI transaction to perform for the current command chain item.

Address: 180_6000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFTRST	CLKGATE	RUN	DEV_IRQ_EN	LOCK_CS	UDMA	COMMAND_MODE		WORD_LENGTH	CS			ADDRESS			ADDRESS_INCREMENT
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

GPMI_CTRL0*n* field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. This will not work if the CLKGATE bit is already set to '1'. CLKGATE must be cleared to '0' before issuing a soft reset. Also the GPMICLK must be running for this to work properly. RUN = 0x0 Allow GPMI to operate normally. RESET = 0x1 Hold GPMI in reset.
30 CLKGATE	Set this bit zero for normal operation. Setting this bit to one (default), gates all of the block level clocks off for miniminizing AC energy consumption. RUN = 0x0 Allow GPMI to operate normally. NO_CLKS = 0x1 Do not clock GPMI gates in order to minimize power consumption.
29 RUN	The GPMI is busy running a command whenever this bit is set to '1'. The GPMI is idle whenever this bit set to zero. This can be set to one by a CPU write. In addition, the DMA sets this bit each time a DMA command has finished its PIO transfer phase. IDLE = 0x0 The GPMI is idle. BUSY = 0x1 The GPMI is busy running a command.

Table continues on the next page...

GPMI_CTRL0*n* field descriptions (continued)

Field	Description
28 DEV_IRQ_EN	When set to '1' and ATA_IRQ pin is asserted, the GPMI_IRQ output will assert.
27 LOCK_CS	For ATA/NAND mode: 0= Deassert chip select (CS) after RUN is complete. 1= Continue to assert chip select (CS) after RUN is complete. For Camera Mode: 0= Dont wait for VSYNC rising edge before capturing data. 1= Wait for VSYNC rising edge before capturing data (Camera mode only). DISABLED = 0x0 Deassert chip select (CS) after RUN is complete. ENABLED = 0x1 Continue to assert chip select (CS) after RUN is complete.
26 UDMA	DISABLED = 0x0 Use ATA-PIO mode on the external bus. ENABLED = 0x1 Use ATA-Ultra DMA mode on the external bus. 0 Use ATA-PIO mode on the external bus. 1 Use ATA-Ultra DMA mode on the external bus.
25–24 COMMAND_ MODE	WRITE = 0x0 Write mode. READ = 0x1 Read mode. READ_AND_COMPARE = 0x2 Read and Compare mode (setting sense flop). WAIT_FOR_READY = 0x3 Wait for Ready mode. For ATA WAIT_FOR_READY command set CS=01. 00 Write mode. 01 Read Mode. 10 Read and Compare Mode (setting sense flop). 11 Wait for Ready.
23 WORD_LENGTH	This bit should only be changed when RUN==0. Reserve = 0x0 Reserved. 8_BIT = 0x1 8-bit Data Bus mode. 0 Reserved. 1 8-bit Data Bus mode.
22–20 CS	Selects which chip select is active for this command. For ATA WAIT_FOR_READY command, this must be set to b01.
19–17 ADDRESS	Specifies the three address lines for ATA mode. In NAND mode, use A0 for CLE and A1 for ALE. NAND_DATA = 0x0 In NAND mode, this address is used to read and write data bytes. NAND_CLE = 0x1 In NAND mode, this address is used to write command bytes. NAND_ALE = 0x2 In NAND mode, this address is used to write address bytes.
16 ADDRESS_ INCREMENT	In ATA mode, the address will increment with each cycle. In NAND mode, the address will increment once, after the first cycle (going from CLE to ALE). DISABLED = 0x0 Address does not increment. ENABLED = 0x1 Increment address. 0 Address does not increment. 1 Increment address.
XFER_COUNT	Number of bytes to transfer for this command. A value of zero will transfer 64K bytes.

27.6.2 GPMI Compare Register Description (GPMI_COMPARE)

The GPMI compare register specifies the expect data and the xor mask for comparing to the status values read from the device. This register is used by the Read and Compare command.

GPMI_COMPARE 0x010

Address: 180_6000h base + 10h offset = 180_6010h

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

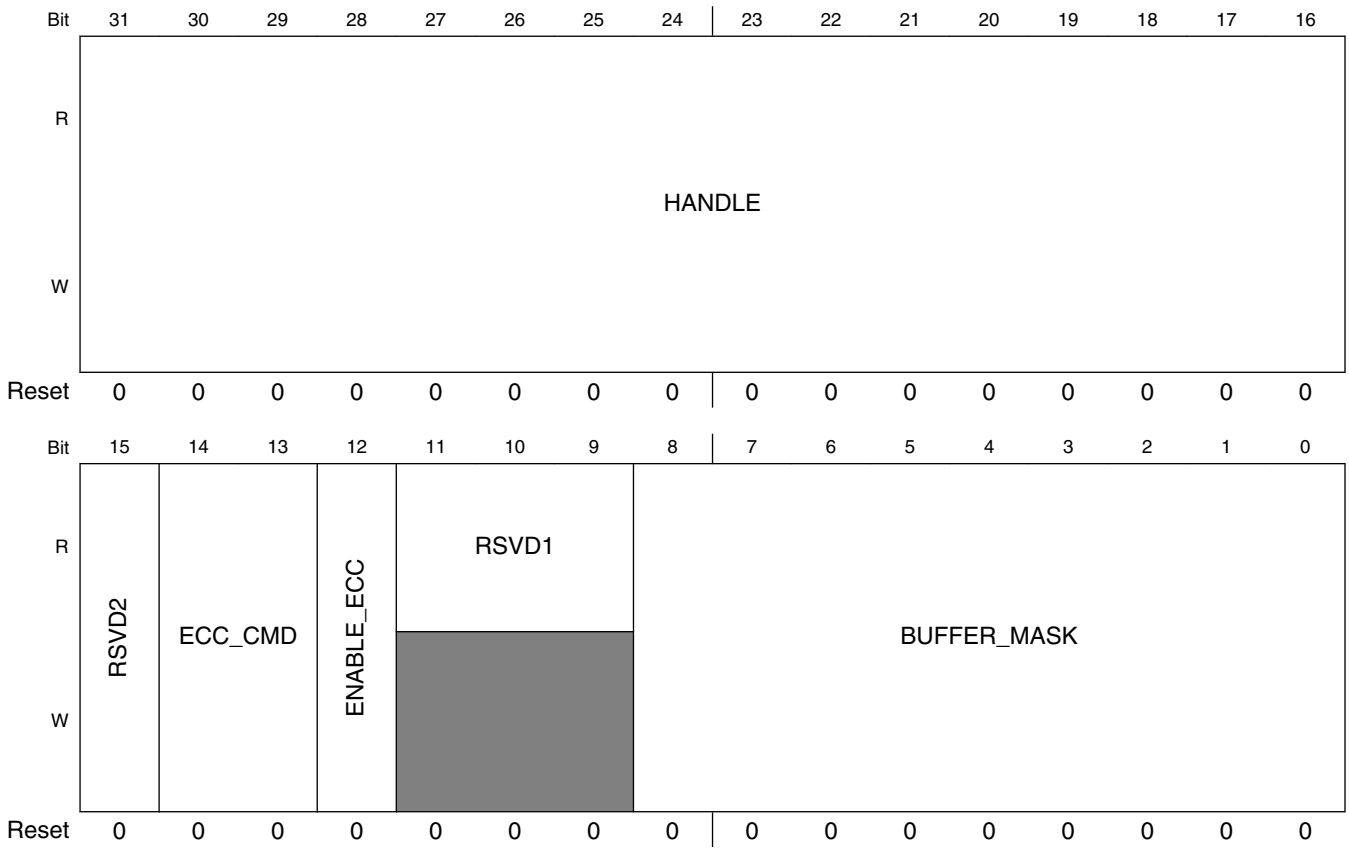
GPMI_COMPARE field descriptions

Field	Description
31–16 MASK	16-bit mask which is applied after the read data is XORed with the REFERENCE bit field.
REFERENCE	16-bit value which is XORed with data read from the NAND device.

27.6.3 GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL*n*)

The GPMI ECC control register handles configuration of the integrated ECC accelerator.

Address: 180_6000h base + 20h offset + (4d × i), where i=0d to 3d



GPMI_ECCCTRL*n* field descriptions

Field	Description
31–16 HANDLE	This is a register available to software to attach an identifier to a transaction in progress. This handle will be available from the ECC register space when the completion interrupt occurs.
15 RSVD2	Always write zeroes to this bit field.
14–13 ECC_CMD	ECC Command information. DECODE = 0x0 Decode. ENCODE = 0x1 Encode. RESERVE2 = 0x2 Reserved. RESERVE3 = 0x3 Reserved.

Table continues on the next page...

GPMI_ECCCTRL*n* field descriptions (continued)

Field	Description
12 ENABLE_ECC	Enable ECC processing of GPMI transfers. ENABLE = 0x1 Use integrated ECC for read and write transfers. DISABLE = 0x0 Integrated ECC remains in idle.
11–9 RSVD1	Always write zeroes to this bit field.
BUFFER_MASK	ECC buffer information. The BCH error correction only allows two configurations of the buffer mask - software may either read just the first block on the flash page or the entire flash page. Write operations must be for the entire flash page. Invalid buffer mask values will cause the DMA descriptor command to be terminated. BCH_AUXONLY = 0x100 Set to request transfer from only the auxiliary buffer (block 0 on flash). BCH_PAGE = 0x1FF Set to request transfer to/from the entire page.

27.6.4 GPMI Integrated ECC Transfer Count Register Description (GPMI_ECCCOUNT)

The GPMI ECC Transfer Count Register contains the count of bytes that flow through the ECC subsystem.

GPMI_ECCCOUNT 0x030

Address: 180_6000h base + 30h offset = 180_6030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD2																COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_ECCCOUNT field descriptions

Field	Description
31–16 RSVD2	Always write zeroes to this bit field.
COUNT	Number of bytes to pass through ECC. This is the GPMI transfer count plus the syndrome count that will be inserted into the stream by the ECC. In DMA2ECC_MODE this count must match the GPMI_CTRL0_XFER_COUNT. A value of zero will transfer 64K words.

27.6.5 GPMI Payload Address Register Description (GPMI_PAYLOAD)

The GPMI payload address register specifies the location of the data buffers in system memory. This value must be word aligned.

GPMI_PAYLOAD 0x040

Address: 180_6000h base + 40h offset = 180_6040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDRESS														RSVD0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_PAYLOAD field descriptions

Field	Description
31–2 ADDRESS	Pointer to an array of one or more 512 byte payload buffers.
RSVD0	Always write zeroes to this bit field.

27.6.6 GPMI Auxiliary Address Register Description (GPMI_AUXILIARY)

The GPMI auxiliary address register specifies the location of the auxiliary buffers in system memory. This value must be word aligned.

GPMI_AUXILIARY 0x050

Address: 180_6000h base + 50h offset = 180_6050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDRESS														RSVD0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_AUXILIARY field descriptions

Field	Description
31–2 ADDRESS	Pointer to ECC control structure and meta-data storage.
RSVD0	Always write zeroes to this bit field.

27.6.7 GPMI Control Register 1 Description (GPMI_CTRL1*n*)

The GPMI control register 1 specifies additional control fields that are not used on a per-transaction basis.

Address: 180_6000h base + 60h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEV_CLK_STOP	SSYNC_CLK_STOP	WRITE_CLK_STOP	TOGGLE_MODE	GPMI_CLK_DIV2_EN	UPDATE_CS	SSYNCMODE	DECOUPLE_CS	WRN_DLY_SEL		TEST_TRIGGER	TIMEOUT_IRQ_EN	GANGED_RDYBUSY	BCH_MODE	DLL_ENABLE	HALF_PERIOD
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RDN_DELAY				DMA2ECC_MODE	DEV_IRQ	TIMEOUT_IRQ	BURST_EN	ABORT_WAIT_REQUEST	ABORT_WAIT_FOR_READY_CHANNEL			DEV_RESET	ATA_IRQRDY_POLARITY	CAMERA_MODE	GPMI_MODE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

GPMI_CTRL1*n* field descriptions

Field	Description
31 DEV_CLK_STOP	set this bit to 1 will stop gpmi io working clk.
30 SSYNC_CLK_STOP	set this bit to 1 will stop the source synchronous mode clk.
29 WRITE_CLK_STOP	In onfi source synchronous mode, host may save power during the data write cycles by holding the CLK signal high (i.e. stopping the CLK). The host may only stop the CLK during data write, by setting this bit to 1, if the device supports this feature as indicated in the parameter page.
28 TOGGLE_MODE	enable samsung toggle mode.
27 GPMI_CLK_DIV2_EN	This bit should be reset to 0 in asynchronous mode. The frequence ratio of (device clock : ccm gpmi clock) will be (1 : 1). This bit should be set to 1, in source synchronous mode or toggle mode. The frequence raito of (device clock : ccm gpmi clock) will be (1 : 2). enable the gpmi clk divider. 0x0 internal factor-2 clock divider is disabled 0x1 internal factor-2 clock divider is enabled.

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
26 UPDATE_CS	force the CS value is be updated to external chip select pin, even GPMI is idle.
25 SSYNCMODE	source synchronouse mode 1 or asynchrous mode 0. ASYNC = 0x0 Asynchronous mode. SSYNC = 0x1 Source Synchronous mode.
24 DECOUPLE_CS	Decouple Chip Select from DMA Channel. Setting this bit to 1 will allow a DMA channel to specify any value in the CTRL0_CS register field. Software can use one DMA channel to access all 8 Nand devices.
23–22 WRN_DLY_SEL	Since the GPMI write strobe (WRN) is a fast clock pin, the delay on this signal can be programmed to match the load on this pin. 0 = ~2ns; 1 = ~4ns; 2 = ~6ns; 3 = no delay.
21 TEST_TRIGGER	Test Trigger Enable 0 Disable 1 Enable
20 TIMEOUT_IRQ_EN	Setting this bit to '1' will enable timeout IRQ for transfers in ATA mode only, and for WAIT_FOR_READY commands in both ATA and Nand mode. The Device_Busy_Timeout value is used for this timeout.
19 GANGED_RDYBUSY	Set this bit to 1 will force all Nand RDY_BUSY inputs to be sourced from (tied to) RDY_BUSY0. This will free up all, except one, RDY_BUSY input pins.
18 BCH_MODE	This bit selects which error correction unit will access GPMI. This bit must always be set to '1', since only the BCH unit is available in this design.
17 DLL_ENABLE	Set this bit to 1 to enable the GPMI DLL. This is required for fast NAND reads (above 30 MHz read strobe). After setting this bit, wait 64 GPMI clock cycles for the DLL to lock before performing a NAND read.
16 HALF_PERIOD	Set this bit to 1 if the GPMI clock period is greater than 16ns for proper DLL operation. DLL_ENABLE must be zero while changing this field.
15–12 RDN_DELAY	This variable is a factor in the calculated delay to apply to the internal read strobe for correct read data sampling. The applied delay (AD) is between 0 and 1.875 times the reference period (RP). RP is one half of the GPMI clock period if HALF_PERIOD=1 otherwise it is the full GPMI clock period. The equation is: AD = RDN_DELAY x 0.125 x RP. This value must not exceed 16ns. This variable is used to achieve faster NAND access. For example if the Read Strobe is asserted from time 0 to 13ns but the read access time is 20ns, then choose AD=12ns will cause the data to be sampled at time 25ns (13+12) giving a 5ns data setup time. If RP=13ns then RDN_DELAY = 12/(0.125 x 13ns) = 7.38 (0111b). DLL_ENABLE must be zero while changing this field.
11 DMA2ECC_MODE	This is mainly for testing HWECC without involving the Nand device. Setting this bit will cause DMA write data to redirected to HWECC module (instead of Nand Device) for encoding or decoding.
10 DEV_IRQ	This bit is set when an Interrupt is received from the ATA device. Write 0 to clear.
9 TIMEOUT_IRQ	This bit is set when a timeout occurs using the Device_Busy_Timeout value. Write 0 to clear.

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
8 BURST_EN	When set to 1 each DMA request will generate a 4-transfer burst on the APB bus.
7 ABORT_WAIT_REQUEST	Request to abort "wait for ready" command on channel indicated by ABORT_WAIT_FOR_READY_CHANNEL. Hardware will clear this bit when abort is done.
6–4 ABORT_WAIT_FOR_READY_CHANNEL	Abort a wait for ready command on selected channel. Set the ABORT_WAIT_REQUEST to kick of operation.
3 DEV_RESET	ENABLED = 0x0 NANDF_WP_B(WPN) pin is held low (asserted). DISABLED = 0x1 NANDF_WP_B(WPN) pin is held high (de-asserted). 0 NANDF_WP_B pin is held low (asserted). 1 NANDF_WP_B pin is held high (de-asserted).
2 ATA_IRQRDY_POLARITY	For ATA MODE: Note NAND_RDY_BUSY[3:2] are not affected by this bit. ACTIVELOW = 0x0 ATA IORDY and IRQ are active low, or NAND_RDY_BUSY[1:0] are active low ready. ACTIVEHIGH = 0x1 ATA IORDY and IRQ are active high, or NAND_RDY_BUSY[1:0] are active high ready. 0 External ATA IORDY and IRQ are active low. 1 External ATA IORDY and IRQ are active high. For NAND MODE: 0 External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when low and busy when high. 1 External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when high and busy when low.
1 CAMERA_MODE	When set to 1 and ATA UDMA is enabled the UDMA interface becomes a camera interface.
0 GPMI_MODE	ATA mode is only supported on channel zero. If ATA mode is selected, then only channel three is available for NAND use. NAND = 0x0 NAND mode. ATA = 0x1 ATA mode. 0 NAND mode. 1 ATA mode.

27.6.8 GPMI Timing Register 0 Description (GPMI_TIMING0)

The GPMI timing register 0 specifies the timing parameters that are used by the cycle state machine to guarantee the various setup, hold and cycle times for the external media type.

GPMI_TIMING0 0x070

Address: 180_6000h base + 70h offset = 180_6070h

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

GPMI_TIMING0 field descriptions

Field	Description
31–24 RSVD1	Always write zeroes to this bit field.
23–16 ADDRESS_ SETUP	Number of GPMICLK cycles that the CE/ADDR signals are active before a strobe is asserted. A value of zero is interpreted as 0. For ATA PIO modes this is known in the ATA7 specification as "Address valid to DIOR-/DIOW- setup"
15–8 DATA_HOLD	Data bus hold time in GPMICLK cycles. Also the time that the data strobe is de-asserted in a cycle. A value of zero is interpreted as 256. For ATA PIO modes this is known in the ATA7 specification as "DIOR-/DIOW- recovery time"
DATA_SETUP	Data bus setup time in GPMICLK cycles. Also the time that the data strobe is asserted in a cycle. This value must be greater than 2 for ATA devices that use IORDY to extend transfer cycles. A value of zero is interpreted as 256. For ATA PIO modes this is known in the ATA7 specification as ""DIOR-/DIOW-"

27.6.9 GPMI Timing Register 1 Description (GPMI_TIMING1)

The GPMI timing register 1 specifies the timeouts used when monitoring the NAND READY pin or the ATA IRQ and IOWAIT signals.

GPMI_TIMING1 0x080

Address: 180_6000h base + 80h offset = 180_6080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEVICE_BUSY_TIMEOUT																RSVD1															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_TIMING1 field descriptions

Field	Description
31–16 DEVICE_BUSY_ TIMEOUT	Timeout waiting for NAND Ready/Busy or ATA IRQ. Used in WAIT_FOR_READY mode. This value is the number of GPMI_CLK cycles multiplied by 4096.
RSVD1	Always write zeroes to this bit field.

27.6.10 GPMI Timing Register 2 Description (GPMI_TIMING2)

The GPMI timing register 2 specifies the double data rate timing parameters that are used by the cycle state machine to guarantee the various cs delay, pre-amble delay, post-amble delay, command/address delay, data delay, TCR, TRPSTH, and read latency cycle times for the external media type.

GPMI_TIMING2 0x090

Address: 180_6000h base + 90h offset = 180_6090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TRPSTH			TCR		READ_ LATENCY			RSVD0			CE_DELAY					PREAMBLE_ DELAY				POSTAMBLE _DELAY				CMDADD_ PAUSE				DATA_ PAUSE			
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	0

GPMI_TIMING2 field descriptions

Field	Description
31–29 TRPSTH	Only for Toggle NAND timing control delay TRPSTH GPMICLK cycles for CEn_B high to RE_B high, A value of zero is interpreted as 8
28–27 TCR	Only for Toggle NAND timing control delay (TCR+1) GPMICLK cycles for CEn_B low to RE_B low, 0 is less than or equal to TCR, which is less than the PREAMBLE_DELAY
26–24 READ_ LATENCY	This field is for double data rate read latency configuration. others READ LATENCY is 3 000 READ LATENCY is 0 001 READ LATENCY is 1 010 READ LATENCY is 2 011 READ LATENCY is 3 100 READ LATENCY is 4 101 READ LATENCY is 5
23–21 RSVD0	Always write zeroes to this bit field.
20–16 CE_DELAY	GPMI dealy from CEn assert to W/Rn changing edge. value of zero is interpreted as 32.
15–12 PREAMBLE_ DELAY	GPMI pre-amble delay in GPMICLK cycles. A value of zero is interpreted as 16.
11–8 POSTAMBLE_ DELAY	GPMI post-amble delay in GPMICLK cycles. A value of zero is interpreted as 16.
7–4 CMDADD_ PAUSE	GPMI delay time from command or addres pause to command or address resume in GPMICLK cycles. A value of zero is interpreted as 16.
DATA_PAUSE	GPMI delay time from data pause to data resume in GPMICLK cycles. A value of zero is interpreted as 16.

27.6.11 GPMI DMA Data Transfer Register Description (GPMI_DATA)

The GPMI DMA data transfer register is used by the DMA to read or write data to or from the ATA/NAND control state machine.

GPMI_DATA 0x0A0

Address: 180_6000h base + A0h offset = 180_60A0h

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

GPMI_DATA field descriptions

Field	Description
DATA	In 8-bit mode, one, two, three or four bytes can can be accessed to send the same number of bus cycles.

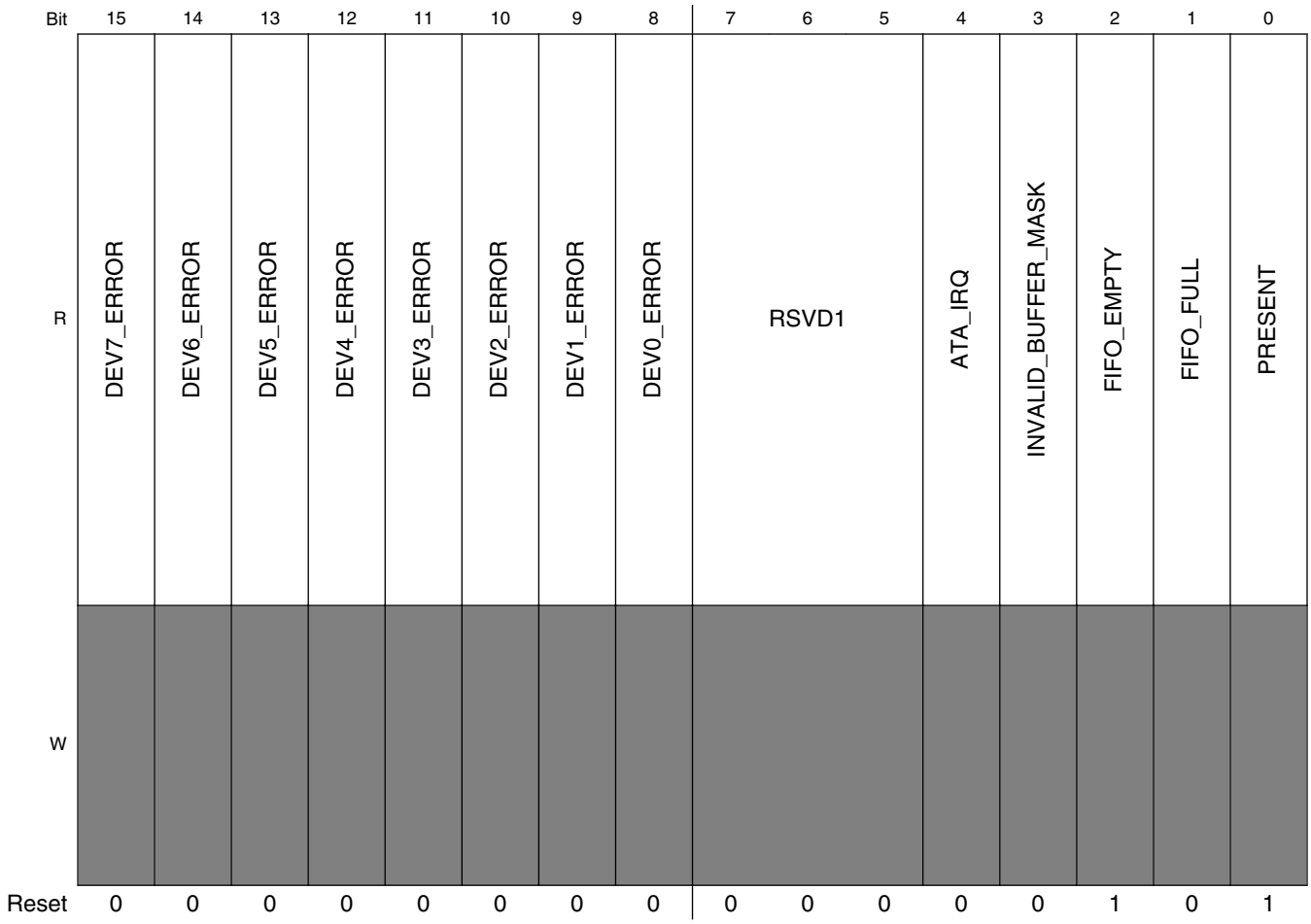
27.6.12 GPMI Status Register Description (GPMI_STAT)

The GPMI control and status register provides a read back path for various operational states of the GPMI controller.

GPMI_STAT 0x0B0

Address: 180_6000h base + B0h offset = 180_60B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	READY_BUSY								RDY_TIMEOUT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



GPMI_STAT field descriptions

Field	Description
31–24 READY_BUSY	Read-only view of NAND Ready_Busy Input pins.
23–16 RDY_TIMEOUT	State of the RDY/BUSY Timeout Flags. When any bit is set to '1' in this field, it indicates that a time out has occurred while waiting for the ready state of the requested NAND device. Multiple bits may be set simultaneously. When GPMI_CTRL1_DECOUPLE_CS = 0, RDY_TIMEOUT[n] is associated with the NAND device on chip_select[n]. When GPMI_CTRL1_DECOUPLE_CS = 1, these flags become associated to a DMA channel instead of a NAND device. For example if DMA channel 6 sends a WAIT_FOR_READY command for NAND Device 2, and a timeout occured on READY_BUSY2, then READY_TIMEOUT[6] will be set instead of READY_TIMEOUT[2].
15 DEV7_ERROR	DMA channel 7 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 7. 1 An Error has occurred on ATA/NAND Device accessed by

Table continues on the next page...

GPMI_STAT field descriptions (continued)

Field	Description
14 DEV6_ERROR	DMA channel 6 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 6. 1 An Error has occurred on ATA/NAND Device accessed by
13 DEV5_ERROR	DMA channel 5 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 5. 1 An Error has occurred on ATA/NAND Device accessed by
12 DEV4_ERROR	DMA channel 4 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 4. 1 An Error has occurred on ATA/NAND Device accessed by
11 DEV3_ERROR	DMA channel 3 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 3. 1 An Error has occurred on ATA/NAND Device accessed by
10 DEV2_ERROR	DMA channel 2 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 2. 1 An Error has occurred on ATA/NAND Device accessed by
9 DEV1_ERROR	DMA channel 1 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 1. 1 An Error has occurred on ATA/NAND Device accessed by
8 DEV0_ERROR	DMA channel 0 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 0. 1 An Error has occurred on ATA/NAND Device accessed by
7–5 RSVD1	Always write zeroes to this bit field.
4 ATA_IRQ	Status of the ATA_IRQ input pin.
3 INVALID_ BUFFER_MASK	Buffer Mask Validity bit. 0 ECC Buffer Mask is not invalid. 1 ECC Buffer Mask is invalid.
2 FIFO_EMPTY	NOT_EMPTY = 0x0 FIFO is not empty. EMPTY = 0x1 FIFO is empty. 0 FIFO is not empty. 1 FIFO is empty.
1 FIFO_FULL	NOT_FULL = 0x0 FIFO is not full. FULL = 0x1 FIFO is full. 0 FIFO is not full. 1 FIFO is full.
0 PRESENT	UNAVAILABLE = 0x0 GPMI is not present in this product. AVAILABLE = 0x1 GPMI is present in this product.

Table continues on the next page...

GPMI_STAT field descriptions (continued)

Field	Description
	0 GPMI is not present in this product. 1 GPMI is present is in this product.

27.6.13 GPMI Debug Information Register Description (GPMI_DEBUG)

The GPMI debug information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

GPMI_DEBUG 0x0C0

Address: 180_6000h base + C0h offset = 180_60C0h

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

GPMI_DEBUG field descriptions

Field	Description
31–24 WAIT_FOR_ READY_END	Read Only view of the Wait_For_Ready End toggle signals to DMA. One per channel
23–16 DMA_SENSE	Read-only view of sense state of the 8 DMA channels. A value of "1" in any bit position indicates that a read and compare command failed or a timeout occured for the corresponding channel.
15–8 DMAREQ	Read-only view of DMA request line for 8 DMA channels. A toggle on any bit position indicates a DMA request for the corresponding channel.
CMD_END	Read Only view of the Command End toggle signals to DMA. One per channel

27.6.14 GPMI Version Register Description (GPMI_VERSION)

This register reflects the version number for the GPMI.

GPMI_VERSION 0x0D0

Address: 180_6000h base + D0h offset = 180_60D0h

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

GPMI_VERSION field descriptions

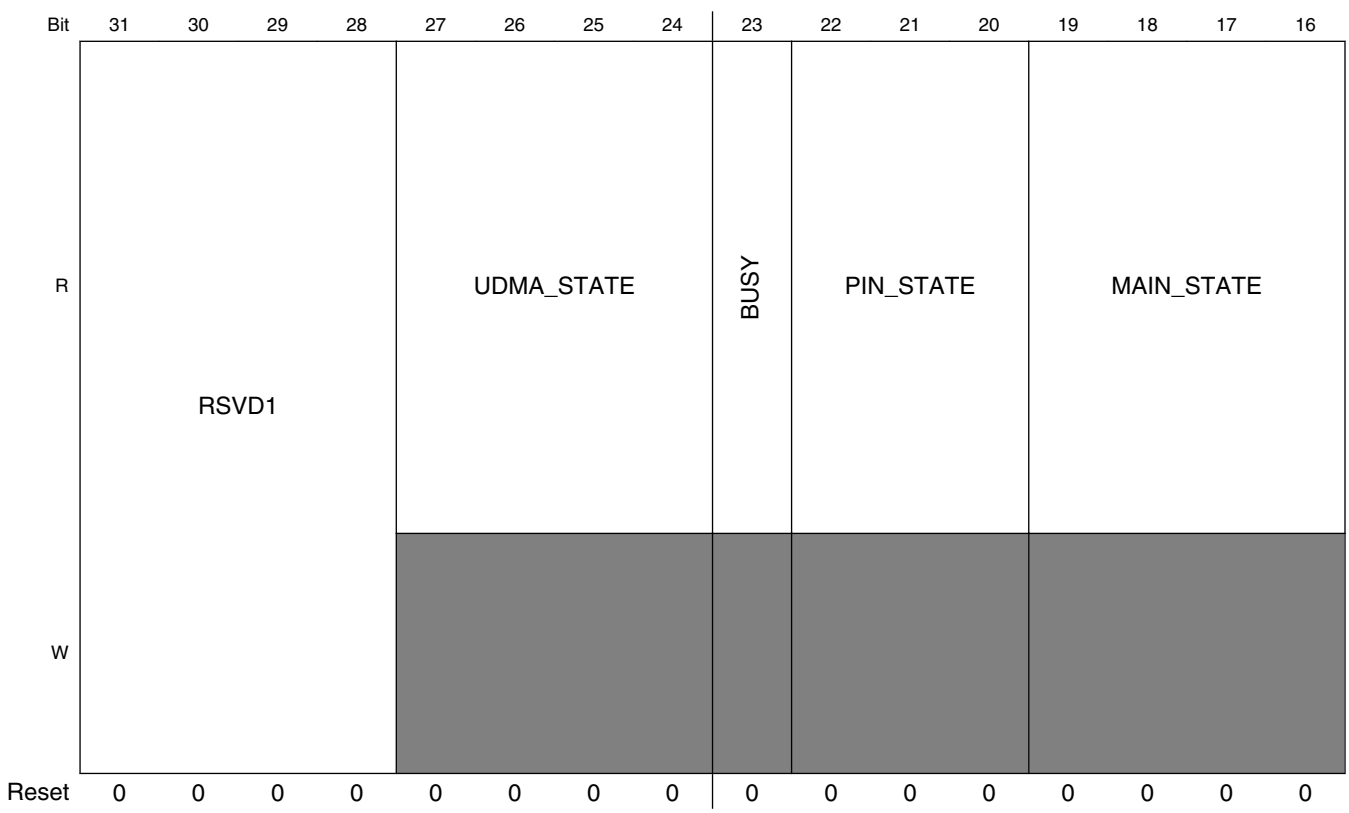
Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

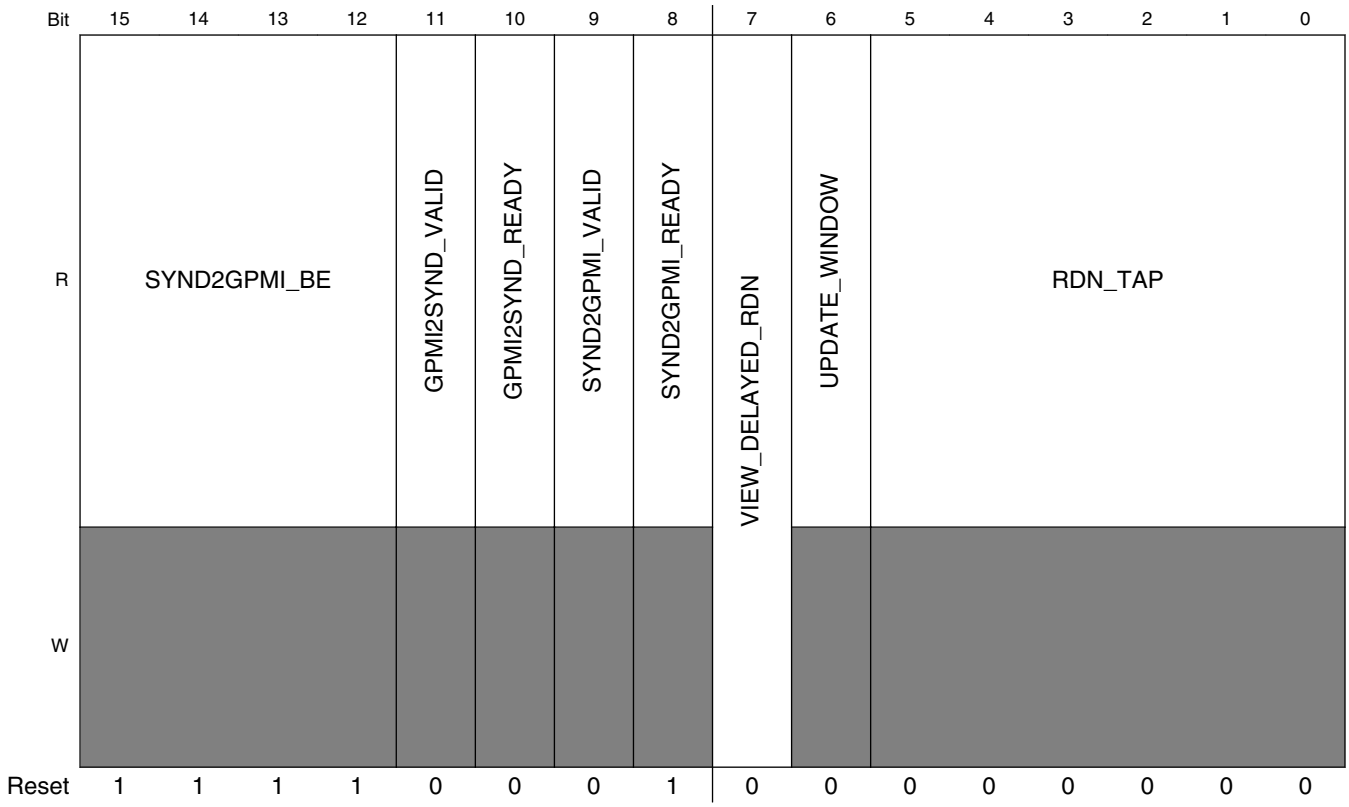
27.6.15 GPMI Debug2 Information Register Description (GPMI_DEBUG2)

The GPMI Debug2 information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

GPMI_DEBUG2 0x0E0

Address: 180_6000h base + E0h offset = 180_60E0h





GPMI_DEBUG2 field descriptions

Field	Description
31–28 RSVD1	Always write zeroes to this bit field.
27–24 UDMA_STATE	USM_IDLE = 4'h0, idle USM_DMARQ = 4'h1, DMA req USM_ACK = 4'h2, DMA ACK USM_FIFO_E = 4'h3, Fifo empty USM_WPAUSE = 4'h4, WR DMA Paused by device USM_TSTRB = 4'h5, Toggle HSTROBE USM_CAPTUR = 4'h6, Capture Stage, (data sampled with DSTROBE is valid) USM_DATOUT = 4'h7, Change Burst DATAOUT USM_CRC = 4'h8, Source CRC to Device USM_WAIT_R = 4'h9, Waiting for DDMARDY- USM_END = 4'ha; Negate DMAACK (end of DMA) USM_WAIT_S = 4'hb, Waiting for DSTROBE USM_RPAUSE = 4'hc, Rd DMA Paused by Host USM_RSTOP = 4'hd, Rd DMA Stopped by Host USM_WTERM = 4'he, Wr DMA Termination State USM_RTERM = 4'hf, Rd DMA Termination state

Table continues on the next page...

GPMI_DEBUG2 field descriptions (continued)

Field	Description
23 BUSY	When asserted the GPMI is busy. Undefined results may occur if any registers are written when BUSY is asserted. DISABLED = 0x0 The GPMI is not busy. ENABLED = 0x1 The GPMI is busy.
22–20 PIN_STATE	parameter PSM_IDLE = 3'h0, PSM_BYTCNT = 3'h1, PSM_ADDR = 3'h2, PSM_STALL = 3'h3, PSM_STROBE = 3'h4, PSM_ATARDY = 3'h5, PSM_DHOLD = 3'h6, PSM_DONE = 3'h7. PSM_IDLE = 0x0 PSM_BYTCNT = 0x1 PSM_ADDR = 0x2 PSM_STALL = 0x3 PSM_STROBE = 0x4 PSM_ATARDY = 0x5 PSM_DHOLD = 0x6 PSM_DONE = 0x7
19–16 MAIN_STATE	parameter MSM_IDLE = 4'h0, MSM_BYTCNT = 4'h1, MSM_WAITFE = 4'h2, MSM_WAITFR = 4'h3, MSM_DMAREQ = 4'h4, MSM_DMAACK = 4'h5, MSM_WAITFF = 4'h6, MSM_LDFIFO = 4'h7, MSM_LDDMAR = 4'h8, MSM_RDCMP = 4'h9, MSM_DONE = 4'hA. MSM_IDLE = 0x0 MSM_BYTCNT = 0x1 MSM_WAITFE = 0x2 MSM_WAITFR = 0x3 MSM_DMAREQ = 0x4 MSM_DMAACK = 0x5 MSM_WAITFF = 0x6 MSM_LDFIFO = 0x7 MSM_LDDMAR = 0x8 MSM_RDCMP = 0x9 MSM_DONE = 0xA
15–12 SYND2GPMI_BE	Data byte enable Input from BCH.
11 GPMI2SYND_VALID	Data handshake output to BCH.
10 GPMI2SYND_READY	Data handshake output to BCH.
9 SYND2GPMI_VALID	Data handshake Input from BCH.
8 SYND2GPMI_READY	Data handshake Input from BCH.

Table continues on the next page...

GPMI_DEBUG2 field descriptions (continued)

Field	Description
7 VIEW_ DELAYED_RDN	Set to a 1 to select the delayed feedback RE_B to drive the GPMI_ADDR[0] (Nand CLE) pin. For debug purposes, this will allow you see if DLL is functioning properly.
6 UPDATE_ WINDOW	A 1 indicates that the DLL is busy generating the required delay.
RDN_TAP	This is the DLL tap calculated by the DLL controller. The selects the amount of delay form the DLL chain.

27.6.16 GPMI Debug3 Information Register Description (GPMI_DEBUG3)

The GPMI Debug3 information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

GPMI_DEBUG3 0x0F0

Address: 180_6000h base + F0h offset = 180_60F0h

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

GPMI_DEBUG3 field descriptions

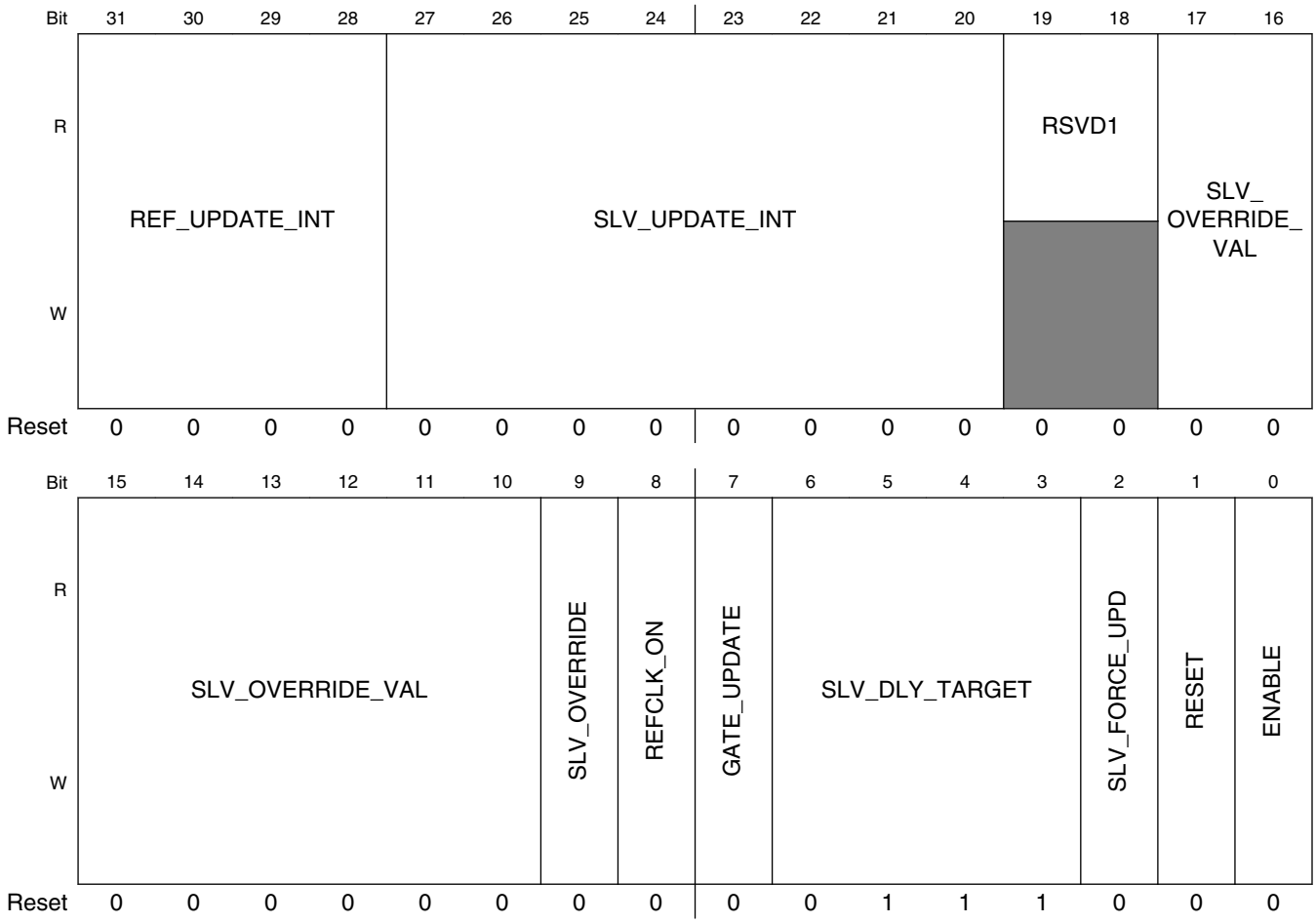
Field	Description
31–16 APB_WORD_ CNTR	Reflects the number of bytes remains to be transferred on the APB bus.
DEV_WORD_ CNTR	Reflects the number of bytes remains to be transferred on the ATA/Nand bus.

27.6.17 GPMI Double Rate Read DLL Control Register Description (GPMI_READ_DDR_DLL_CTRL)

GPMI DDR Read Delay Loop Lock Control Register. This register provides programmability in DDR mode for data input timing and data formats.

GPMI_READ_DDR_DLL_CTRL 0x100

Address: 180_6000h base + 100h offset = 180_6100h



GPMI_READ_DDR_DLL_CTRL field descriptions

Field	Description
31–28 REF_UPDATE_ INT	This field allows the user to add additional delay cycles to the DLL control loop (reference delay line control). By default, the DLL control loop shall update every two GPMICLK cycles. Programming this field results in a DLL control loop update interval of (2 + REF_UPDATE_INT) * GPMICLK. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 SLV_UPDATE_ INT	Setting a value greater than 0 in this field, shall over-ride the default slave delay-line update interval of 256 GPMICLK cycles. A value of 0 results in an update interval of 256 GPMICLK cycles (default setting). A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–18 RSVD1	Reserved
17–10 SLV_ OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 256 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 256.
9 SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0

Table continues on the next page...

GPMI_READ_DDR_DLL_CTRL field descriptions (continued)

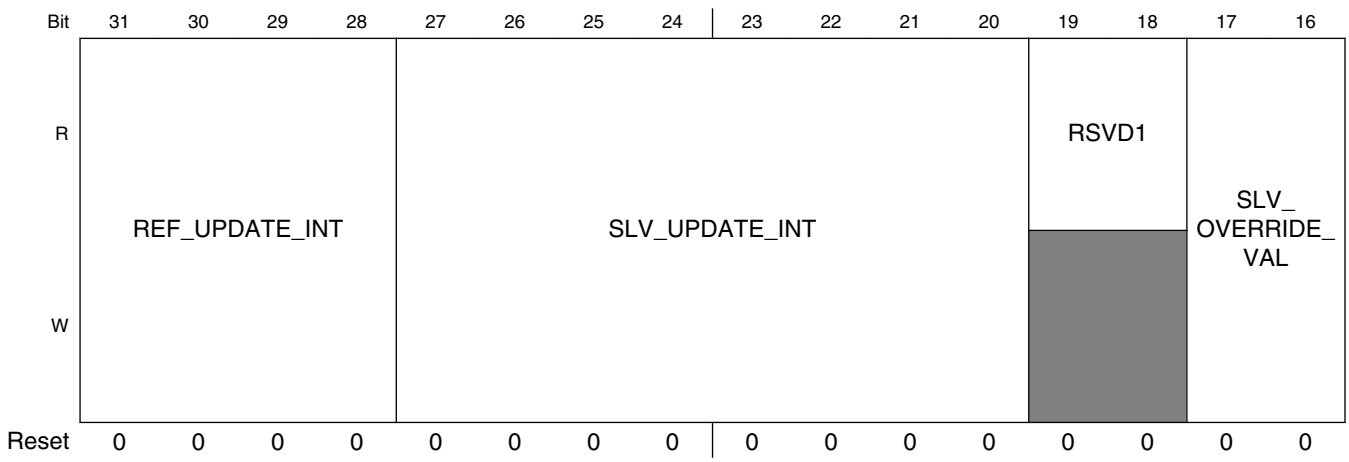
Field	Description
8 REFCLK_ON	set this bit to 1 will turn on the reference clock
7 GATE_UPDATE	Setting this bit to 1, forces the slave delay line not update
6–3 SLV_DLY_ TARGET	The delay target for the read clock is can be programmed in 1/16th increments of an GPMICLK half-period. So the input read-clock can be delayed relative input data from (GPMICLK/2)/16 to GPMICLK/2.
2 SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered).
1 RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an GPMICLK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

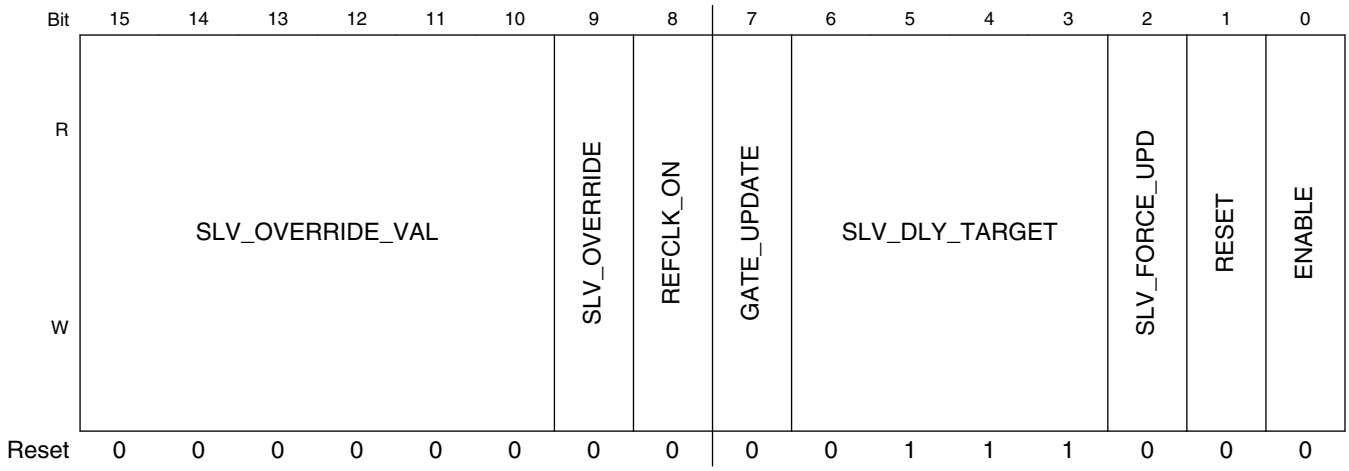
27.6.18 GPMI Double Rate Write DLL Control Register Description (GPMI_WRITE_DDR_DLL_CTRL)

GPMI DDR Write Delay Loop Lock Control Register. This register provides programmability in DDR mode for data output timing and data formats.

GPMI_WRITE_DDR_DLL_CTRL 0x110

Address: 180_6000h base + 110h offset = 180_6110h





GPMI_WRITE_DDR_DLL_CTRL field descriptions

Field	Description
31–28 REF_UPDATE_INT	This field allows the user to add additional delay cycles to the DLL control loop (reference delay line control). By default, the DLL control loop shall update every two GPMICLK cycles. Programming this field results in a DLL control loop update interval of (2 + REF_UPDATE_INT) * GPMICLK. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 SLV_UPDATE_INT	Setting a value greater than 0 in this field, shall over-ride the default slave delay-line update interval of 256 GPMICLK cycles. A value of 0 results in an update interval of 256 GPMICLK cycles (default setting). A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–18 RSVD1	Reserved
17–10 SLV_OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 256 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 256.
9 SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0
8 REFCLK_ON	set this bit to 1 will turn on the reference clock
7 GATE_UPDATE	Setting this bit to 1, forces the slave delay line not update
6–3 SLV_DLY_TARGET	The delay target for the read clock can be programmed in 1/16th increments of an GPMICLK half-period. So the input read-clock can be delayed relative input data from (GPMICLK/2)/16 to GPMICLK/2.
2 SLV_FORCE_UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered).
1 RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an GPMICLK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

27.6.19 GPMI Double Rate Read DLL Status Register Description (GPMI_READ_DDR_DLL_STS)

GPMI Double Rate Read DLL Status Register, Read Only. GPMI DLL status fields are provided in this register.

GPMI_READ_DDR_DLL_STS 0x120

Address: 180_6000h base + 120h offset = 180_6120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1								REF_SEL							REF_LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								SLV_SEL							SLV_LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_READ_DDR_DLL_STS field descriptions

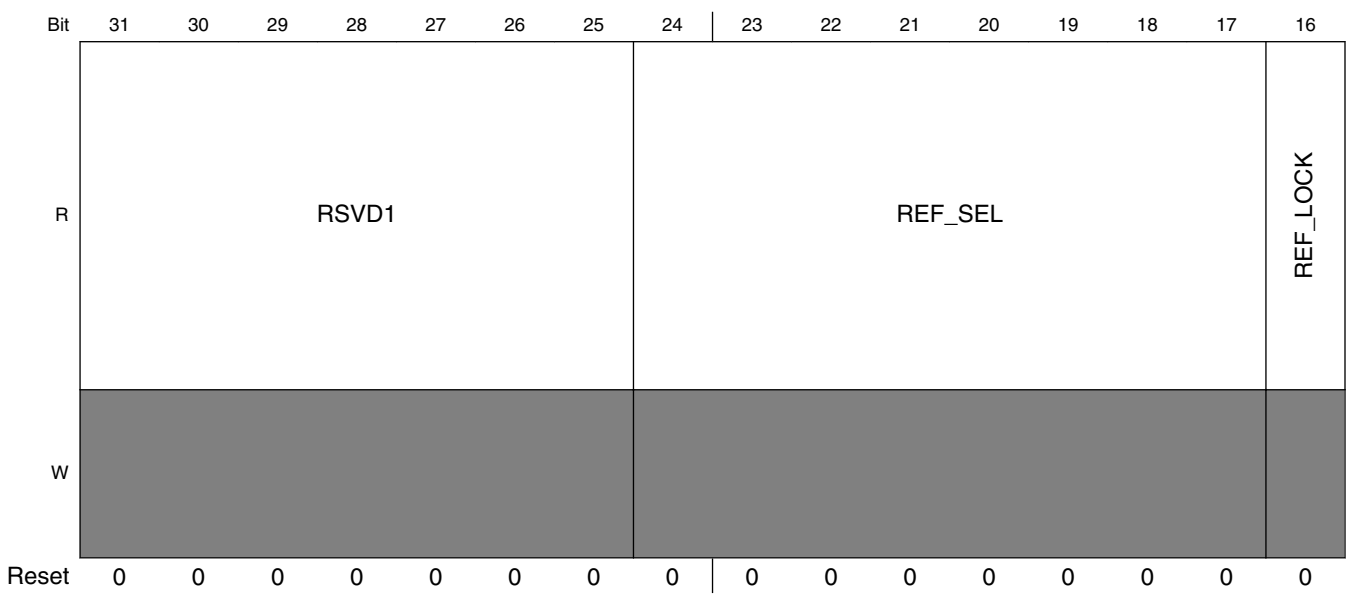
Field	Description
31–25 RSVD1	Reserved
24–17 REF_SEL	Reference delay line select status.
16 REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase GPMICLK shift, allowing the slave delay-line to perform programmed clock delays.
15–9 RSVD0	Reserved
8–1 SLV_SEL	Slave delay line select status
0 SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

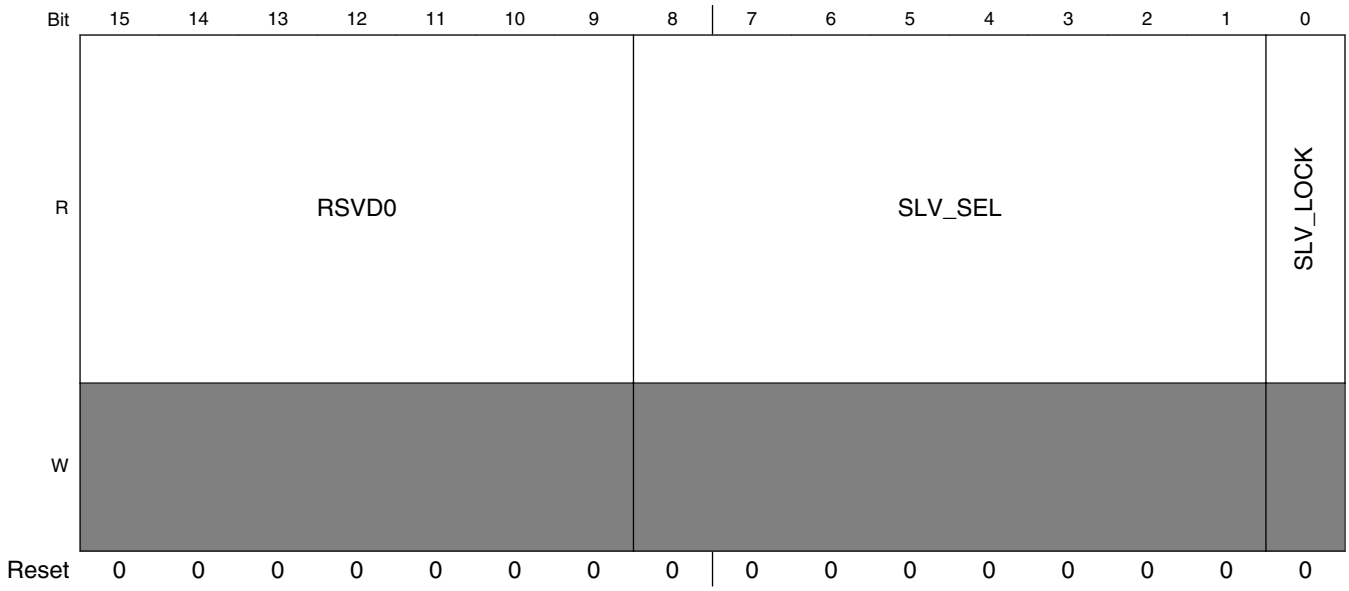
27.6.20 GPMI Double Rate Write DLL Status Register Description (GPMI_WRITE_DDR_DLL_STS)

GPMI Double Rate Write DLL Status Register, Read Only. GPMI DLL status fields are provided in this register.

GPMI_WRITE_DDR_DLL_STS 0x130

Address: 180_6000h base + 130h offset = 180_6130h





GPMI_WRITE_DDR_DLL_STS field descriptions

Field	Description
31–25 RSVD1	Reserved
24–17 REF_SEL	Reference delay line select status.
16 REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase GPMICLK shift, allowing the slave delay-line to perform programmed clock delays.
15–9 RSVD0	Reserved
8–1 SLV_SEL	Slave delay line select status
0 SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

Chapter 28
General Purpose Timer (GPT)

28.1 Overview

This chapter describes the General Purpose Timer (GPT) module interface. It is also a reference for software driver programming.

The GPT has a 32-bit up-counter. The timer counter value can be captured in a register using an event on an external pin. The capture trigger can be programmed to be a rising or/and falling edge. The GPT can also generate an event on the DO_CMPOUT*n* pins and an interrupt when the timer reaches a programmed value. The GPT has a 12-bit prescaler, which provides a programmable clock frequency derived from multiple clock sources.

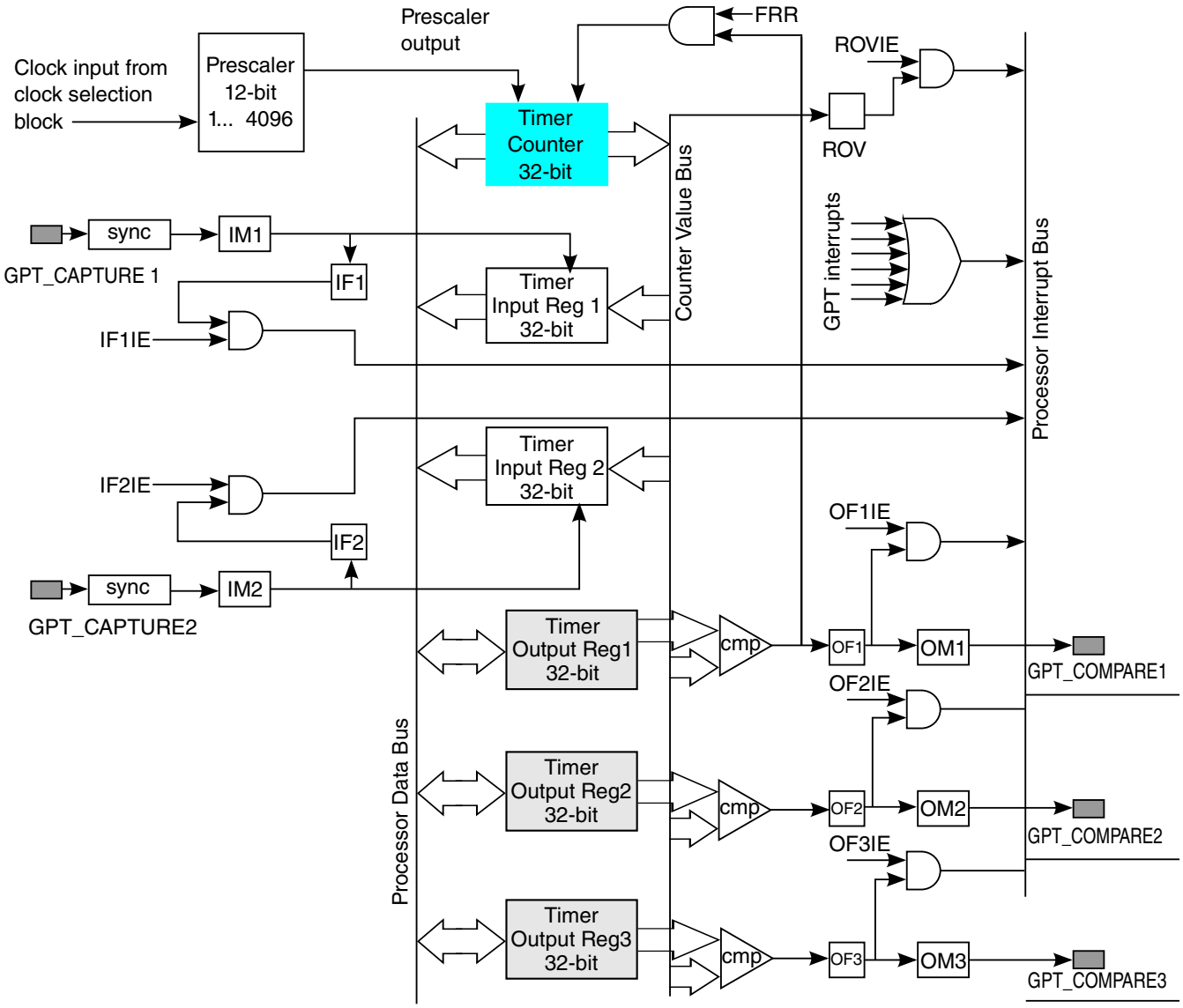


Figure 28-1. GPT Block Diagram

The following figure shows the GPT functional clocking scheme.

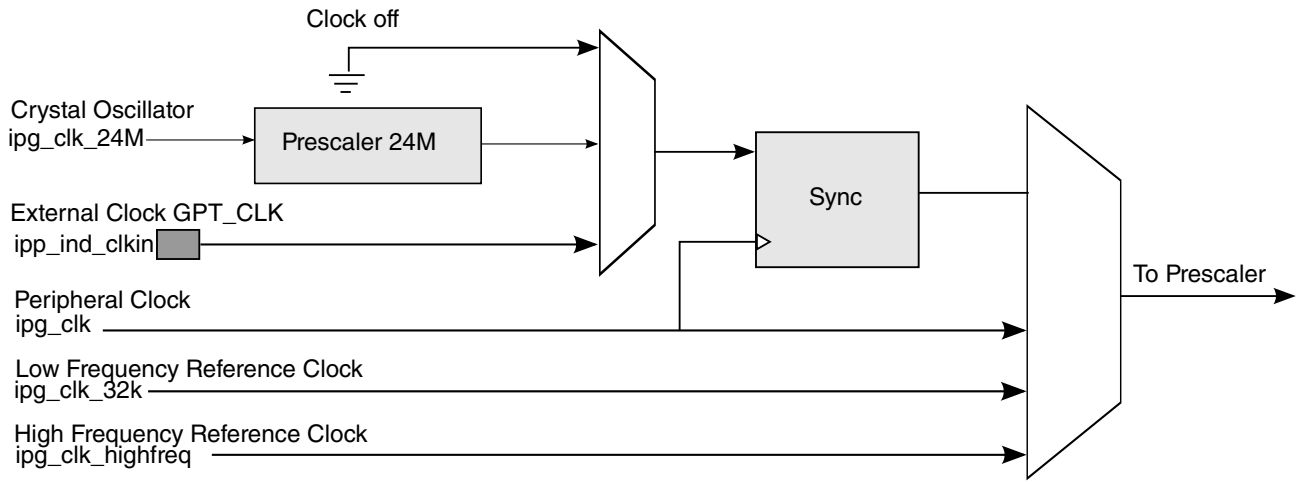


Figure 28-2. GPT Counter Clocks Diagram

28.1.1 Features

- One 32-bit up-counter with clock source selection, including external clock.
- Two input capture channels with a programmable trigger edge.
- Three output compare channels with a programmable output mode. A "forced compare" feature is also available.
- Can be programmed to be *active* in low power and debug modes.
- Interrupt generation at capture, compare, and rollover events.
- Restart or free-run modes for counter operations.

28.1.2 Modes and Operation

The GPT supports the modes described in the indicated sections:

- Operating Modes
  - Restart Mode
  - Free-Run Mode

28.2 External Signals

The GPT follows the IP Bus protocol for interfacing with the processor core. The GPT does not have *any interface signals with any other module inside the chip,* except for the clock and reset inputs (from the clock and reset controller module) and for the interrupt signals *to* the processor interrupt handler. There are functional and clock inputs, and functional output signals going outside the chip boundary.

The following table describes all block signals that connect off-chip.

Table 28-1. GPT External Signals

Signal	Description	Pad	Mode	Direction
GPT1_CLK	Input pin for an external clock that the counter can be operated at.	ENET1_TX_CLK	ALT8	I
		UART1_RX_DATA	ALT4	
GPT1_CAPTURE1	Input pin for a capture event for Input Capture Channel 1.	GPIO1_IO00	ALT1	I
		UART2_TX_DATA	ALT4	
GPT1_CAPTURE2	Input pin for a capture event for Input Capture Channel 2.	ENET1_RX_ER	ALT8	I
		UART2_RX_DATA	ALT4	
GPT1_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	GPIO1_IO01	ALT1	O
		UART1_TX_DATA	ALT4	
GPT1_COMPARE2	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	GPIO1_IO02	ALT1	O
		UART2_CTS_B	ALT4	
GPT1_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	GPIO1_IO03	ALT1	O
		UART2_RTS_B	ALT4	
GPT2_CLK	Input pin for an external clock that the counter can be operated at.	JTAG_MODE	ALT1	I
		SD1_DATA1	ALT1	
GPT2_CAPTURE1	Input pin for a capture event for Input Capture Channel 1.	JTAG_TMS	ALT1	I
		SD1_DATA2	ALT1	
GPT2_CAPTURE2	Input pin for a capture event for Input Capture Channel 2.	JTAG_TDO	ALT1	I
		SD1_DATA3	ALT1	
GPT2_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	JTAG_TDI	ALT1	O
		SD1_CMD	ALT1	
GPT2_COMPARE2	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	JTAG_TCK	ALT1	O
		SD1_CLK	ALT1	
GPT2_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	JTAG_TRST_B	ALT1	O
		SD1_DATA0	ALT1	

There are six signals (three input, three output) in the GPT module that *can be* connected to the chip pads.

28.2.1 External Clock Input

The GPT counter can be operated using an external clock from outside the device, and this is the input pin used for that purpose.

The external clock input GPT_CLK is treated as asynchronous to the peripheral clock. To ensure proper operations of GPT, the external clock input frequency should be less than 1/4 of frequency of the peripheral clock. Hysteresis characteristics on this pad will be required because this is a clock input.

28.2.2 Input Capture Trigger Signals

The GPT counter value can be stored in a register, triggered by an event from *outside the device*.

A positive or/and negative edge on these signals GPT_CAPTURE1 , GPT_CAPTURE2 can trigger this capture event. These signals are treated as asynchronous to the peripheral clock. Only those transitions which occur *at least a single clock cycle* (the clock selected to run the counter) *after the previous recorded transition* are guaranteed to trigger a capture event.

28.2.3 Output Compare Signals

The output compare signals: GPT_COMPARE1, GPT_COMPARE2, GPT_COMPARE3, indicate that output compare events have gone through a specified transition.

28.4 Clocks

The clock that is input to the prescaler can be selected from 4 clock sources:

The following table describes the clock sources for GPT. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 28-2. GPT Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock

Table continues on the next page...

Table 28-2. GPT Clocks (continued)

Clock name	Clock Root	Description
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32 kHz)
ipg_clk_highfreq	perclk_clk_root	High-frequency reference clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

- High-Frequency Clock (ipg_clk_highfreq)

  Provided by the Clock Controller Module (CCM), the High Frequency Clock is intended to be ON in Normal Power mode when the Peripheral Clock is turned OFF, thereby enabling the GPT to be operated using the High Frequency Clock *in Normal Power mode*. The CCM is expected to provide this clock *after* synchronizing it to the System Bus Clock in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the High Frequency Clock in a Low Power mode.

- Low-Reference Clock (ipg_clk_32k)

  This 32 kHz Low Reference Clock (provided by the CCM) is intended to be ON in Low Power mode when the Peripheral Clock is turned OFF, thereby enabling the GPT to be operated using the Low Reference Clock in Low Power mode. The CCM is expected to provide the Low Reference Clock *after* synchronizing it to the System Bus Clock in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the Low Reference Clock in a Low Power mode.

- Peripheral Clock (ipg_clk)

  If the Peripheral Clock or the External Clock is selected(CLKSRC=001 or 011) as Clock Source, then the Peripheral Clock will be ON in normal GPT operations. In Low Power modes, if the GPT is programmed to be disabled (STOPEN or WAITEN or DOZEN=0), then the Peripheral Clock can be switched OFF.

- External Clock

  The External Clock comes from *outside the device* and can be selected to run the GPT counter. The External Clock is treated as *asynchronous to the Peripheral Clock,* and is synchronized to the Peripheral Clock, *inside* the module. Therefore, the External Clock frequency is limited to < 1/4 frequency of the Peripheral Clock, for proper GPT operations. Note that in Low Power modes, *if* the Peripheral Clock is not available, then the External Clock *cannot be used* to run the counter.

- Crystal Oscillator Clock

  This 24 MHz Crystal Oscillator Clock (provided by the CCM) is intended to be used against frequency change of Peripheral Clock changes to provide a more accurate timer clock for operation system. The CCM is expected to provide the 24 MHz

Crystal Oscillator Clock *without* synchronizing it to the System Bus Clock in Normal functional mode. Synchronization is done in GPT module. Before synchronization, the 24 MHz Crystal Oscillator Clock is divided by a 24 MHz clock prescaler, to make sure the clock frequency less than half of System Bus Clock .

The clock input source is configured using the clock source field (CLKSRC, in the GPT_CR control register). The clock input to the prescaler can be disabled by programming the CLKSRC bits (of the GPT_CR control register) to 000. **The CLKSRC field value should be changed** *only after disabling the GPT* (by setting the EN bit in the GPT_CR to 0).

The PRESCALER field selects the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value (from 1 to 4096) and can be changed *at any time*. A change in the value of the PRESCALER field *immediately affects* the output clock frequency.

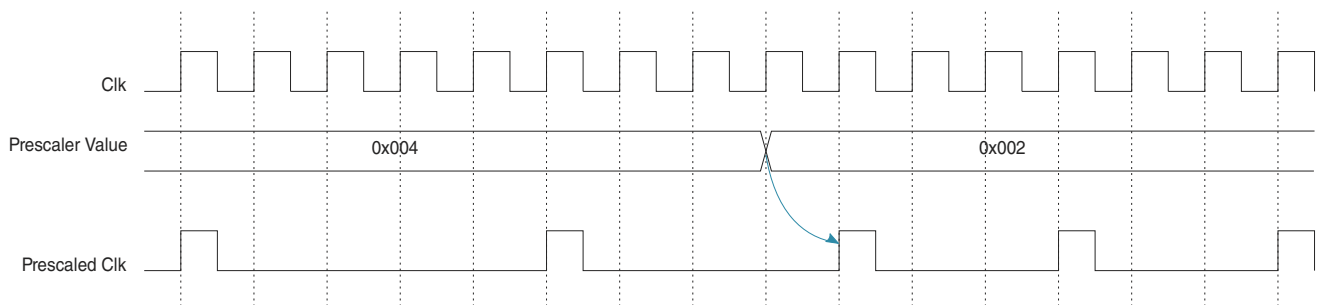


Figure 28-3. Prescaler Value Change Timing Diagram

28.5 Functional Description

This section provides a complete functional description of the GPT.

28.5.1 Operating Modes

The GPT counter can be programmed to work in either of two modes: Restart mode or Free-Run mode.

28.5.1.1 Restart Mode

In Restart mode (selectable through the GPT Control Register GPT_CR), when the counter reaches the compared value, the counter resets and starts again from 0x00000000. The Restart feature is associated only with Compare Channel 1.

Any write access to the Compare register of Channel 1 will reset the GPT counter. This is done to avoid possibly missing a compare event when compare value is changed from a higher value to lower value while counting is proceeding.

For the other two compare channels, when the compare event occurs the counter is *not reset*.

28.5.1.2 Free-Run Mode

In Free-Run mode, when compare events occur for all 3 channels, the counter is *not reset*; instead the counter continues to count until 0xffffffff, and then rolls over (to 0x00000000).

28.5.2 Operation

The General Purpose Timer (GPT) has a single counter (GPT_CNT) that is a 32-bit free-running *up-counter,* which starts counting *after it is enabled by software* (EN=1).

The counter's clock source is the output of the prescaler labelled "Prescaler output" in Figure 28-1.

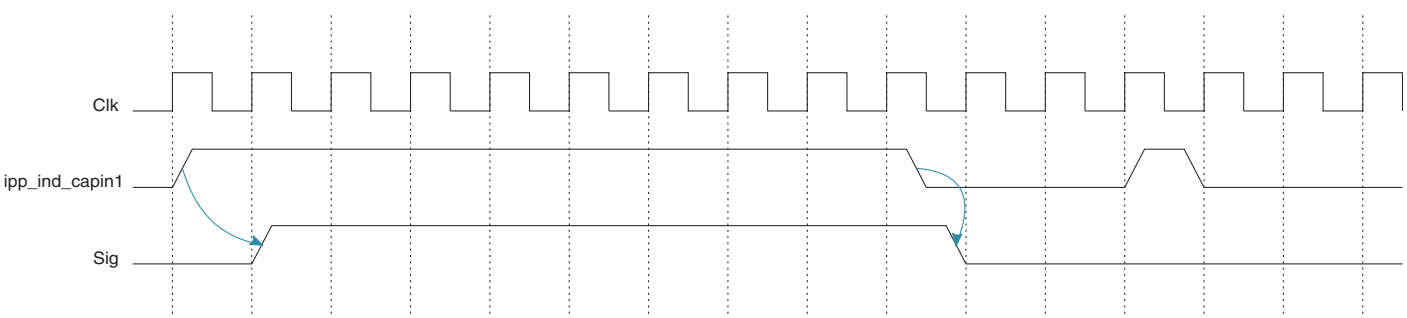
- If the GPT timer is disabled (EN=0), then the Main Counter *and* Prescaler Counter freeze their current count values. The ENMOD bit determines the value of the GPT counter when the EN bit is set and the Counter is enabled again.
  - If the ENMOD bit is set (=1), then the Main Counter and Prescaler Counter values are reset to 0, when GPT is enabled (EN=1).
  - If ENMOD bit is programmed to 0, then the Main Counter and Prescaler Counter restart counting from their frozen values, when GPT is enabled again (EN=1).
- If GPT is programmed to be disabled in a low power mode (STOP/WAIT), then the Main Counter and Prescaler Counter freeze at their current count values *when* GPT enters low power mode. When GPT exits a low power mode, the Main Counter and Prescaler Counter start counting from their frozen values *regardless* of the ENMOD bit value. Note that the GPT_CNT can be read *at any time* by the processor, and that *both* Input Capture Channels use the *same* counter (GPT_CNT).

- A hardware reset resets all the GPT registers to their respective reset values. All registers except the Output Compare Registers (OCR1, OCR2, OCR3) obtain a value of 0x0. The Compare registers are reset to 0xffffffff.
- The software reset (SWR bit in the GPT_CR control register) resets *all* of the register bits *except* the EN, ENMOD, STOPEN, WAITEN, and DBGEN bits. The state of these bits is not affected by a software reset. Note that a software reset can be given *while the GPT is disabled*.

28.5.2.1 Input Capture

There are two Input Capture Channels, and each Input Capture Channel has a dedicated capture pin, capture register and input edge detection/selection logic. Each input capture function has an associated status flag, and can cause the processor to make an interrupt service request.

When a selected edge transition occurs on an Input Capture pin, the contents of the GPT_CNT is captured on the corresponding capture register and the appropriate interrupt status flag is set. An interrupt request can be generated when the transition is detected *if* its corresponding enable bit is set (in the Interrupt Register). The capture can be programmed to occur on the input pin's rising edge, falling edge, on both rising and falling edges, or the capture can be disabled. The events are synchronized with the clock that was selected to run the counter. Only those transitions that occur at least one clock cycle (clock selected to run the counter) *after* the previous recorded transition will be guaranteed to trigger a capture event. There can be up to one clock cycle of uncertainty in the latching of the input transition. The Input Capture registers can be read *at any time* without affecting their values.



Clk = The clock selected from CLKSRC bit field setting

ipp_ind_capin1 = Pad signal for capture channel 1

Sig = Capture signal as sensed by the module

Figure 28-4. Input Capture Event Timing

28.5.2.2 Output Compare

The three Output Compare Channels *use the same counter* (GPT_CNT) as the Input Capture Channels. When the programmed content of an Output Compare register matches the value in GPT_CNT, an output compare status flag is set and an interrupt is generated (if the corresponding bit is set in the interrupt register). Consequently, the Output Compare timer pin will be set, cleared, toggled, not affected at all or provide an active-low pulse for one input clock period (subject to the restriction on the maximum frequency allowed on the pad) according to the mode bits (that were programmed).

There is also a "forced-compare" feature that allows the software to generate a compare event when required, *without the condition of the counter value that is equal to the compare value*. The action taken as a result of a forced compare is the same as when an output compare match occurs, *except that the status flags are not set and no interrupt can be generated*. Forced channels take programmed action immediately after the write to the force-compare bits. These bits are self-negating and always read as zeros.

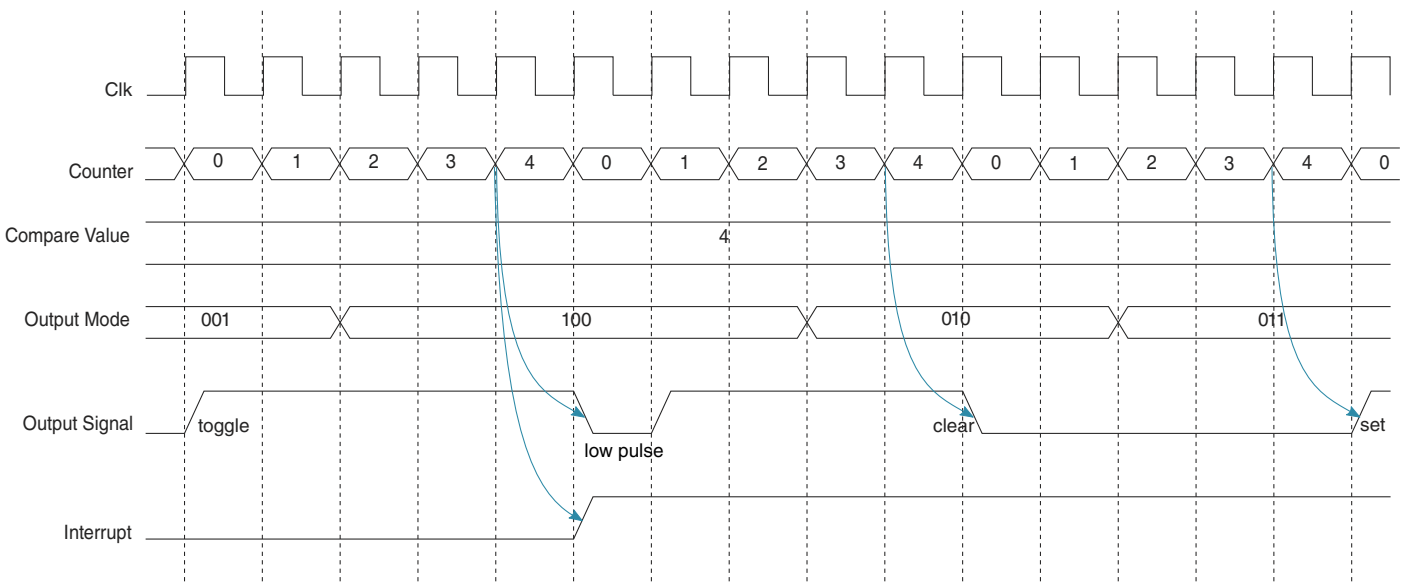


Figure 28-5. Output Compare and Interrupt Timing

28.5.2.3 Interrupts

There are 6 different interrupts that are generated by the GPT. If the selected clock for running the counter is available, then *all interrupts can be generated in Low Power and Debug modes*.

- Rollover Interrupt

The Rollover Interrupt is generated when the GPT counter reaches 0xffffffff, then resets to 0x00000000 and continues counting. The Rollover Interrupt is enabled by the ROVIE bit in the GPT_IR register; the associated status bit is the ROV bit in the GPT_SR register.

- Input Capture Interrupt 1, 2

  After a capture event occurs, the associated Input Capture Channel generates an interrupt. The "capture event" interrupts are enabled by the IF2IE and IF1IE bits (in the GPT_IR register); the associated status bits are IF2 and IF1 (in the GPT_SR register). The capture of the counter value because of a capture event is *not affected by a pending capture interrupt*. The Capture register is updated with a new counter value when a capture event occurs, regardless of whether that Capture Channels' interrupt has been serviced or not.

- Output Compare Interrupt 1, 2, 3

  After a compare event occurs, the associated Output Compare Channel generates an interrupt. The "compare event" interrupts are enabled by the OF3IE, OF2IE, and OF1IE bits (in the GPT_IR register); the associated status bits are OF3, OF2, and OF1 (in the GPT_SR register). A "forced compare" does not generate an interrupt.

A *cumulative* interrupt line is also present, which is asserted whenever any of the above interrupts are posted. The cumulative interrupt line has *no* associated enables or status bits.

28.5.2.4 Low Power Mode Behavior

In Low Power modes, if the clock from the selected clock source is available (except for the External Clock, which can be used *only if* the Peripheral Clock is available), the counter will continue to run depending on whether the control bit for that mode is set. If the clock is not present or if the corresponding low power bit in the GPT_CR control register is 0, the Main Counter and the Prescaler Counter freeze at their current values and resume counting (from their frozen values) when the Low Power mode is exited.

28.5.2.5 Debug Mode Behavior

In Debug mode, the modules in the device have the option of continuing to run or be halted.

- If the DBGEN bit is set, then the GPT timer will continue to run in Debug mode.
- If the DBGEN bit is not set (in the GPT_CR control register), then the GPT timer is halted.

28.6 Initialization/ Application Information

28.6.1 Selecting the Clock Source

The CLKSRC field in the GPT_CR register selects the clock source. The CLKSRC field value should be changed only after disabling the GPT (EN=0).

The software sequence to be followed while changing clock source is:

1. Disable GPT by setting EN=0 in GPT_CR register.
2. Disable GPT interrupt register (GPT_IR).
3. Configure Output Mode to unconnected/ disconnected—Write zeros in OM3, OM2, and OM1 in GPT_CR
4. Disable Input Capture Modes—Write zeros in IM1 and IM2 in GPT_CR
5. Change clock source CLKSRC to the desired value in GPT_CR register.
6. Assert the SWR bit in GPT_CR register.
7. Clear GPT status register (GPT_SR) (i.e., w1c).
8. Set ENMOD=1 in GPT_CR register, to bring GPT counter to 0x00000000.
9. Enable GPT (EN=1) in GPT_CR register.
10. Enable GPT interrupt register (GPT_IR).

28.7 GPT Memory Map/Register Definition

The GPT has 10 user-accessible 32-bit registers, which are used to configure, operate, and monitor the state of the GPT.

An IP bus write access to the GPT Control Register (GPT_CR) and the GPT Output Compare Register1 (GPT_OCR1) results in *one cycle of wait state*, while other valid IP bus accesses incur 0 wait states.

Irrespective of the Response Select signal value, a Write access to the GPT Status Registers (Read-only registers GPT_ICR1, GPT_ICR2, GPT_CNT) will generate a bus exception.

- If the Response Select signal is driven Low, then the Read/Write access to the *unimplemented* address space of GPT (*ips_addr* is greater than or equal to $BASE + $028) will generate a bus exception.
- If the Response Select is driven High, then the Read/Write access to the unimplemented address space of GPT will *not* generate any error response (like a bus exception).

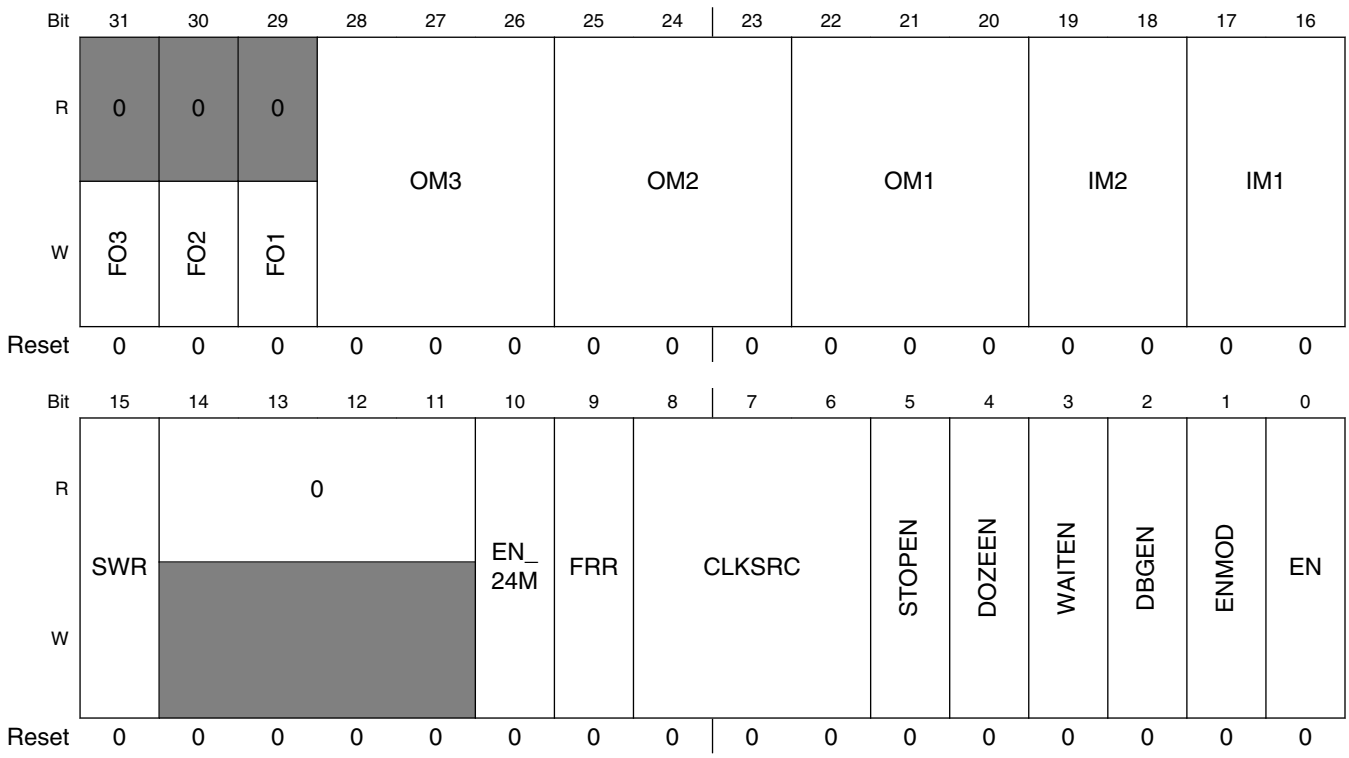
GPT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
209_8000	GPT Control Register (GPT1_CR)	32	R/W	0000_0000h	28.7.1/1230
209_8004	GPT Prescaler Register (GPT1_PR)	32	R/W	0000_0000h	28.7.2/1234
209_8008	GPT Status Register (GPT1_SR)	32	R/W	0000_0000h	28.7.3/1235
209_800C	GPT Interrupt Register (GPT1_IR)	32	R/W	0000_0000h	28.7.4/1236
209_8010	GPT Output Compare Register 1 (GPT1_OCR1)	32	R/W	FFFF_FFFFh	28.7.5/1237
209_8014	GPT Output Compare Register 2 (GPT1_OCR2)	32	R/W	FFFF_FFFFh	28.7.6/1238
209_8018	GPT Output Compare Register 3 (GPT1_OCR3)	32	R/W	FFFF_FFFFh	28.7.7/1238
209_801C	GPT Input Capture Register 1 (GPT1_ICR1)	32	R	0000_0000h	28.7.8/1239
209_8020	GPT Input Capture Register 2 (GPT1_ICR2)	32	R	0000_0000h	28.7.9/1239
209_8024	GPT Counter Register (GPT1_CNT)	32	R	0000_0000h	28.7.10/ 1240
20E_8000	GPT Control Register (GPT2_CR)	32	R/W	0000_0000h	28.7.1/1230
20E_8004	GPT Prescaler Register (GPT2_PR)	32	R/W	0000_0000h	28.7.2/1234
20E_8008	GPT Status Register (GPT2_SR)	32	R/W	0000_0000h	28.7.3/1235
20E_800C	GPT Interrupt Register (GPT2_IR)	32	R/W	0000_0000h	28.7.4/1236
20E_8010	GPT Output Compare Register 1 (GPT2_OCR1)	32	R/W	FFFF_FFFFh	28.7.5/1237
20E_8014	GPT Output Compare Register 2 (GPT2_OCR2)	32	R/W	FFFF_FFFFh	28.7.6/1238
20E_8018	GPT Output Compare Register 3 (GPT2_OCR3)	32	R/W	FFFF_FFFFh	28.7.7/1238
20E_801C	GPT Input Capture Register 1 (GPT2_ICR1)	32	R	0000_0000h	28.7.8/1239
20E_8020	GPT Input Capture Register 2 (GPT2_ICR2)	32	R	0000_0000h	28.7.9/1239
20E_8024	GPT Counter Register (GPT2_CNT)	32	R	0000_0000h	28.7.10/ 1240

28.7.1 GPT Control Register (GPTx_CR)

The GPT Control Register (GPT_CR) is used to program and configure GPT operations. An IP Bus Write to the GPT Control Register occurs after one cycle of wait state, while an IP Bus Read occurs after 0 wait states.

Address: Base address + 0h offset



GPTx_CR field descriptions

Field	Description
31 FO3	FO3 Force Output Compare Channel 3 FO2 Force Output Compare Channel 2 FO1 Force Output Compare Channel 1 The FO*n* bit causes the pin action *programmed* for the timer Output Compare *n* pin (according to the OM*n* bits in this register). • The OF*n* flag (OF3, OF2, OF1) in the status register is **not affected**. • This bit is self-negating and always read as zero. 0 Writing a 0 has no effect. 1 Causes the programmed pin action on the timer Output Compare *n* pin; the OF*n* flag is not set.
30 FO2	See F03

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
29 FO1	See F03
28–26 OM3	OM3 (bits 28-26) controls the Output Compare Channel 3 operating mode. OM2 (bits 25-23) controls the Output Compare Channel 2 operating mode. OM1 (bits 22-20) controls the Output Compare Channel 1 operating mode. The OM*n* bits specify the response that a compare event will generate on the output pin of Output Compare Channel *n*. • The toggle, clear, and set options cause a change on the output pin *only* if a compare event occurs. • When OM*n* is programmed as 1xx (active low pulse), the output pin is set to one immediately on the next input clock; a low pulse (that is an input clock in width) occurs when there is a compare event. Note that here, "input clock" refers to the clock selected by the CLKSRC bits of the GPT Control Register. 000 Output disconnected. No response on pin. 001 Toggle output pin 010 Clear output pin 011 Set output pin 1xx Generate an active low pulse (that is one input clock wide) on the output pin.
25–23 OM2	See OM3
22–20 OM1	See OM3
19–18 IM2	IM2 (bits 19-18, Input Capture Channel 2 operating mode) IM1 (bits 17-16, Input Capture Channel 1 operating mode) The IM*n* bit field determines the transition on the input pin (for Input capture channel *n*), which will trigger a capture event. 00 capture disabled 01 capture on rising edge only 10 capture on falling edge only 11 capture on both edges
17–16 IM1	See IM2
15 SWR	Software reset. This is the software reset of the GPT module. It is a self-clearing bit. • The SWR bit is set when the module is in reset state. • The SWR bit is cleared when the reset procedure finishes. • Setting the SWR bit resets **all of the registers** to their default reset values, except for the CLKSRC, EN, ENMOD, STOPEN, WAITEN, and DBGEN bits in the GPT Control Register (this control register). 0 GPT is not in reset state 1 GPT is in reset state
14–11 Reserved	This read-only field is reserved and always has the value 0.
10 EN_24M	Enable 24 MHz clock input from crystal.

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
	• A hardware reset resets the EN_24M bit. • A software reset *does not affect* the EN_24M bit. 0 24M clock disabled 1 24M clock enabled
9 FRR	Free-Run or Restart mode. The FFR bit determines the behavior of the GPT when a compare event in channel 1 occurs. • In Restart mode, after a compare event, the counter resets to 0x00000000 and resumes counting (after the occurrence of a compare event). • In Free-Run mode, after a compare event, the counter continues counting until 0xFFFFFFFF and then rolls over to 0. 0 Restart mode 1 Free-Run mode
8–6 CLKSRC	Clock Source select. The CLKSRC bits select which clock will go to the prescaler (and subsequently be used to run the GPT counter). • The CLKSRC bit field value should only be changed after disabling the GPT by clearing the EN bit in this register (GPT_CR). • A software reset does not affect the CLKSRC bit. 000 No clock 001 Peripheral Clock 010 High Frequency Reference Clock 011 External Clock (CLKIN) 100 Low Frequency Reference Clock others Reserved
5 STOPEN	GPT Stop Mode enable. The STOPEN read/write control bit enables GPT operation *during Stop mode*. • A hardware reset resets the STOPEN bit. • A software reset *does not affect* the STOPEN bit. 0 GPT is disabled in Stop mode. 1 GPT is enabled in Stop mode.
4 DOZEEN	GPT Doze Mode Enable. • A hardware reset resets the DOZEEN bit. • A software reset *does not affect* the DOZEEN bit. 0 GPT is disabled in doze mode. 1 GPT is enabled in doze mode.
3 WAITEN	GPT Wait Mode enable. The WAITEN read/write control bit enables GPT operation *during Wait mode*. • A hardware reset resets the WAITEN bit. • A software reset *does not affect* the WAITEN bit. 0 GPT is disabled in wait mode. 1 GPT is enabled in wait mode.

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
2 DBGEN	GPT debug mode enable. The DBGEN read/write control bit enables GPT operation *during Debug mode*. • A hardware reset resets the DBGEN bit. • A software reset *does not affect* the DBGEN bit. 0 GPT is disabled in debug mode. 1 GPT is enabled in debug mode.
1 ENMOD	GPT Enable mode. When the GPT is disabled (EN=0), then both the Main Counter and Prescaler Counter *freeze their current count values*. The ENMOD bit determines the value of the GPT counter when Counter is enabled again (if the EN bit is set). • If the ENMOD bit is 1, then the Main Counter and Prescaler Counter values are reset to 0 after GPT is enabled (EN=1). • If the ENMOD bit is 0, then the Main Counter and Prescaler Counter restart counting *from their frozen values* after GPT is enabled (EN=1). • If GPT is programmed to be disabled in a low power mode (STOP/WAIT), then the Main Counter and Prescaler Counter *freeze at their current count values* when the GPT enters low power mode. • When GPT exits low power mode, the Main Counter and Prescaler Counter start counting from their frozen values, regardless of the ENMOD bit value. • Setting the SWR bit will clear the Main Counter and Prescaler Counter values, regardless of the value of EN or ENMOD bits. • A hardware reset resets the ENMOD bit. • A software reset *does not affect* the ENMOD bit. 0 GPT counter will retain its value when it is disabled. 1 GPT counter value is reset to 0 when it is disabled.
0 EN	GPT Enable. The EN bit is the GPT module enable bit. **Before setting the EN bit**, we recommend that *all registers be properly programmed*. • A hardware reset resets the EN bit. • A software reset *does not affect* the EN bit. 0 GPT is disabled. 1 GPT is enabled.

28.7.2 GPT Prescaler Register (GPTx_PR)

The GPT Prescaler Register (GPT_PR) contains bits that determine the *divide value* of the clock that runs the counter.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRESCALER24M				PRESCALER											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPTx_PR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PRESCALER24M	Prescaler bits. 24M crystal clock is divided by [PRESCALER24M + 1] before selected by the CLKSRC field. If 24M crystal clock is not selected, this feild takes no effect. 0x0 Divide by 1 0x1 Divide by 2 ... ... 0xF Divide by 16
PRESCALER	Prescaler bits. The clock selected by the CLKSRC field is divided by [PRESCALER + 1], and then used to run the counter. • A change in the value of the PRESCALER bits cause the Prescaler counter to reset and a new count period to start immediately. • See Figure 28-3 for the timing diagram. 0x000 Divide by 1 0x001 Divide by 2 ... ... 0xFFF Divide by 4096

28.7.3 GPT Status Register (GPTx_SR)

The GPT Status Register (GPT_SR) contains bits that indicate that a counter has rolled over, and if any event has occurred on the Input Capture and Output Compare channels. The bits are cleared by writing a 1 to them.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										ROV	IF2	IF1	OF3	OF2	OF1
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPTx_SR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 ROV	Rollover Flag. The ROV bit indicates that the counter has reached its *maximum possible value* and *rolled over* to 0 (from which the counter continues counting). The ROV bit is only set if the counter has reached 0xFFFFFFFF in both Restart and Free-Run modes. 0 Rollover has not occurred. 1 Rollover has occurred.
4 IF2	IF2 Input capture 2 Flag IF1 Input capture 1 Flag The IF*n* bit indicates that a capture event has occurred on Input Capture channel *n*. 0 Capture event has not occurred. 1 Capture event has occurred.
3 IF1	See IF2
2 OF3	OF3 Output Compare 3 Flag OF2 Output Compare 2 Flag OF1 Output Compare 1 Flag The OF*n* bit indicates that a compare event has occurred on Output Compare channel *n*. 0 Compare event has not occurred. 1 Compare event has occurred.

Table continues on the next page...

GPTx_SR field descriptions (continued)

Field	Description
1 OF2	See OF3
0 OF1	See OF3

28.7.4 GPT Interrupt Register (GPTx_IR)

The GPT Interrupt Register (GPT_IR) contains bits that control whether interrupts are generated after rollover, input capture and output compare events.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										ROVIE	IF2IE	IF1IE	OF3IE	OF2IE	OF1IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPTx_IR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 ROVIE	Rollover Interrupt Enable. The ROVIE bit controls the Rollover interrupt. 0 Rollover interrupt is disabled. 1 Rollover interrupt enabled.
4 IF2IE	IF2IE Input capture 2 Interrupt Enable IF1IE Input capture 1 Interrupt Enable The IF*n*IE bit controls the IF*n*IE Input Capture *n* Interrupt Enable. 0 IF2IE Input Capture *n* Interrupt Enable is disabled. 1 IF2IE Input Capture *n* Interrupt Enable is enabled.

Table continues on the next page...

GPTx_IR field descriptions (continued)

Field	Description
3 IF1IE	See IF2IE
2 OF3IE	OF3IE Output Compare 3 Interrupt Enable OF2IE Output Compare 2 Interrupt Enable OF1IE Output Compare 1 Interrupt Enable The OF*n*IE bit controls the Output Compare Channel *n* interrupt. 0 Output Compare Channel *n* interrupt is disabled. 1 Output Compare Channel *n* interrupt is enabled.
1 OF2IE	See OF3IE
0 OF1IE	See OF3IE

28.7.5 GPT Output Compare Register 1 (GPTx_OCR1)

The GPT Compare Register 1 (GPT_OCR1) holds the value that determines when a compare event will be generated on Output Compare Channel 1. Any write access to the Compare register of Channel 1 while in Restart mode (FRR=0) will reset the GPT counter.

An IP Bus Write access to the GPT Output Compare Register1 (GPT_OCR1) occurs *after* one cycle of wait state; an IP Bus Read access occurs *immediately* (0 wait states).

Address: Base address + 10h offset

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

GPTx_OCR1 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 1.

28.7.6 GPT Output Compare Register 2 (GPTx_OCR2)

The GPT Compare Register 2 (GPT_OCR2) holds the value that determines when a compare event will be generated on Output Compare Channel 2.

Address: Base address + 14h offset

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

GPTx_OCR2 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 2.

28.7.7 GPT Output Compare Register 3 (GPTx_OCR3)

The GPT Compare Register 3 (GPT_OCR3) holds the value that determines when a compare event will be generated on Output Compare Channel 3.

Address: Base address + 18h offset

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

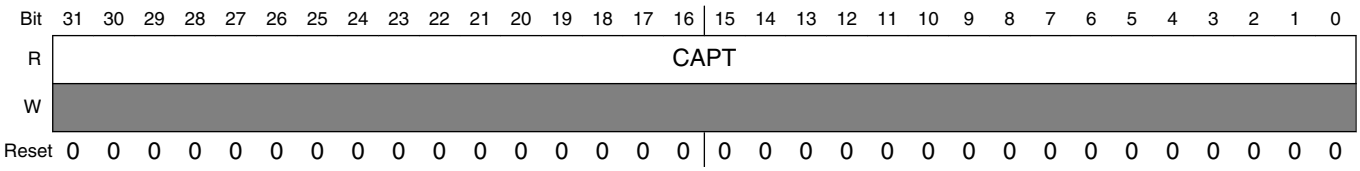
GPTx_OCR3 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 3.

28.7.8 GPT Input Capture Register 1 (GPTx_ICR1)

The GPT Input Capture Register 1 (GPT_ICR1) is a read-only register that holds the value *that was in the counter during the last capture event* on Input Capture Channel 1.

Address: Base address + 1Ch offset



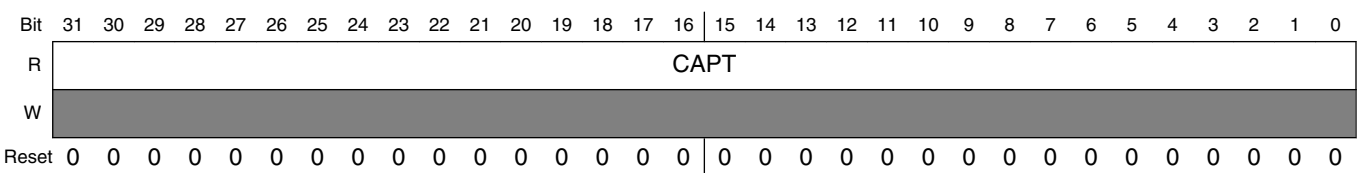
GPTx_ICR1 field descriptions

Field	Description
CAPT	Capture Value. After a capture event on Input Capture Channel 1 occurs, the current value of the counter is loaded into GPT Input Capture Register 1.

28.7.9 GPT Input Capture Register 2 (GPTx_ICR2)

The GPT Input capture Register 2 (GPT_ICR2) is a read-only register which holds the value that was in the counter during the last capture event on input capture channel 2.

Address: Base address + 20h offset



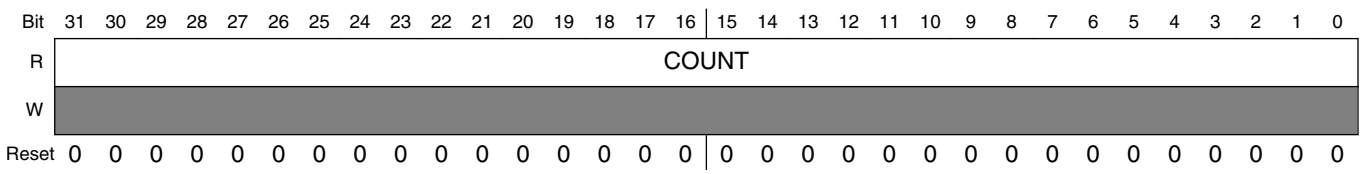
GPTx_ICR2 field descriptions

Field	Description
CAPT	Capture Value. After a capture event on Input Capture Channel 2 occurs, the current value of the counter is loaded into GPT Input Capture Register 2.

28.7.10 GPT Counter Register (GPTx_CNT)

The GPT Counter Register (GPT_CNT) is the main counter's register. GPT_CNT is a read-only register and can be read *without affecting the counting process* of the GPT.

Address: Base address + 24h offset



GPTx_CNT field descriptions

Field	Description
COUNT	Counter Value. The COUNT bits show the current count value of the GPT counter.

Chapter 29
I2C Controller (I2C)

29.1 Overview

This chapter describes block-level operation and programming of I2C. The chapter is intended for a block-driver software developer. To understand how the block is integrated at the SoC level, a system software developer should see discussions of the block in the appropriate SoC-level chapter(s).

References: This document assumes an understanding of the following document:

- *The I2C Bus Specification*, Version 2.1, by Philips Semiconductor

The Inter IC (I2C) provides functionality of a standard I2C slave and master. The I2C is designed to be compatible with the standard NXP I2C bus protocol.

NOTE

independent I2C channels are available.

I2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I2C standard allows additional devices to be connected to the bus for expansion and system development. See the connection diagram in the figure below.

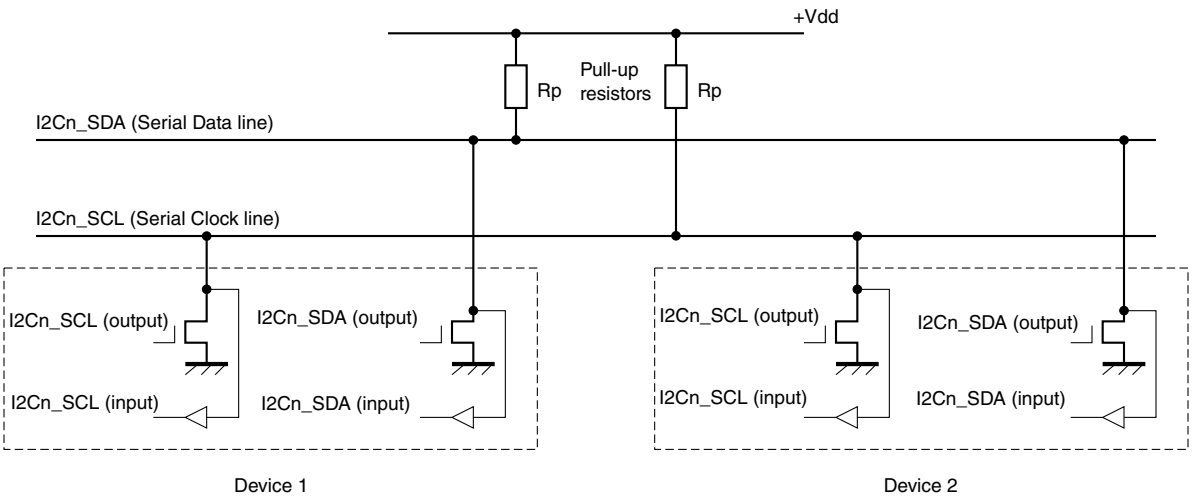


Figure 29-1. Connection of devices to I2C bus

The I2C interface speed is dependent on the I2C bus loading and timing characteristics. For pin requirement details, see *The I2C Bus Specification*. The I2C system is a true multimaster bus including arbitration and collision detection that prevents data corruption if multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer. The figure below shows the block diagram of I2C.

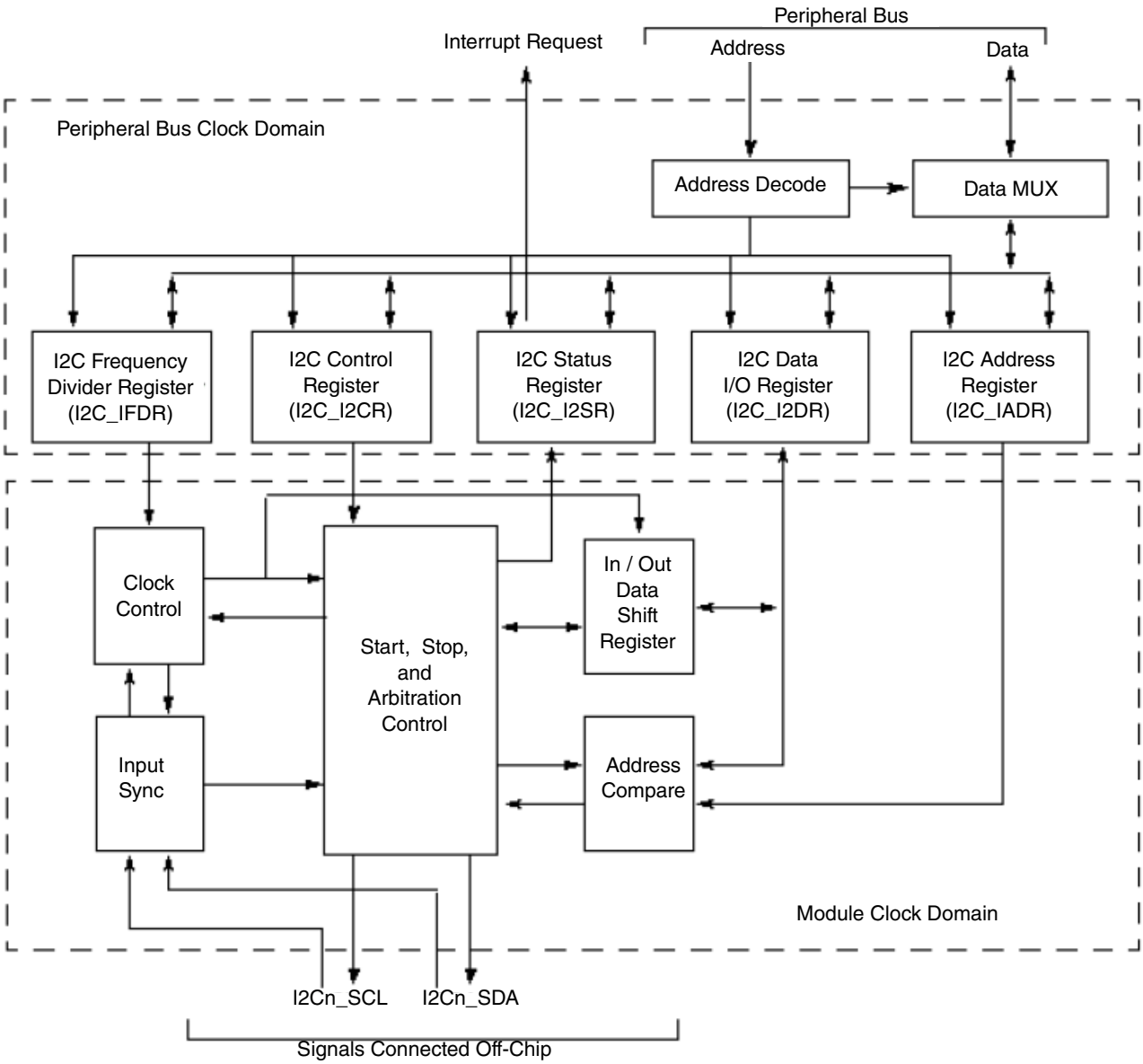


Figure 29-2. I2C block diagram

29.1.1 Features

The I2C has the following key features:

- Compatibility with I2C bus standard
- Multimaster operation
- Software programmability for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave

- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated Start signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

29.1.2 Modes and operations

The I2C operates primarily in two functional modes: Standard mode and Fast mode.

- In Standard mode, I2C supports the data transfer rates up to 100 kbits/s.
- In Fast mode, data transfer rates up to 400 kbits/s can be achieved. Per block operation, there is no special configuration required for Fast or Standard mode. It is the data transfer rate that distinguishes Standard and Fast mode.

29.2 External Signals

This section discusses I2C signals that connect off-chip.

For I2C compliance, all devices connected to the I2Cn_SCL and I2Cn_SDA signals must have open-drain or open-collector outputs. The logic AND function is implemented on both lines with external pull-up resistors.

Inputs of I2Cn_SCL and I2Cn_SDA also need to be manually enabled by setting the SION bit in the IOMUX after the corresponding PADs are selected as I2C function.

The table below describes all I2C signals that connect off-chip.

Table 29-1. I2C External Signals

Signal	Description	Pad	Mode	Direction
I2C1_SCL	Serial Clock	CSI_PIXCLK	ALT3	IO
		GPIO1_IO02	ALT0	
		UART4_TX_DATA	ALT2	
I2C1_SDA	Serial Data	CSI_MCLK	ALT3	IO
		GPIO1_IO03	ALT0	
		UART4_RX_DATA	ALT2	
I2C2_SCL	Serial Clock	CSI_HSYNC	ALT3	IO
		GPIO1_IO00	ALT0	
		UART5_TX_DATA	ALT2	
I2C2_SDA	Serial Data	CSI_VSYNC	ALT3	IO

Table continues on the next page...

Table 29-1. I2C External Signals (continued)

Signal	Description	Pad	Mode	Direction
		GPIO1_IO01	ALT0	
		UART5_RX_DATA	ALT2	
I2C3_SCL	Serial Clock	ENET2_RX_DATA0	ALT3	IO
		LCD_DATA01	ALT4	
		UART1_TX_DATA	ALT2	
I2C3_SDA	Serial Data	ENET2_RX_DATA1	ALT3	IO
		LCD_DATA00	ALT4	
		UART1_RX_DATA	ALT2	
I2C4_SCL	Serial Clock	ENET2_RX_EN	ALT3	IO
		LCD_DATA03	ALT4	
		UART2_TX_DATA	ALT2	
I2C4_SDA	Serial Data	ENET2_TX_DATA0	ALT3	IO
		LCD_DATA02	ALT4	
		UART2_RX_DATA	ALT2	

29.3 Clocks

There are two input clocks for I2C.

The following table describes the clock sources for I2C. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 29-2. I2C Clocks

Clock name	Clock Root	Description
ipg_clk_patref	perclk_clk_root	Module clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

- Peripheral clock: This clock is used for peripheral bus register read/writes.
- Module clock: This is the functional clock of the I2C. The serial bit clock frequency is derived from the module clock. The module clock and peripheral clocks are synchronous with each other. The minimum frequency of the module clock should be 12.8 MHz for Fast mode to achieve 400-kbps operation.

29.4 Functional description

This section provides a complete functional description of the block.

29.4.1 I2C system configuration

After a reset, the I2C defaults to Slave Receive operations. Thus, when not operating as a master or responding to a slave transmit address, the I2C defaults to the Slave Receive state.

For exceptions, see Initialization sequence.

NOTE

The I2C is designed to be compatible with the Philips™ I2C bus protocol. For information on system configuration, protocol, and restrictions, see the *I2C Bus Specification*, version 2.1, by Philips Semiconductors. The I2C supports Standard and Fast modes only.

29.4.2 Arbitration procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices, and the high period equals the shortest. A data arbitration procedure determines the relative priority of competing devices.

A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to Slave Receive mode and stops driving I2Cn_SDA. In this case, the transition from master to Slave mode does not generate a Stop condition. Meanwhile, hardware sets the arbitration lost bit in the I2C Status register (I2C_I2SR[IAL] to indicate loss of arbitration).

29.4.3 Clock synchronization

Because wire-AND logic is used, a high-to-low transition on SCL affects devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the Clock High state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period. Therefore, the device with the longest low period holds the synchronized clock SCL low.

Devices with shorter low periods enter a High Wait state during this time (see Figure 29-3). When all devices involved have counted off their low periods, the synchronized clock SCL is released and pulled high. There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

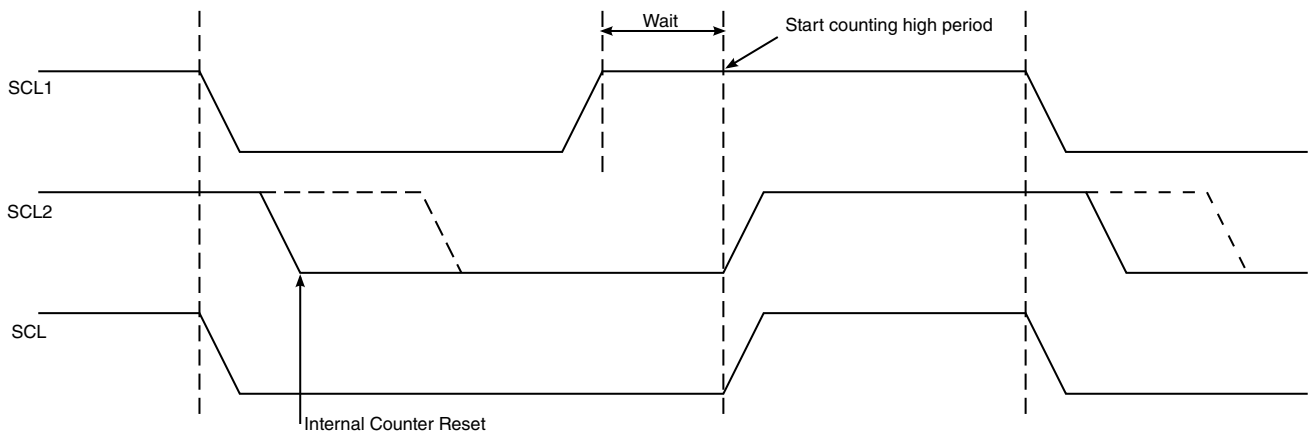


Figure 29-3. Synchronized clock SCL

29.4.4 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into a Wait state until the slave releases SCL.

29.4.5 Clock stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

29.4.6 Peripheral bus accesses

I2C is a 16-bit block. Only half-word accesses should be performed to the block.

29.4.7 Generation of transfer error on IP bus

If an address is received on the peripheral slave bus interface but it is not implemented, an access error is generated.

29.4.8 Reset

The I2C can be reset in the following ways:

- Global reset: A hard asynchronous reset of the whole I2C
- Software reset: An internal reset for the whole I2C (except for I2C_IADR and I2C_IFDR registers) initiated by deasserting the I2C_I2CR[IEN] bit

29.4.9 Interrupts

There is only one interrupt from the block, which is enabled by setting the I2C_I2CR[IIEN] bit.

The interrupt is generated in any one of the following conditions:

- One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock).
- An address is received that matches its own specific address in Slave Receive mode.
- Arbitration is lost.

29.4.10 Byte order

The block only supports the Little-Endian mode.

29.5 Initialization

NOTE

Ensure the input select pins for IOMUXC are configured correctly for I2C.

29.5.1 Initialization sequence

Before the interface can transfer serial data, registers must be initialized, as listed here.

1. Set the data sampling rate (I2C_IFDR[IC] to obtain SCL frequency from the system bus clock.
2. Update the address in the (I2C_IADR) to define its slave address (address can range from 0 to 0x7f).
3. Set the I2C enable bit (I2C_I2CR[IEN]) to enable the I2C bus interface system.
4. Modify the bits in the I2C_I2CR to select Master/Slave mode, Transmit/Receive mode, and Interrupt-Enable or not.

29.5.2 Generation of Start

After completion of the initialization procedure, serial data can be transmitted by selecting the Master Transmit mode. On a multimaster bus system, the busy bus (I2C_I2SR[IBB]) must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the Start signal and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the LSB indicates the transfer direction.

The free time between a Stop and the next Start condition is built into the hardware that generates the Start cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I2C is not busy after writing the calling address to the data register (I2C_I2DR), before proceeding to load data into the data register (I2C_I2DR).

29.5.3 Post-transfer software response

Sending or receiving a byte sets the data transferring bit (I2C_I2SR[ICF]), which indicates one byte of communication is finished. Upon completion, the interrupt status (I2C_I2SR[IIF]) is also set. An external interrupt is generated if the interrupt enable (I2C_I2CR[IIEN]) is set. The software must first clear the interrupt status (I2C_I2SR[IIF]) in the interrupt routine.

See the flow chart in Figure 29-5.

The data transferring bit (I2C_I2SR[ICF]) is cleared either by reading from I2C_I2DR in Receive mode or by writing to this register in Transmit mode.

The software can service the I2C I/O in the main program by monitoring the interrupt status (I2C_I2SR[IIF]) if the interrupt enable is deasserted. In this case, the interrupt status should be polled in the data transferring bit (I2C_I2SR[ICF]) because the operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in Transmit mode; that is, the address is sent. If Master Receive mode is required, then I2C_I2CR[MTX] should be toggled and a dummy read of the I2C_I2DR register must be executed to trigger receive data.

During Slave-mode address cycles (I2C_I2SR[IAAS] = 1), the slave read/write bit I2C_I2SR[SRW] is read to determine the direction of the next transfer. The transmit/receive bit (I2C_I2CR[MTX]) should also be programmed accordingly. For Slave-mode data cycles (IAAS = 0), SRW is invalid. MTX should be read to determine the current transfer direction.

29.5.4 Generation of Stop

A data transfer ends when the master signals a Stop, which can occur after all data is sent.

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting the transmit acknowledge bit (I2C_I2CR[TXAK]) before reading the next-to-last byte. Before the last byte is read, a Stop signal must be generated.

29.5.5 Generation of Repeated Start

After the data transfer, if the master still requires the bus, it can signal another Start followed by another slave address without signaling a Stop.

29.5.6 Slave mode

In the slave interrupt service routine (see Figure 29-5), the block addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the Transmit/Receive mode select bit (I2C_I2CR[MTX]) according to the I2C_I2SR[SRW]. Writing to the I2C_I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2C_I2DR for slave transmits, or read from I2C_I2DR in Slave Receive mode. A dummy read of I2C_I2DR in Slave Receive mode releases SCL, allowing the master to send data.

In the slave transmitter routine, the receive acknowledge bit (I2C_I2SR[RXAK]) must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which the software must switch it from Transmit to Receiver mode. Reading the data register (I2C_I2DR) then releases SCL so the master can generate a Stop signal.

29.5.7 Arbitration lost

If several devices try to engage the bus at the same time, one becomes master. Hardware immediately switches devices that lose arbitration to Slave Receive mode. Data output to 12Cn_SDA stops, but 12Cn_SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer if the arbitration is lost (I2C_I2SR[IAL] = 1), and the Slave mode is selected (I2C_I2CR[MSTA] = 0).

See the flow chart in Figure 29-5.

If a device that is not a master tries to transmit or do a Start, hardware inhibits the transmission, clears MSTA without signaling a Stop, generates an interrupt to the ARM platform, and sets I2C_I2SR[IAL] to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test I2C_I2SR[IAL], and the software should clear it if it is set.

For Multimaster mode, when an I2C is enabled when the bus is busy and asserts Start, the I2C_I2SR[IAL] bit gets set only for 12Cn_SDA=0, 12Cn_SCL=0/1, 12Cn_SDA=1, and 12Cn_SCL=0; but not for 12Cn_SDA=1 and I2Cn_SCL=1, which is the equivalent of Bus Idle state.

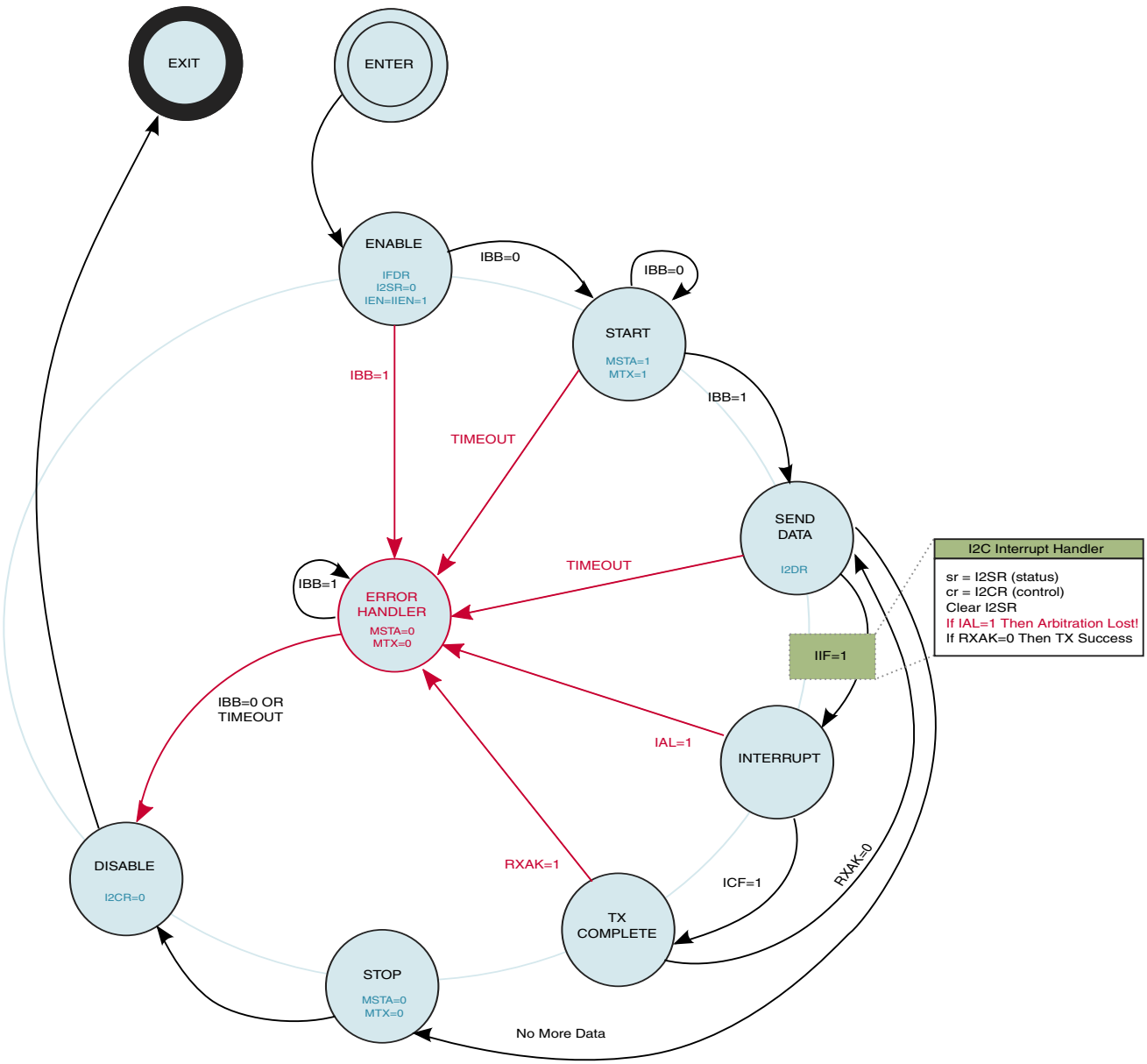


Figure 29-4. I2C Programming state diagram

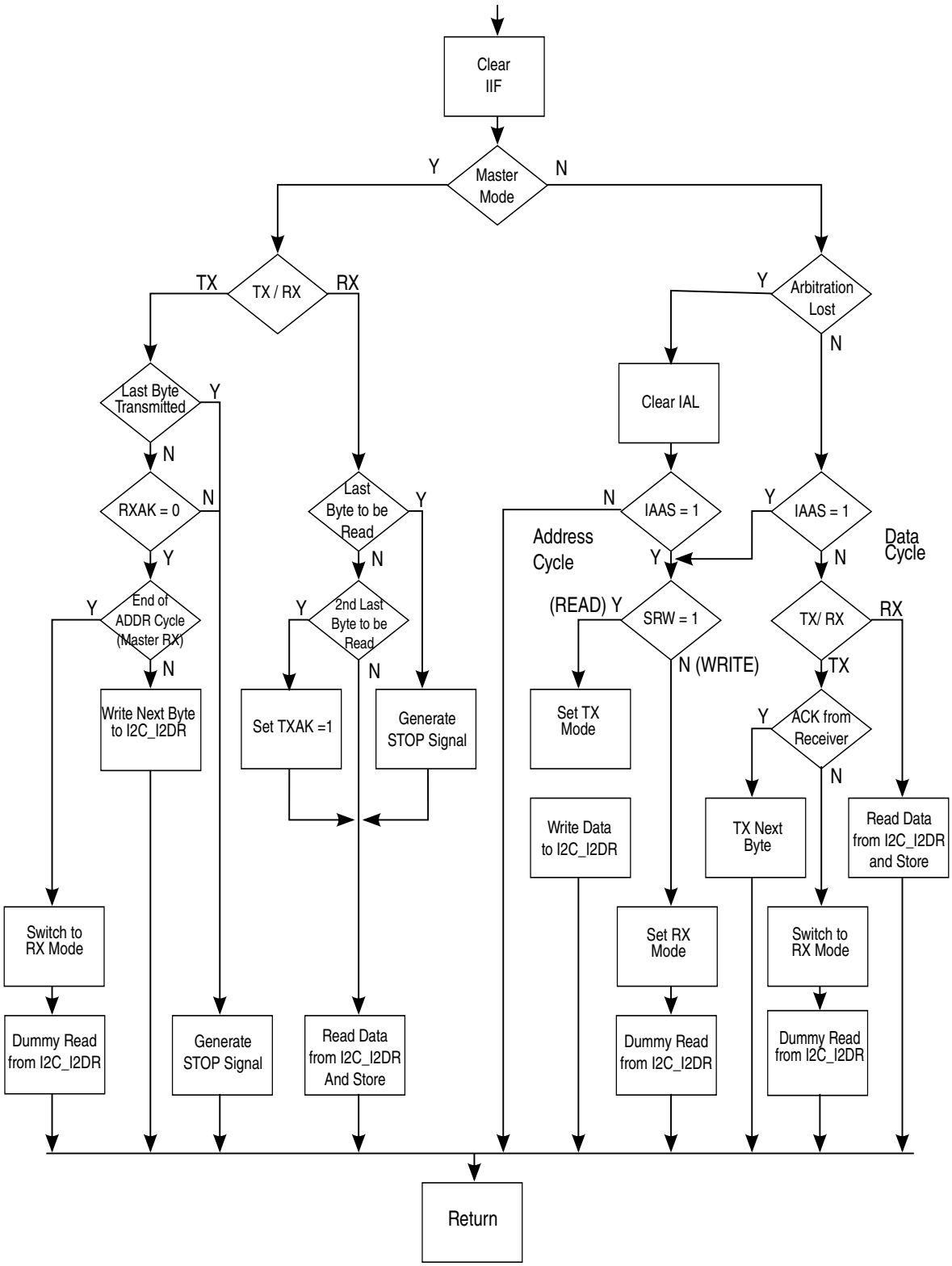


Figure 29-5. Flowchart of typical I2C interrupt routine

NOTE

For a Repeated Start only, the Stop-generation stage does not occur in Master mode. A loop repeats itself without stopping for the next start.

For Master Receive mode, I2C is programmed as Master Transmit during Address mode and after slave address transfer; the MTX bit should be cleared and a dummy read on the I2C_I2DR register should be performed so I2C can read the next receive data.

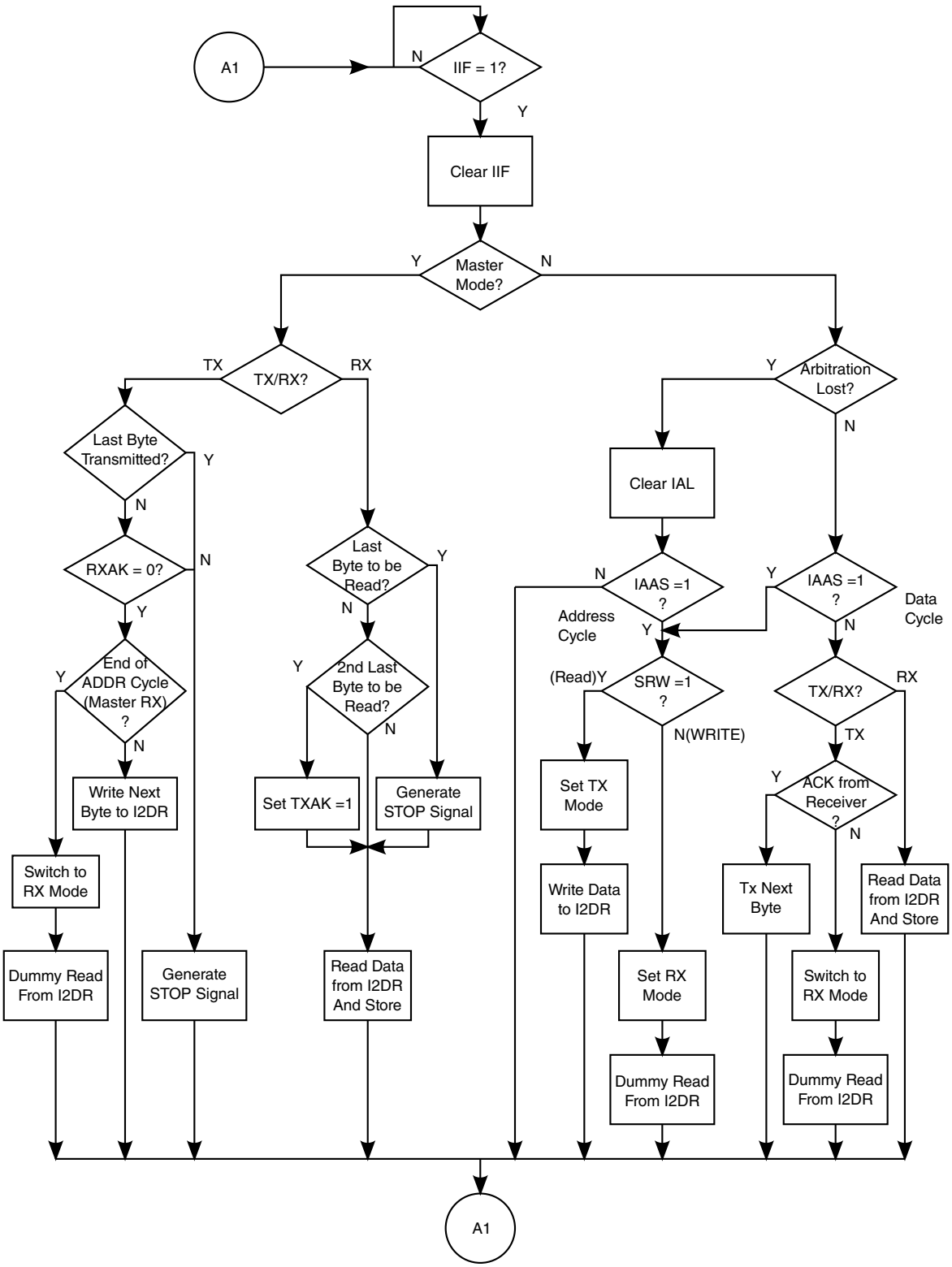


Figure 29-6. Flowchart for typical I2C polling routine

NOTE

The timeout value depends on the bus frequency at which I2C is operating. The minimum timeout for polling the IIF bit at a maximum I2C bus frequency of 400 kHz is T_{min} = 25 µs (=2.5 x 10 µs). This value can be calculated for any bus frequency. The formula is $T_{min} = 10/F_{SCL}$, where F_{SCL} is the frequency of the I2C clock (SCL).

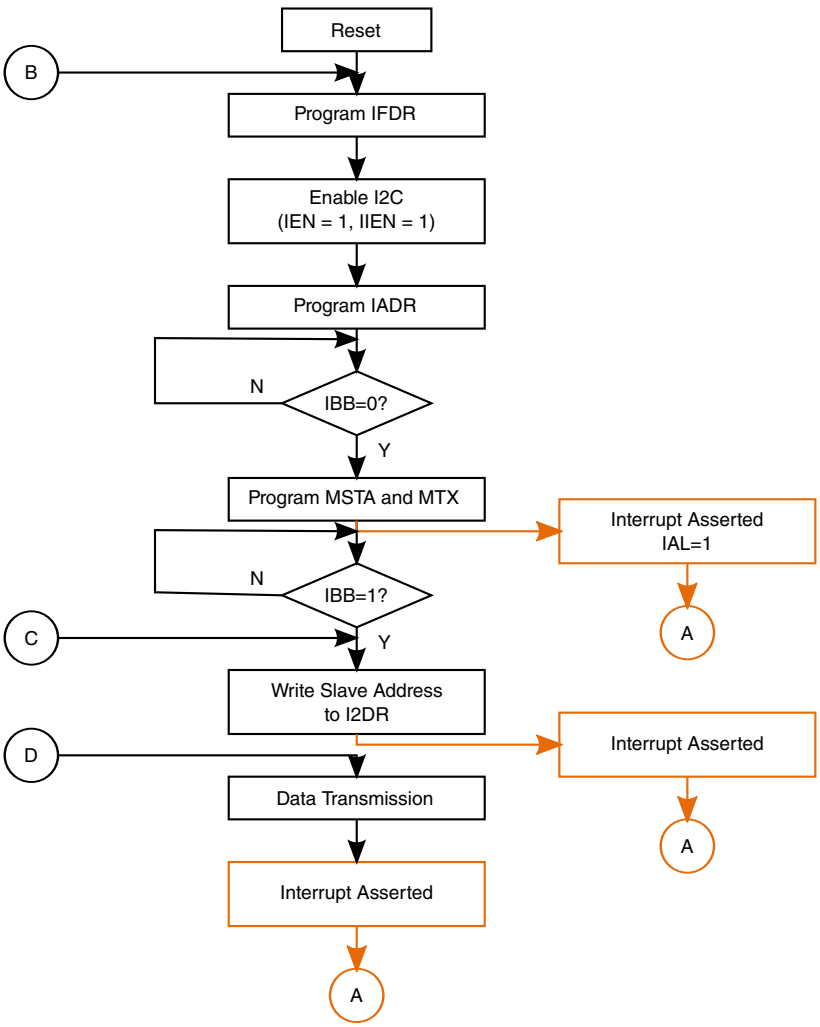


Figure 29-7. Detailed flowchart of a typical I2C Master Transmit mode, part 1

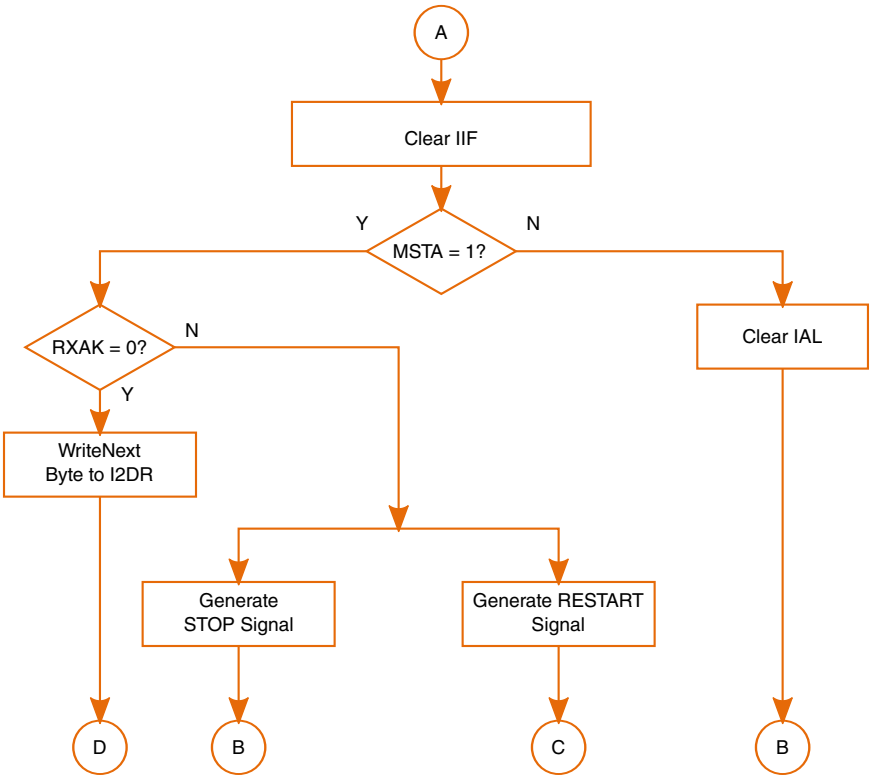


Figure 29-8. Detailed flowchart of a typical I2C Master Transmit mode, part 2

Figure 29-7 and Figure 29-8 show the Master Transmit mode operation with interrupt subroutine. If an interrupt is generated and the MSTA bit is 0, then bus arbitration is lost and IAL is set. Software can clear the IAL bit and reprogram I2C. If the MSTA bit is 1, then it is a transfer-generated interrupt. In this case, software can check the RXAK bit for a data receive acknowledgement by the slave and, accordingly, decide to do one of the following:

- Generate a STOP
- Generate a REPEATED START by writing to the I2C_I2CR register
- Perform the next data transfer by writing to the I2C_I2DR register

NOTE

The IBB bit is asserted by a Start condition on the bus, and it is deasserted by a Stop condition on the bus. Therefore, if arbitration is lost due to an unexpected Stop condition during transfer, then IBB is cleared. If arbitration is lost due to a data mismatch, then it is not cleared. Software should always clear the IEN bit and then set it if arbitration is lost.

29.6 Software restriction

Software should ensure that there is a delay of at least two module clock cycles after it sets the I2C_I2CR[RSTA] bit and before writing to the I2C_I2DR register. The maximum possible clock period of the module clock is 78 ns.

29.7 I2C Memory Map/Register Definition

The I2C contains five 16-bit registers.

NOTE

Registers at offsets 0x0002, 0x0006, 0x000A, and 0x000E are reserved for future additions.

I2C memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21A_0000	I2C Address Register (I2C1_IADR)	16	R/W	0000h	29.7.1/1259
21A_0004	I2C Frequency Divider Register (I2C1_IFDR)	16	R/W	0000h	29.7.2/1259
21A_0008	I2C Control Register (I2C1_I2CR)	16	R/W	0000h	29.7.3/1261
21A_000C	I2C Status Register (I2C1_I2SR)	16	R/W	0081h	29.7.4/1262
21A_0010	I2C Data I/O Register (I2C1_I2DR)	16	R/W	0000h	29.7.5/1264
21A_4000	I2C Address Register (I2C2_IADR)	16	R/W	0000h	29.7.1/1259
21A_4004	I2C Frequency Divider Register (I2C2_IFDR)	16	R/W	0000h	29.7.2/1259
21A_4008	I2C Control Register (I2C2_I2CR)	16	R/W	0000h	29.7.3/1261
21A_400C	I2C Status Register (I2C2_I2SR)	16	R/W	0081h	29.7.4/1262
21A_4010	I2C Data I/O Register (I2C2_I2DR)	16	R/W	0000h	29.7.5/1264
21A_8000	I2C Address Register (I2C3_IADR)	16	R/W	0000h	29.7.1/1259
21A_8004	I2C Frequency Divider Register (I2C3_IFDR)	16	R/W	0000h	29.7.2/1259
21A_8008	I2C Control Register (I2C3_I2CR)	16	R/W	0000h	29.7.3/1261
21A_800C	I2C Status Register (I2C3_I2SR)	16	R/W	0081h	29.7.4/1262
21A_8010	I2C Data I/O Register (I2C3_I2DR)	16	R/W	0000h	29.7.5/1264
21F_8000	I2C Address Register (I2C4_IADR)	16	R/W	0000h	29.7.1/1259
21F_8004	I2C Frequency Divider Register (I2C4_IFDR)	16	R/W	0000h	29.7.2/1259
21F_8008	I2C Control Register (I2C4_I2CR)	16	R/W	0000h	29.7.3/1261
21F_800C	I2C Status Register (I2C4_I2SR)	16	R/W	0081h	29.7.4/1262
21F_8010	I2C Data I/O Register (I2C4_I2DR)	16	R/W	0000h	29.7.5/1264

29.7.1 I2C Address Register (I2Cx_IADR)

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								ADR							0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_IADR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
7–1 ADR	Slave address. Contains the specific slave address to be used by the I2C. Slave mode is the default I2C mode for an address match on the bus. **NOTE:** The I2C_IADR holds the address to which the I2C responds when addressed as a slave. The slave address is not the address sent on the bus during the address transfer. The register is not reset by a software reset.
0 Reserved	This read-only field is reserved and always has the value 0.

29.7.2 I2C Frequency Divider Register (I2Cx_IFDR)

The I2C_IFDR provides a programmable prescaler to configure the clock for bit-rate selection. The register does not get reset by a software reset.

I2C clock is sourced from PERCLK_ROOT which is routed from IPG_CLK_ROOT. I2C clock frequency can easily obtained by using the following formula:

I2C clock Frequency = (PERCLK_ROOT frequency)/(division factor corresponding to IFDR)

By default, IPG_CLK_ROOT and PERCLK_ROOT frequencies are set to 49.5 MHz, where the root clock is sourced from PLL2's PFD2. Obtaining the frequencies can be accomplished by:

PLL2 = 528 MHz

PLL2_PFD2 = 528 MHz * 18 / 24 = 396 MHz

IPG_CLK_ROOT = (PLL2_PFD2 / ahb_podf)/ ipg_podf = (396 MHz/4)/2 = 49.5 MHz

PER_CLK_ROOT = IPG_CLK_ROOT/perclk_podf = 49.5 MHz/1 = 49.5 MHz

NOTE

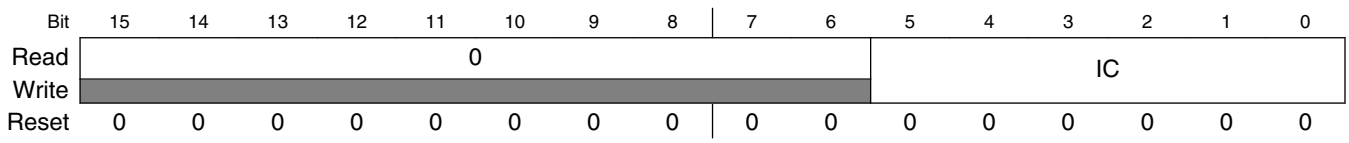
The above calculation assumes that the default CCM register settings, routing, and division factors are used. If different routing, PFD values, and/or division factors are used, the user must adjust the parameters accordingly to calculate the correct clock frequency.

The following table describes the divider and register values for the register field "IC."

Table 29-3. I2C_IFDR Register Field Values

IC	Divider		IC	Divider		IC	Divider		IC	Divider
0x00	30		0x10	288		0x20	22		0x30	160
0x01	32		0x11	320		0x21	24		0x31	192
0x02	36		0x12	384		0x22	26		0x32	224
0x03	42		0x13	480		0x23	28		0x33	256
0x04	48		0x14	576		0x24	32		0x34	320
0x05	52		0x15	640		0x25	36		0x35	384
0x06	60		0x16	768		0x26	40		0x36	448
0x07	72		0x17	960		0x27	44		0x37	512
0x08	80		0x18	1152		0x28	48		0x38	640
0x09	88		0x19	1280		0x29	56		0x39	768
0x0A	104		0x1A	1536		0x2A	64		0x3A	896
0x0B	128		0x1B	1920		0x2B	72		0x3B	1024
0x0C	144		0x1C	2304		0x2C	80		0x3C	1280
0x0D	160		0x1D	2560		0x2D	96		0x3D	1536
0x0E	192		0x1E	3072		0x2E	112		0x3E	1792
0x0F	240		0x1F	3840		0x2F	128		0x3F	2048

Address: Base address + 4h offset



I2Cx_IFDR field descriptions

Field	Description
15–6 Reserved	This read-only field is reserved and always has the value 0.
IC	I2C clock rate. Prescales the clock for bit-rate selection. Due to potentially slow I2Cn_SCL and I2Cn_SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency may be lower than IPG_CLK_ROOT divided by the divider shown in the I2C Data I/O Register. **NOTE:** The IC value should not be changed during the data transfer, however, it can be changed before a Repeat Start or Start programming sequence in I2C. The I2C protocol supports bit rates of up to 400 kbps. The IC bits need to be programmed in accordance with this constraint.

29.7.3 I2C Control Register (I2Cx_I2CR)

The I2C_I2CR is used to enable the I2C and the I2C interrupt. It also contains bits that govern operation as a slave or a master.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	IEN	IIEN	MSTA	MTX	TXAK	0	0	
Write						RSTA		
Reset	0	0	0	0	0	0	0	0

I2Cx_I2CR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 IEN	I2C enable. Also controls the software reset of the entire I2C. Resetting the bit generates an internal reset to the block. If the block is enabled in the middle of a byte transfer, Slave mode ignores the current bus transfer and starts operating when the next Start condition is detected. Master mode is not aware that the bus is busy, so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I2C to lose arbitration. Subsequently, bus operation returns to normal. 0 The block is disabled, but registers can still be accessed. 1 The I2C is enabled. This bit must be set before any other I2C_I2CR bits have an effect.
6 IIEN	I2C interrupt enable. **NOTE:** If data is written during the Start condition, that is, just after setting the I2C_I2CR[MSTA] and I2C_I2CR[MTX] bits, then the ICF bit is cleared at the falling edge of SCLK after Start. If data is written after the Start condition and falling edge of SCLK, then the ICF bit is cleared as soon as data is written. 0 I2C interrupts are disabled, but the status flag I2C_I2SR[IIF] continues to be set when an Interrupt condition occurs. 1 I2C interrupts are enabled. An I2C interrupt occurs if I2C_I2SR[IIF] is also set.
5 MSTA	Master/Slave mode select bit. If the master loses arbitration, MSTA is cleared without generating a Stop signal. **NOTE:** The module clock should be on for writing to the MSTA bit. **NOTE:** The MSTA bit is cleared by software to generate a Stop condition; it can also be cleared by hardware when the I2C loses the bus arbitration.

Table continues on the next page...

I2Cx_I2CR field descriptions (continued)

Field	Description
	0 Slave mode. Changing MSTA from 1 to 0 generates a Stop and selects Slave mode. 1 Master mode. Changing MSTA from 0 to 1 signals a Start on the bus and selects Master mode.
4 MTX	Transmit/Receive mode select bit. Selects the direction of master and slave transfers. 0 Receive. When a slave is addressed, the software should set MTX according to the slave read/write bit in the I2C status register (I2C_I2SR[SRW]). 1 Transmit. In Master mode, MTX should be set according to the type of transfer required. Therefore, for address cycles, MTX is always 1.
3 TXAK	Transmit acknowledge enable. Specifies the value driven onto I2Cn_SDA during acknowledge cycles for both master and slave receivers. **NOTE:** Writing TXAK applies only when the I2C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, the acknowledge bit = 1).
2 RSTA	Repeat start. Always reads as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a Repeated Start condition
Reserved	This read-only field is reserved and always has the value 0.

29.7.4 I2C Status Register (I2Cx_I2SR)

The I2C_I2SR contains bits that indicate transaction direction and status.

Address: Base address + Ch offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
Write								
Reset	1	0	0	0	0	0	0	1

I2Cx_I2SR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer is in progress. 1 Transfer is complete. This bit is set by the falling edge of the ninth clock of the last byte transfer.
6 IAAS	I2C addressed as a slave bit. The ARM platform is interrupted if the interrupt enable (I2C_I2CR[IIEN]) is set. The ARM platform must check the slave read/write bit (SRW) and set its Transfer/Receive mode accordingly. Writing to I2C_I2CR clears this bit. 0 Not addressed 1 Addressed as a slave. Set when its own address (I2C_IADR) matches the calling address.
5 IBB	I2C bus busy bit. Indicates the status of the bus. **NOTE:** When I2C is enabled (I2C_I2CR[IEN] = 1), it continuously polls the bus data (SDA) and clock (SCL) signals to determine a Start or Stop condition. 0 Bus is idle. If a Stop signal is detected, IBB is cleared. 1 Bus is busy. When Start is detected, IBB is set.
4 IAL	Arbitration lost. Set by hardware in the following circumstances (IAL must be cleared by software by writing a "0" to it at the start of the interrupt service routine): • I2Cn_SDA input samples low when the master drives high during an address or data-transmit cycle. • I2Cn_SDA input samples low when the master drives high during the acknowledge bit of a data-receive cycle. For the above two cases, the bit is set at the falling edge of the ninth I2Cn_SCL clock during the ACK cycle. • A Start cycle is attempted when the bus is busy. • A Repeated Start cycle is requested in Slave mode. • A Stop condition is detected when the master did not request it. **NOTE:** Software cannot set the bit. 0 No arbitration lost. 1 Arbitration is lost.
3 Reserved	This read-only field is reserved and always has the value 0.
2 SRW	Slave read/write. When the I2C is addressed as a slave, IAAS is set, and the slave read/write bit (SRW) indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I2C is a slave and has an address match. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IIF	I2C interrupt. Must be cleared by the software by writing a "0" to it in the interrupt routine. **NOTE:** The software cannot set the bit. 0 No I2C interrupt pending. 1 An interrupt is pending.

Table continues on the next page...

I2Cx_I2SR field descriptions (continued)

Field	Description
	This causes a processor interrupt request (if the interrupt enable is asserted [IIEN = 1]). The interrupt is set when one of the following occurs: • One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock). • An address is received that matches its own specific address in Slave Receive mode. • Arbitration is lost.
0 RXAK	Received acknowledge. This is the value received from the I2Cn_SDA input for the acknowledge bit during a bus cycle. 0 An "acknowledge" signal was received after the completion of an 8-bit data transmission on the bus. 1 A "No acknowledge" signal was detected at the ninth clock.

29.7.5 I2C Data I/O Register (I2Cx_I2DR)

In Master Receive mode, reading the data register allows a read to occur and initiates the next byte to be received. In Slave mode, the same function is available after it is addressed.

Address: Base address + 10h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								DATA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_I2DR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
DATA	Data Byte. Holds the last data byte received or the next data byte to be transferred. Software writes the next data byte to be transmitted or reads the data byte received. **NOTE:** The core-written value in I2C_I2DR cannot be read back by the core. Only data written by the I2C bus side can be read.

Chapter 30
IOMUX Controller (IOMUXC)

30.1 Overview

The IOMUX Controller (IOMUXC), together with the IOMUX, enables the IC to share one pad to several functional blocks. This sharing is done by multiplexing the pad's input and output signals.

Every module requires a specific pad setting (such as pull up or keeper), and for each pad, there are up to 8 muxing options (called ALT modes). The pad settings parameters are controlled by the IOMUXC.

The IOMUX consists only of combinatorial logic combined from several basic IOMUX cells. Each basic IOMUX cell handles only one pad signal's muxing.

Figure 30-1 illustrates the IOMUX/IOMUXC connectivity in the system.

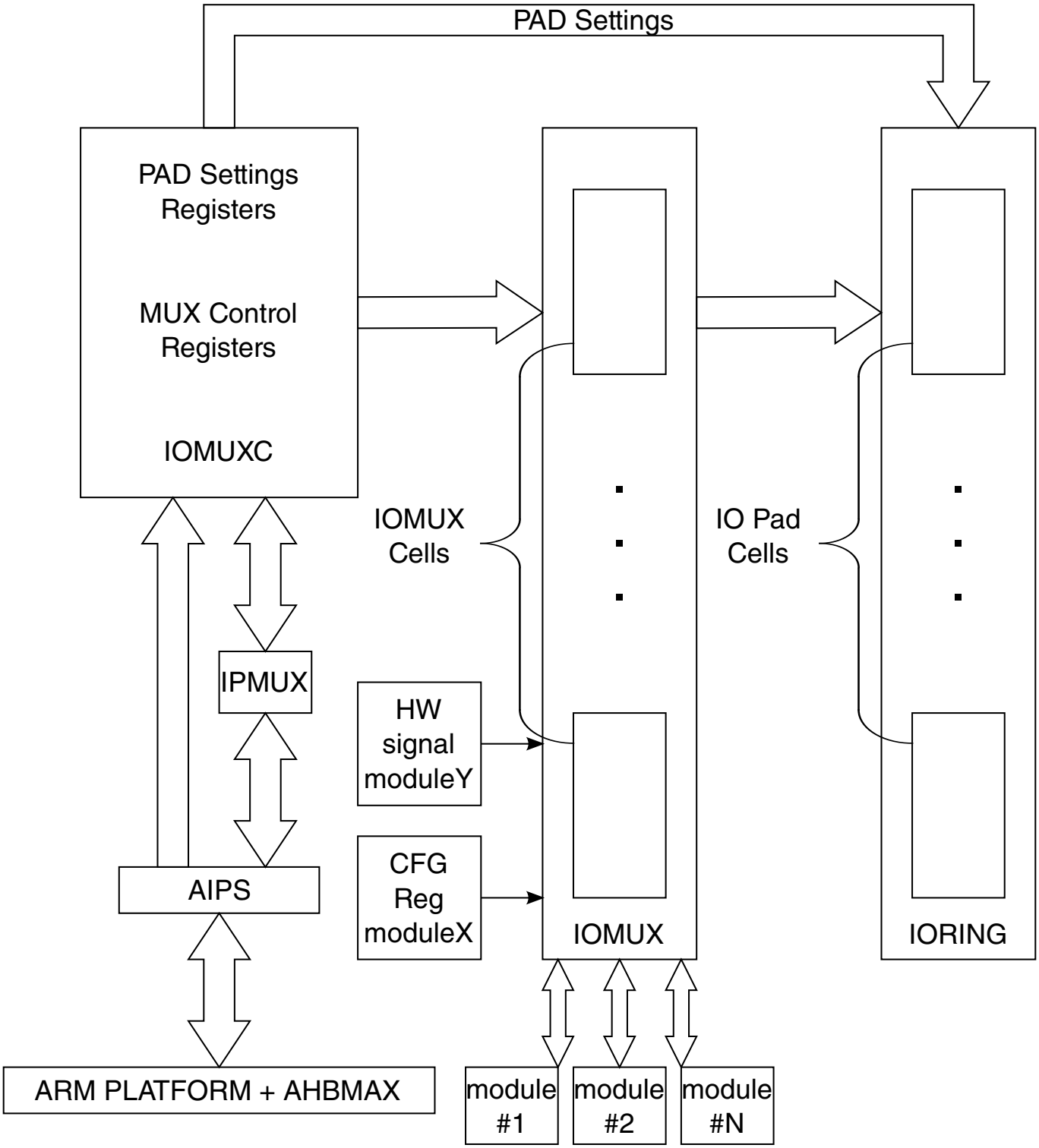


Figure 30-1. IOMUX SoC Level Block Diagram

30.1.1 Features

The IOMUXC features are:

- 32-bit software mux control registers (IOMUXC_SW_MUX_CTL_PAD_<PAD NAME> or IOMUXC_SW_MUX_CTL_GRP_<GROUP NAME>) to configure 1 of 8 alternate (ALT) MUX_MODE feilds of each pad or a predefined group of pads and to enable the forcing of an input path of the pad(s) (SION bit).
- 32-bit software pad control registers (IOMUXC_SW_PAD_CTL_PAD_<PAD_NAME> or IOMUXC_SW_PAD_CTL_GRP_<GROUP NAME>) to configure specific pad settings of each pad, or a predefined group of pads.
- 32-bit general purpose registers - 14 (GPR0 to GPR13) 32-bit registers according to SoC requirements for any usage.
- 32-bit input select control registers to control the input path to a module when more than one pad drives this module input.

Each SW MUX/PAD CTL IOMUXC register handles only one pad or one pad's group.

Only the minimum number of registers required by software are implemented by hardware. For example, if only ALT0 and ALT1 modes are used on Pad x then only one bit register will be generated as the MUX_MODE control field in the software mux control register of Pad x.

The software mux control registers may allow the forcing of pads to become input (input path enabled) regardless of the functional direction driven. This may be useful for loopback and GPIO data capture.

30.2 Clocks

The table found here describes the clock sources for IOMUXC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 30-1. IOMUXC Clocks

Clock name	Clock Root	Description
ipt_clk_io		IO clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

30.3 Functional description

This section provides a complete functional description of the block.

The IOMUXC consists of two sub-blocks:

- IOMUXC_REGISTERS includes all of the IOMUXC registers (see Features).
- IOMUXC_LOGIC includes all of the IOMUXC combinatorial logic (IP interface controls, address decoder, observability muxes).

The IOMUX consists of a number (about the number of pads in the SoC) of basic iomux_cell units. If only one functional mode is required for a specific pad, there is no need for IOMUX and the signals can be connected directly from the module to the I/O. The IOMUX cell is required whenever two or more functional modes are required for a specific pad or when one functional mode and the one test mode are required.

The basic iomux_cell design, which allows two levels of HW signal control (in ALT6 and ALT7 modes - ALT7 gets highest priority) is shown in Figure 30-2.

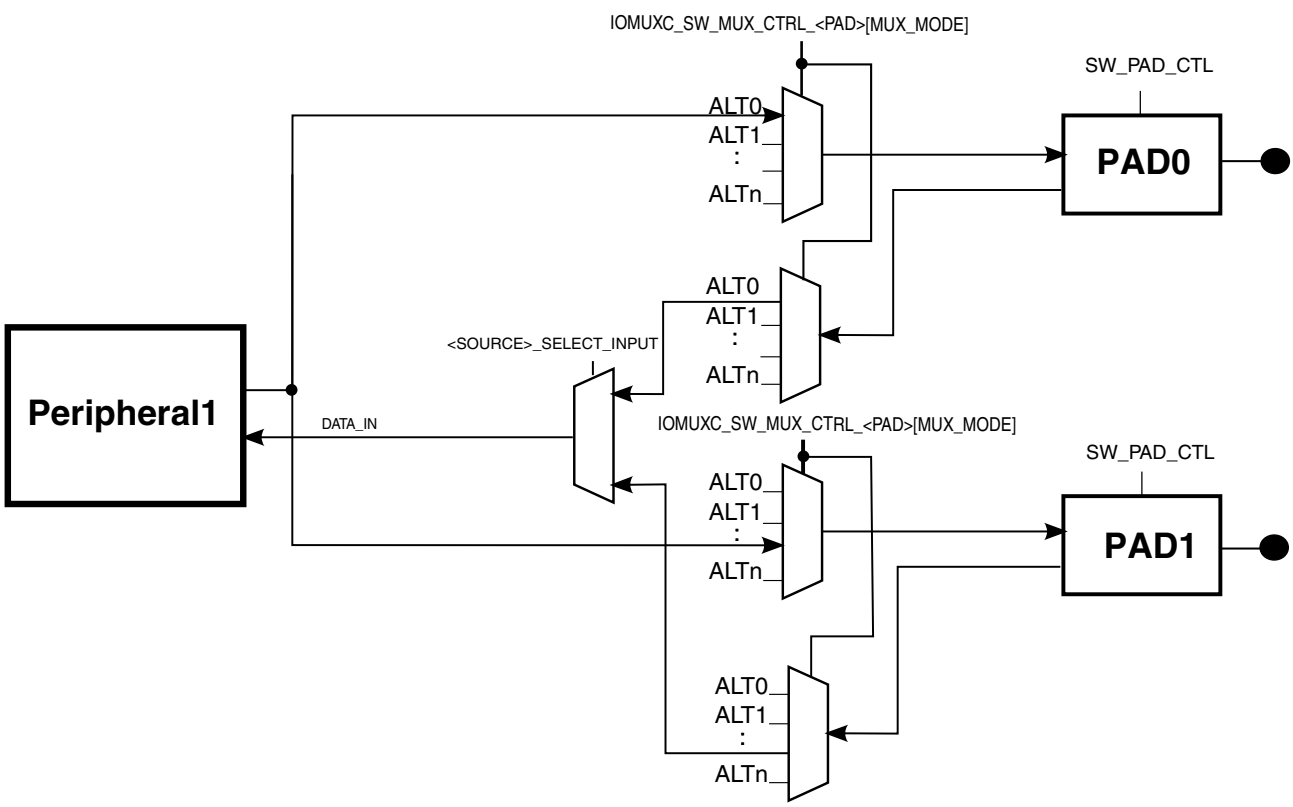


Figure 30-2. IOMUX Cell Block Diagram

30.3.1 ALT6 and ALT7 extended muxing modes

The ALT7 and ALT6 extended muxing modes allow any signal in the system (such as fuse, pad input, JTAG, or software register) to override any software configuration and to force the ALT6/ALT7 muxing mode.

It also allows an IOMUX software register to control a group of pads.

30.3.2 SW Loopback through SION bit

A limited option exists to override the default pad functionality and force the input path to be active (ipp_ibe==1'b1) regardless of the value driven by the corresponding module. This can be done by setting the SION (Software Input On) bit in the IOMUXC_SW_MUX_CTL register (when available) to "1".

Uses include:

- LoopBack - Module x drives the pad and also receives pad value as an input.
- GPIO Capture - Module x drives the pad and the value is captured by GPIO.

30.3.3 Daisy chain - multi pads driving same module input pin

In some cases, more than one pad may drive a single module input pin. Such cases require the addition of one more level of IOMUXing; all of these input signals are muxed, and a dedicated software controlled register controls the mux in order to select the required input path.

A module port involved in "daisy chain" requires two software configuration commands, one for selecting the mode for this pad (programable via the IOMUXC_SW_MUX_CTL_<PAD> registers) and one for defining it as the input path (via the daisy chain registers).

This means that a module port involved in "daisy chain" requires two software configuration commands, one for selecting the mode for this pad (programable via the IOMUXC_SW_MUX_CTL_<PAD> registers) and one for defining it as the input path (via the daisy chain registers). The daisy chain is illustrated in the figure below.

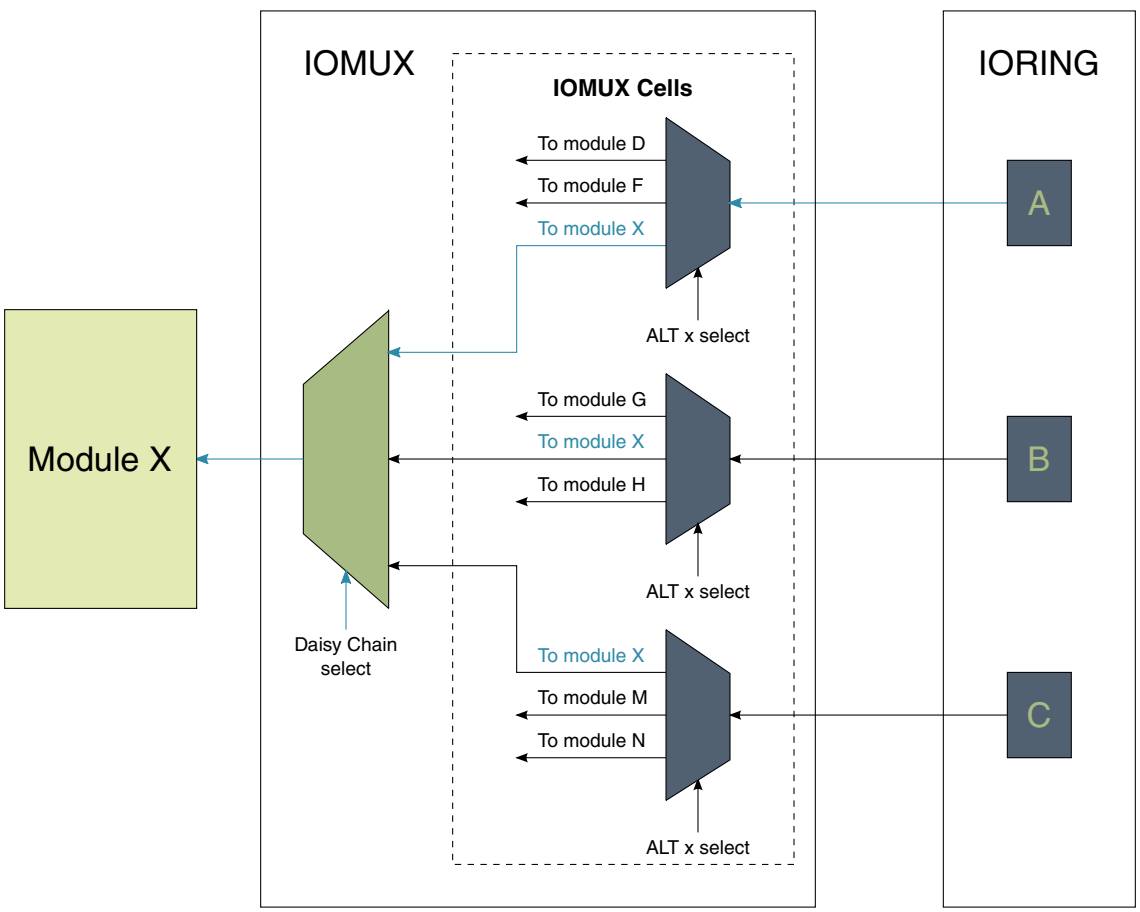


Figure 30-3. Daisy chain illustration

30.4 IOMUXC GPR Memory Map/Register Definition

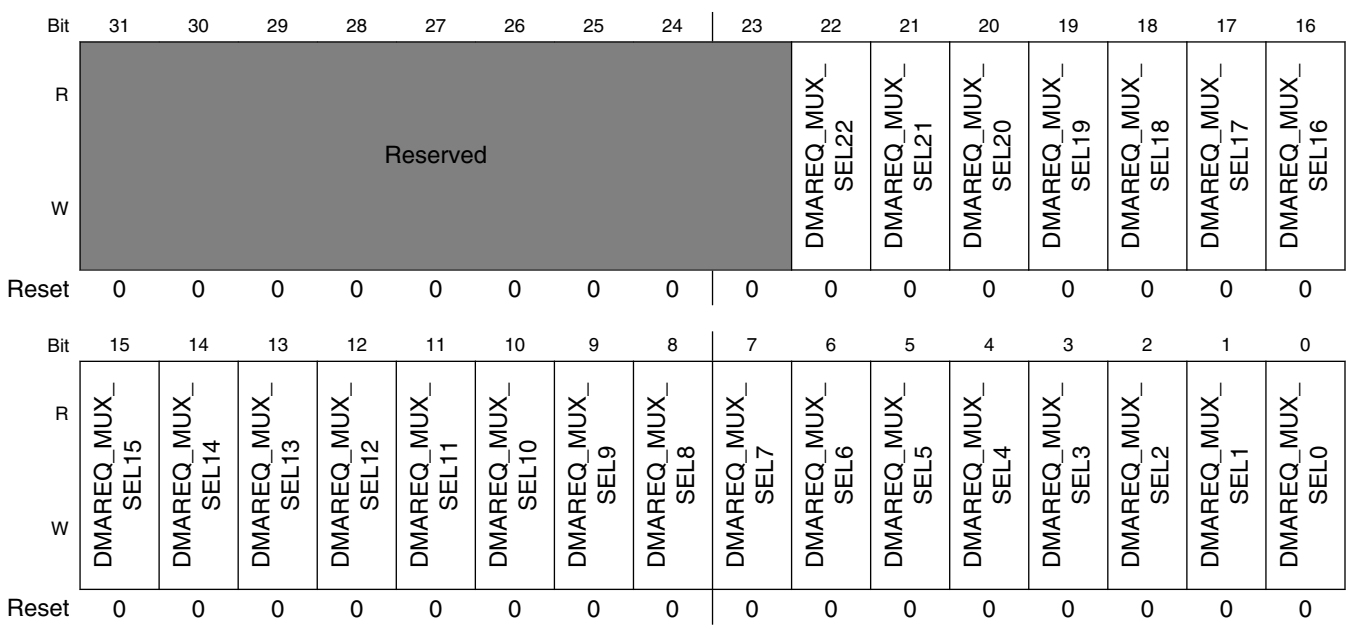
IOMUXC_GPR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_4000	GPR0 General Purpose Register (IOMUXC_GPR_GPR0)	32	R/W	0000_0000h	30.4.1/1271
20E_4004	GPR1 General Purpose Register (IOMUXC_GPR_GPR1)	32	R/W	0F40_0005h	30.4.2/1274
20E_4008	GPR2 General Purpose Register (IOMUXC_GPR_GPR2)	32	R/W	0000_0000h	30.4.3/1277
20E_400C	GPR3 General Purpose Register (IOMUXC_GPR_GPR3)	32	R/W	0000_0FFFh	30.4.4/1280
20E_4010	GPR4 General Purpose Register (IOMUXC_GPR_GPR4)	32	R/W	0000_0000h	30.4.5/1283
20E_4014	GPR5 General Purpose Register (IOMUXC_GPR_GPR5)	32	R/W	0000_0000h	30.4.6/1286
20E_4024	GPR9 General Purpose Register (IOMUXC_GPR_GPR9)	32	R/W	0000_0000h	30.4.7/1288
20E_4028	GPR10 General Purpose Register (IOMUXC_GPR_GPR10)	32	R/W	0000_0007h	30.4.8/1289
20E_4038	GPR14 General Purpose Register (IOMUXC_GPR_GPR14)	32	R/W	0000_0000h	30.4.9/1290

30.4.1 GPR0 General Purpose Register (IOMUXC_GPR_GPR0)

GPR Register

Address: 20E_4000h base + 0h offset = 20E_4000h



IOMUXC_GPR_GPR0 field descriptions

Field	Description
31–23 -	Reserved This field is reserved.

Table continues on the next page...

IOMUXC_GPR_GPR0 field descriptions (continued)

Field	Description
22 DMAREQ_MUX_SEL22	Selects between two possible sources for SDMA_EVENT[30]: 0 uart3.ipd_uart_tx_dmareq_b (default) 1 Reserved
21 DMAREQ_MUX_SEL21	Selects between two possible sources for SDMA_EVENT[29]: 0 uart3.ipd_uart_rx_dmareq_b (default) 1 Reserved
20 DMAREQ_MUX_SEL20	Selects between two possible sources for SDMA_EVENT[2]: 0 iomux_top.sdma_events[14] (default) 1 csi2.ipi_csi_int_b
19 DMAREQ_MUX_SEL19	Selects between two possible sources for SDMA_EVENT[47]: 0 uart6.ipd_uart_tx_dmareq_b (default) 1 Reserved
18 DMAREQ_MUX_SEL18	Selects between two possible sources for SDMA_EVENT[34]: 0 uart5.ipd_uart_tx_dmareq_b (default) 1 sai2.ipd_req_sai_tx
17 DMAREQ_MUX_SEL17	Selects between two possible sources for SDMA_EVENT[33]: 0 uart5.ipd_uart_rx_dmareq_b (default) 1 sai2.ipd_req_sai_rx
16 DMAREQ_MUX_SEL16	Selects between two possible sources for SDMA_EVENT[32]: 0 uart4.ipd_uart_tx_dmareq_b (default) 1 sai1.ipd_req_sai_tx
15 DMAREQ_MUX_SEL15	Selects between two possible sources for SDMA_EVENT[47]: 0 epit2.ipi_int_epit_oc 1 pxp.pxp_irq
14 DMAREQ_MUX_SEL14	Selects between two possible sources for SDMA_EVENT[0]: 0 ecspi3.ipd_req_cspi_rdma 1 i2c1.ipi_int
13 DMAREQ_MUX_SEL13	Selects between two possible sources for SDMA_EVENT[34]: 0 ecspi3.ipd_req_cspi_tdma 1 i2c2.ipi_int
12 DMAREQ_MUX_SEL12	Selects between two possible sources for SDMA_EVENT[33]: 0 ecspi4.ipd_req_cspi_rdma 1 i2c3.ipi_int
11 DMAREQ_MUX_SEL11	Selects between two possible sources for SDMA_EVENT[46]: 0 ecspi4.ipd_req_cspi_tdma 1 i2c4.ipi_int

Table continues on the next page...

IOMUXC_GPR_GPR0 field descriptions (continued)

Field	Description
10 DMAREQ_MUX_SEL10	Selects between two possible sources for SDMA_EVENT[45]: 0 epit1.ipi_int_epit_oc 1 csi.ipi_csi_int
9 DMAREQ_MUX_SEL9	Selects between two possible sources for SDMA_EVENT[44]: 0 gpt2.ipi_int_gpt 1 lcdif.lcdif_irq
8 DMAREQ_MUX_SEL8	Selects between two possible sources for SDMA_EVENT[43]: 0 adc2.ipd_req 1 tsc_dig.interrupt
7 DMAREQ_MUX_SEL7	Selects between two possible sources for SDMA_EVENT[13]: 0 enet1.ipd_req_mac0_timer[0] 1 uart7.ipd_uart_rx_dmareq
6 DMAREQ_MUX_SEL6	Selects between two possible sources for SDMA_EVENT[24]: 0 enet1.ipd_req_mac0_timer[1] 1 uart7.ipd_uart_tx_dmareq
5 DMAREQ_MUX_SEL5	Selects between two possible sources for SDMA_EVENT[16]: 0 enet2.ipd_req_mac0_timer[0] 1 uart8.ipd_uart_rx_dmareq
4 DMAREQ_MUX_SEL4	Selects between two possible sources for SDMA_EVENT[10]: 0 enet2.ipd_req_mac0_timer[1] 1 uart8.ipd_uart_tx_dmareq
3 DMAREQ_MUX_SEL3	Selects between two possible sources for SDMA_EVENT[9]: 0 sim1.ipd_sim_rx_dmareq 1 uart5.ipd_uart_rx_dmareq
2 DMAREQ_MUX_SEL2	Selects between two possible sources for SDMA_EVENT[8]: 0 sim1.ipd_sim_tx_dmareq 1 uart5.ipd_uart_tx_dmareq
1 DMAREQ_MUX_SEL1	Selects between two possible sources for SDMA_EVENT[7]: 0 sim2.ipd_sim_rx_dmareq 1 uart6.ipd_uart_rx_dmareq
0 DMAREQ_MUX_SEL0	Selects between two possible sources for SDMA_EVENT[2]: 0 sim2.ipd_sim_tx_dmareq 1 uart6.ipd_uart_tx_dmareq

30.4.2 GPR1 General Purpose Register (IOMUXC_GPR_GPR1)

GPR Register

Address: 20E_4000h base + 4h offset = 20E_4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved					ARMA7_CLK_AHB_EN	ARMA7_CLK_ATB_EN	ARMA7_CLK_APB_DBG_EN	TZASC1_BOOT_LOCK	EXC_MON	SAI3_MCLK_DIR	SAI2_MCLK_DIR	SAI1_MCLK_DIR	ENET2_TX_CLK_DIR	ENET1_TX_CLK_DIR	ADD_DS
W																
Reset	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	USB_EXP_MODE	ENET2_CLK_SEL	ENET1_CLK_SEL	GINT	ADDRS3[10]		ACT_CS3	ADDRS2[10]		ACT_CS2	ADDRS1[10]		ACT_CS1	ADDRS0[10]		ACT_CS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_GPR_GPR1 field descriptions

Field	Description
31–27 -	Reserved This field is reserved.
26 ARMA7_CLK_AHB_EN	ARM A7 platform AHB clock enable 0 AHB clock is not running (gated) 1 AHB clock is running (enabled)
25 ARMA7_CLK_ATB_EN	ARM A7 platform ATB clock enable 0 ATB clock is not running (gated) 1 ATB clock is running (enabled)
24 ARMA7_CLK_APB_DBG_EN	ARM A7 platform APB clock enable 0 APB clock is not running (gated) 1 APB clock is running (enabled)
23 TZASC1_BOOT_LOCK	TZASC-1 secure boot lock 0 secure boot lock is disabled 1 secure boot lock is enabled

Table continues on the next page...

IOMUXC_GPR_GPR1 field descriptions (continued)

Field	Description
22 EXC_MON	Exclusive monitor response select of illegal command 0 OKAY response 1 SLVError response (default)
21 SAI3_MCLK_DIR	LCD_CLK data direction control when sai3.MCLK is selected (ALT3) 0 LCD_CLK output driver is disabled when configured for ALT3 1 LCD_CLK output driver is enabled when configured for ALT3
20 SAI2_MCLK_DIR	SD1_CLK data direction control when sai2.MCLK is selected (ALT2) 0 SD1_CLK output driver is disabled when configured for ALT2 1 SD1_CLK output driver is enabled when configured for ALT2
19 SAI1_MCLK_DIR	LCD_DATA00 data direction control when sai1.MCLK is selected (ALT8) 0 LCD_DATA00 output driver is disabled when configured for ALT8 1 LCD_DATA00 output driver is enabled when configured for ALT8
18 ENET2_TX_ CLK_DIR	ENET2_TX_CLK data direction control when anatop. ENET_REF_CLK2 is selected (ALT1) 0 ENET2_TX_CLK output driver is disabled when configured for ALT1 1 ENET2_TX_CLK output driver is enabled when configured for ALT1
17 ENET1_TX_ CLK_DIR	ENET1_TX_CLK data direction control when anatop. ENET_REF_CLK1 is selected (ALT1) 0 ENET1_TX_CLK output driver is disabled when configured for ALT1 1 ENET1_TX_CLK output driver is enabled when configured for ALT1
16 ADD_DS	Setting ADD_DS to 0 will make the output driver of the SD3 pins ~10% stronger at highest drive strength (DSE=111). This is for use if the I/O buffer operation at WCS and 200 MHz is marginal. 0 output driver ~10% stronger 1 output driver is normal
15 USB_EXP_ MODE	USB Exposure mode 0 Exposure mode is disabled. 1 Exposure mode is enabled.
14 ENET2_CLK_ SEL	ENET2 reference clock mode select. 0 ENET2 TX reference clock driven by ref_enetpll. This clock is also output to pins via the IOMUX. ENET_REF_CLK2 function. 1 Gets ENET2 TX reference clk from the ENET2_TX_CLK pin. In this use case, an external OSC provides the clock for both the external PHY and the internal controller
13 ENET1_CLK_ SEL	ENET1 reference clock mode select. 0 ENET1 TX reference clock driven by ref_enetpll. This clock is also output to pins via the IOMUX. ENET_REF_CLK1 function. 1 Gets ENET1 TX reference clk from the ENET1_TX_CLK pin. In this use case, an external OSC provides the clock for both the external PHY and the internal controller
12 GINT	Global interrupt "0" bit (connected to ARM A7 IRQ#0 and GPC) 0 Global interrupt request is not asserted 1 Global interrupt request is asserted

Table continues on the next page...

IOMUXC_GPR_GPR1 field descriptions (continued)

Field	Description
11–10 ADDRS3[10]	Active Chip Select and Address Space. Each of the ACT_CSx represents one of the four chip selects of the EIM. When ACT_CSx=1'b1, the corresponding chip select is active and has a valid address space according to its address space configuration determined by ADDRSx[10] bits ADDRSx[10] is setting the space for each chip select which is active. The address space of the first active chip select must be the biggest one, the following active chip select address spaces may be equal or lower. Total address space size is 128 MByte. The supported configurations are: CS0(128M), CS1 (0M), CS2 (0M), CS3(0M) [default configuration] CS0(64M), CS1(64M), CS2(0M), CS3(0M) CS0(64M), CS1(32M), CS2(32M), CS3(0M) CS0(32M), CS1(32M), CS2(32M), CS3(32M) Address Space Configuration options (ADDRSx[10]): 00 32 MByte 01 64 MByte 10 128 MByte 11 Reserved
9 ACT_CS3	See description for ADDRS3[10]
8–7 ADDRS2[10]	See description for ADDRS3[10]
6 ACT_CS2	See description for ADDRS3[10]
5–4 ADDRS1[10]	See description for ADDRS3[10]
3 ACT_CS1	See description for ADDRS3[10]
2–1 ADDRS0[10]	See description for ADDRS3[10]
0 ACT_CS0	See description for ADDRS3[10]

30.4.3 GPR2 General Purpose Register (IOMUXC_GPR_GPR2)

GPR Register

Address: 20E_4000h base + 8h offset = 20E_4008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DRAM_CKE_BYPASS	DRAM_CKE1	DRAM_CKE0	DRAM_RESET	DRAM_RESET_BYPASS	MQS_OVERSAMPLE	MQS_EN	MQS_SW_RST	MQS_CLK_DIV							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	L2_MEM_LIGHTSLEEP	L2_MEM_DEEPSLEEP	L2_MEM_SHUTDOWN	L2_MEM_EN_POWERSAVING	LCDIF2_MEM_LIGHTSLEEP	LCDIF2_MEM_DEEPSLEEP	LCDIF2_MEM_SHUTDOWN	LCDIF2_MEM_EN_POWERSAVING	LCDIF1_MEM_LIGHTSLEEP	LCDIF1_MEM_DEEPSLEEP	LCDIF1_MEM_SHUTDOWN	LCDIF1_MEM_EN_POWERSAVING	PXP_MEM_LIGHTSLEEP	PXP_MEM_DEEPSLEEP	PXP_MEM_SHUTDOWN	PXP_MEM_EN_POWERSAVING
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR2 field descriptions

Field	Description
31 DRAM_CKE_BYPASS	DRAM CKE Bypass Select 0 DRAM CKE1, CKE0 driven by MMDC PHY Controller 1 DRAM CKE1, CKE0 driven by GPR2 register bits [30:29]
30 DRAM_CKE1	CKE1 Bypass Value 0 Drive CKE1 with 0 1 Drive CKE1 with 1
29 DRAM_CKE0	CKE0 Bypass Value 0 Drive CKE0 with 0 1 Drive CKE0 with 1
28 DRAM_RESET	DRAM Reset Value 0 Drive DRAM reset with 0 1 Drive DRAM reset with 1
27 DRAM_RESET_BYPASS	DRAM Reset Bypass Select

Table continues on the next page...

IOMUXC_GPR_GPR2 field descriptions (continued)

Field	Description
	0 DRAM reset driven by MMDC PHY Controller 1 DRAM reset driven by GPR2 register bit [28]
26 MQS_ OVERSAMPLE	Used to control the PWM oversampling rate compared with mclk. 0 32 1 64
25 MQS_EN	MQS enable. 0 Disable MQS 1 Enable MQS
24 MQS_SW_RST	MQS software reset. 0 Exit software reset for MQS 1 Enable software reset for MQS
23–16 MQS_CLK_DIV	Divider ratio control for mclk from hmclk. mclk frequency = 1/(n+1) * hmclk frequency. 00000000 mclk frequency = hmclk frequency 00000001 mclk frequency = 1/2 * hmclk frequency 00000010 mclk frequency = 1/3 * hmclk frequency 11111111 mclk frequency = 1/256 * hmclk frequency
15 L2_MEM_ LIGHTSLEEP	set to bring memory to light sleep state (Low leakage mode, maintain memory contents, no change to memory output)
14 L2_MEM_ DEEPSLEEP	control how memory enter Deep Sleep mode (shutdown periphery power, but maintain memory contents, outputs of memory are pulled low) 0 no force sleep control supported, memory deep sleep mode only entered when whole system in stop mode 1 force memory into deep sleep mode
13 L2_MEM_ SHUTDOWN	set to bring memory to shutdown state (most power saving state, Shut Down periphery and core, no memory retention)
12 L2_MEM_EN_ POWERSAVING	enable power saving features on L2 memory 0 none memory power saving features enabled, SHUTDOWN/DEEPSLEEP/LIGHTSLEEP will have no effect 1 memory power saving features enabled, set SHUTDOWN/DEEPSLEEP/LIGHTSLEEP(priority high to low) to enable power saving levels
11 LCDIF2_MEM_ LIGHTSLEEP	set to bring memory to light sleep state (Low leakage mode, maintain memory contents, no change to memory output)
10 LCDIF2_MEM_ DEEPSLEEP	control how memory enter Deep Sleep mode (shutdown periphery power, but maintain memory contents, outputs of memory are pulled low) 0 no force sleep control supported, memory deep sleep mode only entered when whole system in stop mode 1 force memory into deep sleep mode
9 LCDIF2_MEM_ SHUTDOWN	set to bring memory to shutdown state (most power saving state, Shut Down periphery and core, no memory retention)

Table continues on the next page...

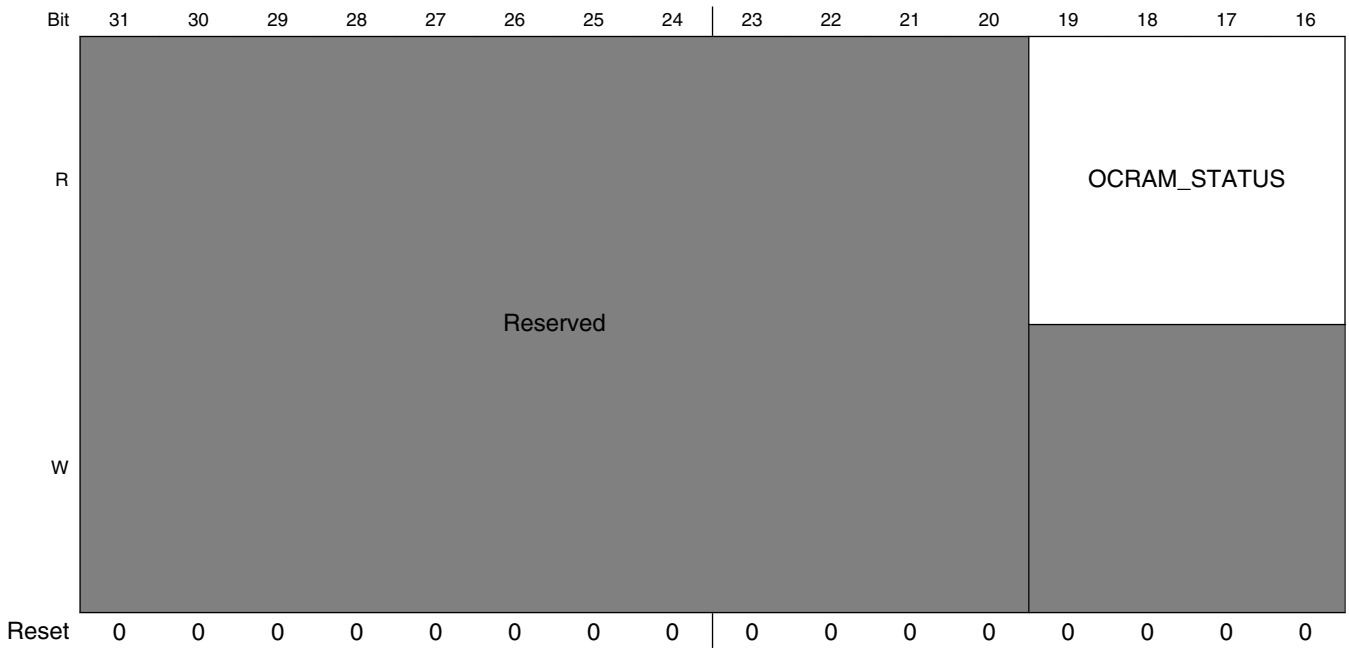
IOMUXC_GPR_GPR2 field descriptions (continued)

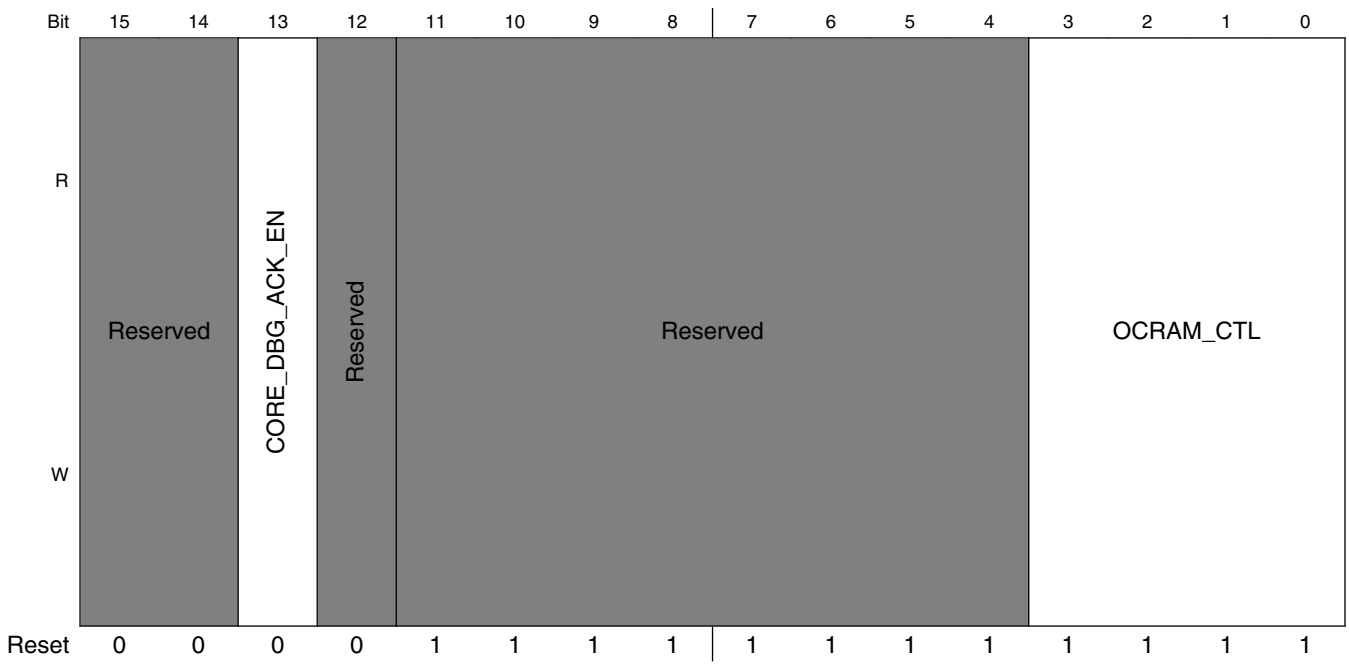
Field	Description
8 LCDIF2_MEM_EN_POWERSAVING	enable power saving features on LCDIF memory 0 none memory power saving features enabled, SHUTDOWN/DEEPSLEEP/LIGHTSLEEP will have no effect 1 memory power saving features enabled, set SHUTDOWN/DEEPSLEEP/LIGHTSLEEP(priority high to low) to enable power saving levels
7 LCDIF1_MEM_LIGHTSLEEP	set to bring memory to light sleep state (Low leakage mode, maintain memory contents, no change to memory output)
6 LCDIF1_MEM_DEEPSLEEP	control how memory enter Deep Sleep mode (shutdown periphery power, but maintain memory contents, outputs of memory are pulled low) 0 no force sleep control supported, memory deep sleep mode only entered when whole system in stop mode 1 force memory into deep sleep mode
5 LCDIF1_MEM_SHUTDOWN	set to bring memory to shutdown state (most power saving state, Shut Down periphery and core, no memory retention)
4 LCDIF1_MEM_EN_POWERSAVING	enable power saving features on LCDIF memory 0 none memory power saving features enabled, SHUTDOWN/DEEPSLEEP/LIGHTSLEEP will have no effect 1 memory power saving features enabled, set SHUTDOWN/DEEPSLEEP/LIGHTSLEEP(priority high to low) to enable power saving levels
3 PXP_MEM_LIGHTSLEEP	set to bring memory to light sleep state (Low leakage mode, maintain memory contents, no change to memory output)
2 PXP_MEM_DEEPSLEEP	control how memory enter Deep Sleep mode (shutdown periphery power, but maintain memory contents, outputs of memory are pulled low) 0 no force sleep control supported, memory deep sleep mode only entered when whole system in stop mode 1 force memory into deep sleep mode
1 PXP_MEM_SHUTDOWN	set to bring memory to shutdown state (most power saving state, Shut Down periphery and core, no memory retention)
0 PXP_MEM_EN_POWERSAVING	enable power saving features on PXP memory 0 none memory power saving features enabled, SHUTDOWN/DEEPSLEEP/LIGHTSLEEP will have no effect 1 memory power saving features enabled, set SHUTDOWN/DEEPSLEEP/LIGHTSLEEP(priority high to low) to enable power saving levels

30.4.4 GPR3 General Purpose Register (IOMUXC_GPR_GPR3)

GPR Register

Address: 20E_4000h base + Ch offset = 20E_400Ch





IOMUXC_GPR_GPR3 field descriptions

Field	Description
31–20 -	This field is reserved. Reserved
19–16 OCRAM_ STATUS	This field shows the OCRAM pipeline settings status, controlled by OCRAM_CTL[24:21] bits respectively. When the control bit is changed, the corresponding status bit goes high and keeps high until this new configuration is applied the internal logic. This provides a way for software to detect that the configuration has become valid. The suggested flow for changing the configuration in software is: • set/clear the control bit • poll the status bit until it goes to 0 OCRAM_STATUS[19] shows the write address pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM memory. OCRAM_STATUS[18] shows the write data pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM memory. OCRAM_STATUS[17] shows the read address pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM memory. OCRAM_STATUS[16] shows the read data pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM memory. 0 read data pipeline configuration valid 1 read data pipeline control bit changed
15–14 -	This field is reserved. Reserved
13 CORE_DBG_ ACK_EN	Mask control of Core debug acknowledge to global debug acknowledge 0 Core debug acknowledge is part of global acknowledge. 1 Core debug acknowledge is masked by this bit, and it is not part of global acknowledge.
12 -	This field is reserved. Reserved
11–4 -	This field is reserved. Reserved
OCRAM_CTL	OCRAM_CTL[3] write address pipeline control bit. When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data. When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle). 0 write address pipeline is disabled 1 write address pipeline is enabled OCRAM_CTL[2] - write data pipeline control bit When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data.

Table continues on the next page...

IOMUXC_GPR_GPR3 field descriptions (continued)

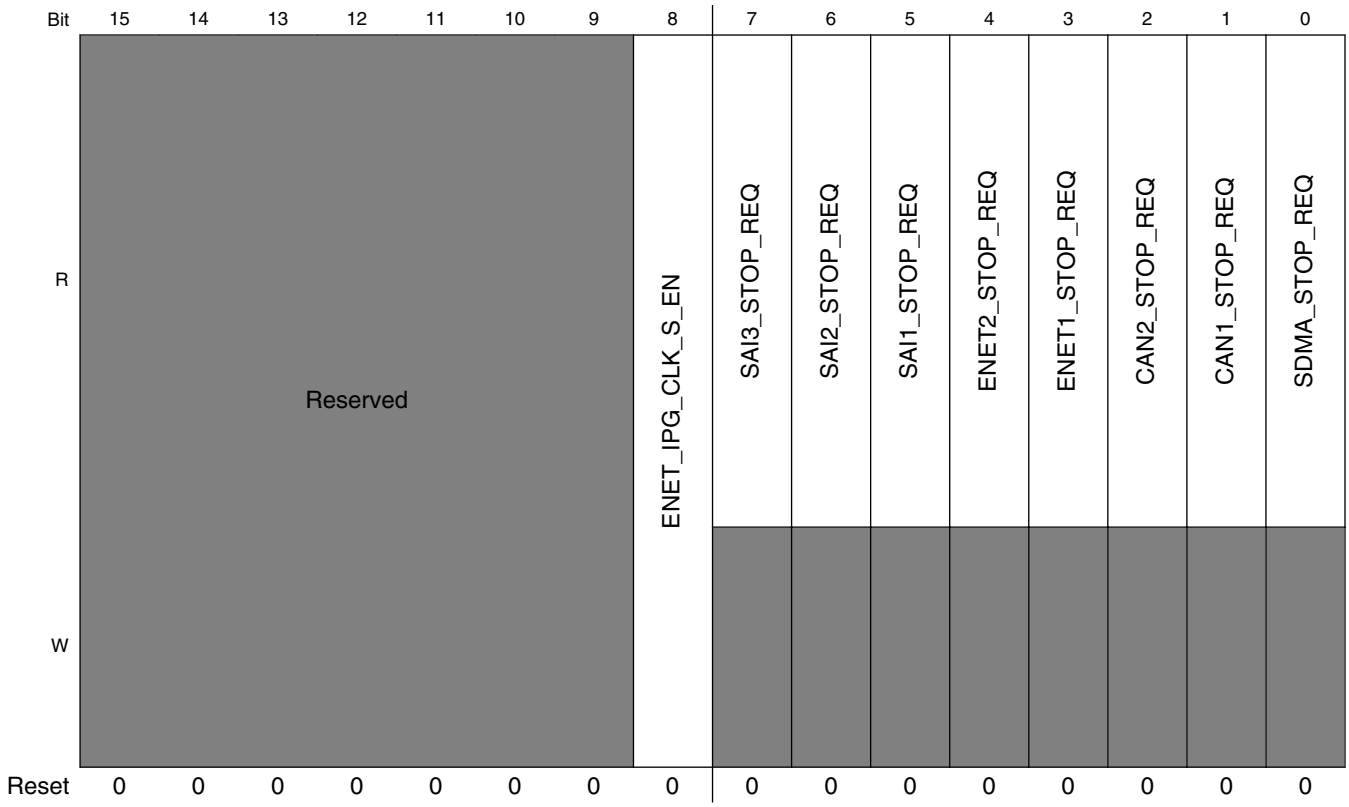
Field	Description
	When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle). 0 write data pipeline is disabled 1 write data pipeline is enabled OCRAM_CTL[1] read address pipeline control bit. When this feature is enabled, the read address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the read access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI read transaction, i.e., at most 1 more clock cycle for each read burst with multiple beats of data. When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait). 0 read address pipeline is disabled 1 read address pipeline is enabled OCRAM_CTL[0] - read data wait state control bit When thread data wait state is enabled, it will cost 2 cycles for each read access, (each beat of a read burst). This can avoid the potential timing problem caused by the relatively longer memory access time at higher frequency. When this feature is disabled, it only costs 1 clock cycle to finish a read transaction, i.e., get read data back in the next cycle of read request becomes valid on the bus. 0 read data pipeline is disabled 1 read data pipeline is enabled

30.4.5 GPR4 General Purpose Register (IOMUXC_GPR_GPR4)

GPR Register

Address: 20E_4000h base + 10h offset = 20E_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ARM_WFE	ARM_WFI	Reserved						SAI3_STOP_ACK	SAI2_STOP_ACK	SAI1_STOP_ACK	ENET2_STOP_ACK	ENET1_STOP_ACK	CAN2_STOP_ACK	CAN1_STOP_ACK	SDMA_STOP_ACK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



IOMUXC_GPR_GPR4 field descriptions

Field	Description
31 ARM_WFE	ARM A7 WFE event out indication on WFE state of the cores (these are status, read only bits) 0 ARM Core[GPR5-index - 4] is not in WFE mode 1 ARM Core[GPR5-index - 4] is in WFE mode
30 ARM_WFI	ARM A7 WFI event out indicating on WFI state of the cores (these are status, read only bits) 0 ARM Core[GPR5-index] is not in WFI mode 1 ARM Core[GPR5-index] is in WFI mode
29–24 -	This field is reserved. Reserved
23 SAI3_STOP_ ACK	SAI3 stop acknowledge. This is a status (read-only) bit 0 SAI3 stop acknowledge is not asserted 1 SAI3 stop acknowledge is asserted, SDMA is in STOP mode
22 SAI2_STOP_ ACK	SAI2 stop acknowledge. This is a status (read-only) bit 0 SAI2 stop acknowledge is not asserted 1 SAI2 stop acknowledge is asserted, SDMA is in STOP mode
21 SAI1_STOP_ ACK	SAI1 stop acknowledge. This is a status (read-only) bit 0 SAI1 stop acknowledge is not asserted 1 SAI1 stop acknowledge is asserted, SDMA is in STOP mode

Table continues on the next page...

IOMUXC_GPR_GPR4 field descriptions (continued)

Field	Description
20 ENET2_STOP_ACK	ENET2 stop acknowledge. This is a status (read-only) bit 0 ENET2 stop acknowledge is not asserted 1 ENET2 stop acknowledge is asserted, SDMA is in STOP mode
19 ENET1_STOP_ACK	ENET1 stop acknowledge. This is a status (read-only) bit 0 ENET1 stop acknowledge is not asserted 1 ENET1 stop acknowledge is asserted, SDMA is in STOP mode
18 CAN2_STOP_ACK	CAN2 stop acknowledge. This is a status (read-only) bit 0 CAN2 stop acknowledge is not asserted 1 CAN2 stop acknowledge is asserted, SDMA is in STOP mode
17 CAN1_STOP_ACK	CAN1 stop acknowledge. This is a status (read-only) bit 0 CAN1 stop acknowledge is not asserted 1 CAN1 stop acknowledge is asserted, SDMA is in STOP mode
16 SDMA_STOP_ACK	SDMA stop acknowledge. This is a status (read-only) bit 0 SDMA stop acknowledge is not asserted 1 SDMA stop acknowledge is asserted, SDMA is in STOP mode
15–9 -	This field is reserved. Reserved
8 ENET_IPG_CLK_S_EN	ENET ipg_clk_s clock gating enable 0 ipg_clk_s is gated when there's no IPS access 1 ipg_clk_s is always on
7 SAI3_STOP_REQ	SAI3 stop request. 0 stop request off 1 stop request on
6 SAI2_STOP_REQ	SAI2 stop request. 0 stop request off 1 stop request on
5 SAI1_STOP_REQ	SAI1 stop request. 0 stop request off 1 stop request on
4 ENET2_STOP_REQ	ENET2 stop request. 0 stop request off 1 stop request on
3 ENET1_STOP_REQ	ENET1 stop request. 0 stop request off 1 stop request on
2 CAN2_STOP_REQ	CAN2 stop request.

Table continues on the next page...

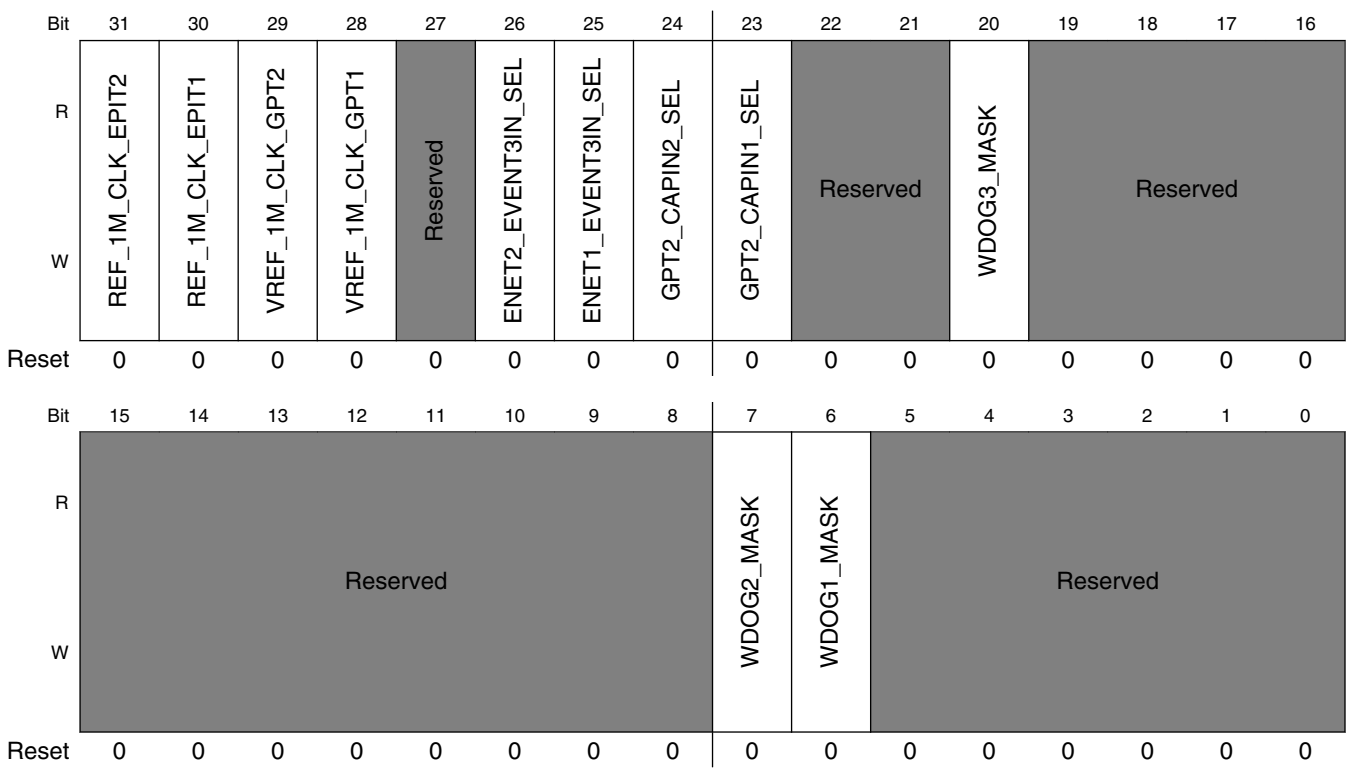
IOMUXC_GPR_GPR4 field descriptions (continued)

Field	Description
	0 stop request off 1 stop request on
1 CAN1_STOP_REQ	CAN1 stop request. 0 stop request off 1 stop request on
0 SDMA_STOP_REQ	SDMA stop request. 0 stop request off 1 stop request on

30.4.6 GPR5 General Purpose Register (IOMUXC_GPR_GPR5)

GPR Register

Address: 20E_4000h base + 14h offset = 20E_4014h



IOMUXC_GPR_GPR5 field descriptions

Field	Description
31 REF_1M_CLK_EPIT2	EPIT2 1 MHz clock source select

Table continues on the next page...

IOMUXC_GPR_GPR5 field descriptions (continued)

Field	Description
	0 EPIT2 ipg_clk_highfreq driven by IPG_PERCLK 1 EPIT2 ipg_clk_highfreq driven by anatop 1 MHz clock
30 REF_1M_CLK_EPIT1	EPIT1 1 MHz clock source select 0 EPIT1 ipg_clk_highfreq driven by IPG_PERCLK 1 EPIT1 ipg_clk highfreq driven by anatop 1 MHz clock
29 VREF_1M_CLK_GPT2	GPT2 1 MHz clock source select 0 GPT2 ipg_clk_highfreq driven by IPG_PERCLK 1 GPT2 ipg_clk_highfreq driven by anatop 1 MHz clock
28 VREF_1M_CLK_GPT1	GPT1 1 MHz clock source select 0 GPT1 ipg_clk_highfreq driven by IPG_PERCLK 1 GPT1 ipg_clk_highfreq driven by anatop 1 MHz clock
27 -	This field is reserved. Reserved
26 ENET2_EVENT3IN_SEL	ENET2 input timer event3 source select 0 event3 source input from pad 1 event3 source input from gpt2.ipp_do_cmpout2
25 ENET1_EVENT3IN_SEL	ENET1 input timer event3 source select 0 event3 source input from pad 1 event3 source input from gpt2.ipp_do_cmpout1
24 GPT2_CAPIN2_SEL	GPT2 input capture channel 2 source select 0 source from pad 1 source from enet2.ipp_do_mac0_timer[3]
23 GPT2_CAPIN1_SEL	GPT2 input capture channel 1 source select 0 source from pad 1 source from enet1.ipp_do_mac0_timer[3]
22–21 -	This field is reserved. Reserved
20 WDOG3_MASK	WDOG3 Timeout Mask 0 WDOG3 Timeout behaves normally 1 WDOG3 Timeout is masked
19–8 -	This field is reserved. Reserved
7 WDOG2_MASK	WDOG2 Timeout Mask 0 WDOG2 Timeout behaves normally 1 WDOG2 Timeout is masked
6 WDOG1_MASK	WDOG1 Timeout Mask 0 WDOG1 Timeout behaves normally 1 WDOG1 Timeout is masked

Table continues on the next page...

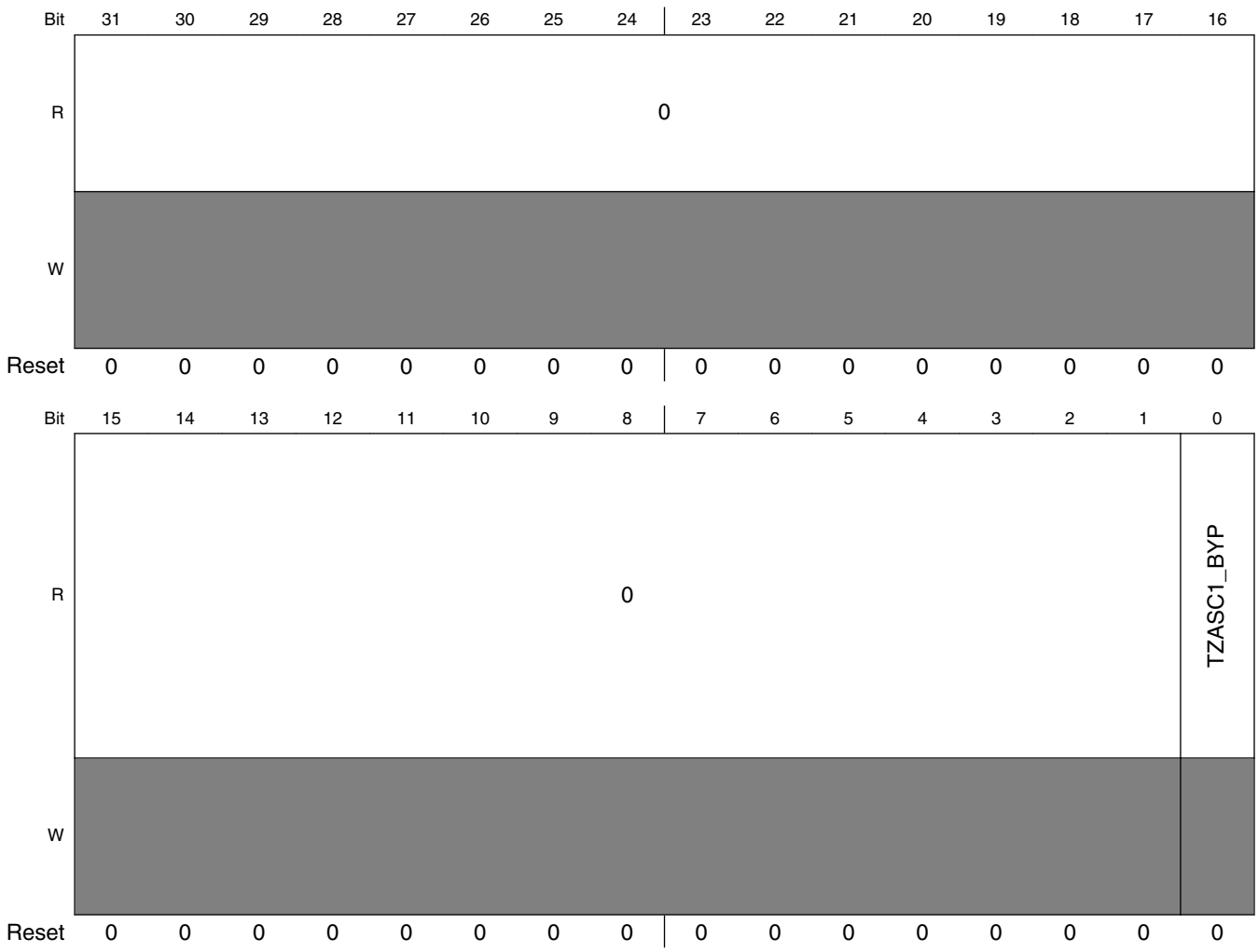
IOMUXC_GPR_GPR5 field descriptions (continued)

Field	Description
-	This field is reserved. Reserved

30.4.7 GPR9 General Purpose Register (IOMUXC_GPR_GPR9)

GPR Register

Address: 20E_4000h base + 24h offset = 20E_4024h



IOMUXC_GPR_GPR9 field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

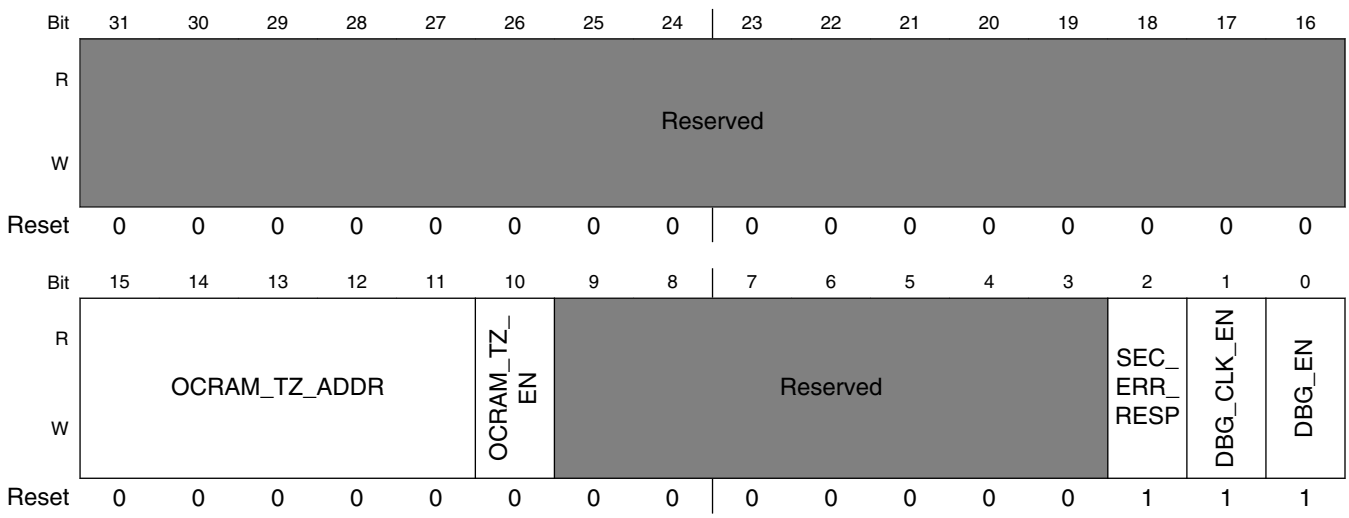
IOMUXC_GPR_GPR9 field descriptions (continued)

Field	Description
0 TZASC1_BYP	TZASC-1 BYPASS MUX control 0 The TZASC-1 is bypassed and the transactions to DDR are not being checked. 1 The TZASC-1 is not bypassed and the transactions to DDR are being monitored / checked.

30.4.8 GPR10 General Purpose Register (IOMUXC_GPR_GPR10)

GPR Register

Address: 20E_4000h base + 28h offset = 20E_4028h



IOMUXC_GPR_GPR10 field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–11 OCRAM_TZ_ADDR	OCRAM TrustZone (TZ) start address. This is the start address of the secure memory region within the OCRAM memory space is 4KB granularity. The start address affects the OCRAM transactions only if OCRAM_TZ_EN bit is set. The OCRAM TZ ENDADDR is not configurable and is set to the end of OCRAM memory space.
10 OCRAM_TZ_EN	OCRAM TrustZone (TZ) enable. 0 The TrustZone feature is disabled. Entire OCRAM space is available for all access types (secure/non-secure/user/supervisor). 1 The TrustZone feature is enabled. Access to address in the range specified by [ENDADDR:STARTADDR] follows the execution mode access policy described in CSU chapter.
9–3 -	This field is reserved. Reserved
2 SEC_ERR_RESP	Security error response enable for all security gaskets (on both AHB and AXI busses)

Table continues on the next page...

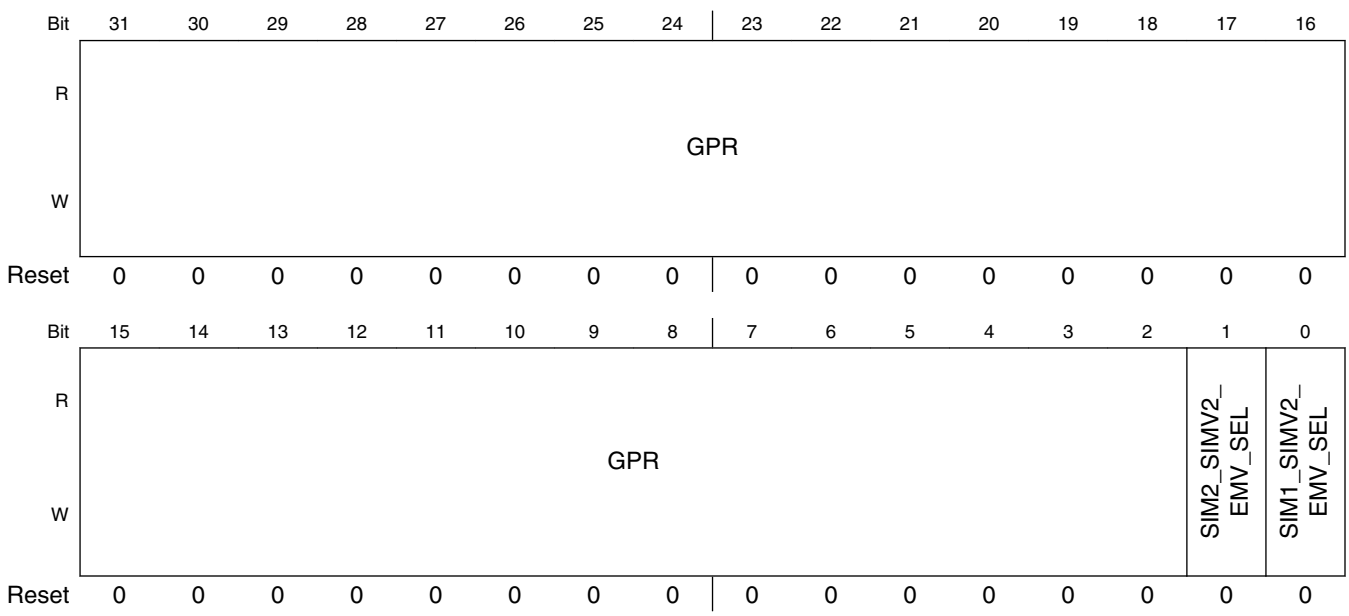
IOMUXC_GPR_GPR10 field descriptions (continued)

Field	Description
	0 OKEY response 1 SLVError (default)
1 DBG_CLK_EN	ARM Debug clock enable 0 Debug turned off. 1 Debug enabled (default).
0 DBG_EN	ARM non secure (non-invasive) debug enable 0 Debug turned off. 1 Debug enabled (default).

30.4.9 GPR14 General Purpose Register (IOMUXC_GPR_GPR14)

GPR Register

Address: 20E_4000h base + 38h offset = 20E_4038h



IOMUXC_GPR_GPR14 field descriptions

Field	Description
31–2 GPR	General purpose bits General purpose bits to be used by SoC integration. Bit Type: DEFAULT
1 SIM2_SIMV2_ EMV_SEL	SIM2 emv/simv2 select 0 select simv2 1 select emv

Table continues on the next page...

IOMUXC_GPR_GPR14 field descriptions (continued)

Field	Description
0 SIM1_SIMV2_ EMV_SEL	SIM1 emv/simv2 select 0 select simv2 1 select emv

30.5 IOMUXC Memory Map/Register Definition

IOMUXC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0014	SW_MUX_CTL_PAD_BOOT_MODE0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0)	32	R/W	0000_0000h	30.5.1/1311
20E_0018	SW_MUX_CTL_PAD_BOOT_MODE1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1)	32	R/W	0000_0000h	30.5.2/1312
20E_001C	SW_MUX_CTL_PAD_SNVS_TAMPER0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER0)	32	R/W	0000_0000h	30.5.3/1313
20E_0020	SW_MUX_CTL_PAD_SNVS_TAMPER1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER1)	32	R/W	0000_0000h	30.5.4/1314
20E_0024	SW_MUX_CTL_PAD_SNVS_TAMPER2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER2)	32	R/W	0000_0000h	30.5.5/1315
20E_0028	SW_MUX_CTL_PAD_SNVS_TAMPER3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER3)	32	R/W	0000_0000h	30.5.6/1316
20E_002C	SW_MUX_CTL_PAD_SNVS_TAMPER4 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER4)	32	R/W	0000_0000h	30.5.7/1317
20E_0030	SW_MUX_CTL_PAD_SNVS_TAMPER5 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER5)	32	R/W	0000_0000h	30.5.8/1318
20E_0034	SW_MUX_CTL_PAD_SNVS_TAMPER6 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER6)	32	R/W	0000_0000h	30.5.9/1319
20E_0038	SW_MUX_CTL_PAD_SNVS_TAMPER7 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER7)	32	R/W	0000_0000h	30.5.10/ 1320
20E_003C	SW_MUX_CTL_PAD_SNVS_TAMPER8 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER8)	32	R/W	0000_0000h	30.5.11/ 1321
20E_0040	SW_MUX_CTL_PAD_SNVS_TAMPER9 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER9)	32	R/W	0000_0000h	30.5.12/ 1322

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0044	SW_MUX_CTL_PAD_JTAG_MOD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_MOD)	32	R/W	0000_0000h	30.5.13/ 1323
20E_0048	SW_MUX_CTL_PAD_JTAG_TMS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TMS)	32	R/W	0000_0000h	30.5.14/ 1324
20E_004C	SW_MUX_CTL_PAD_JTAG_TDO SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TDO)	32	R/W	0000_0000h	30.5.15/ 1325
20E_0050	SW_MUX_CTL_PAD_JTAG_TDI SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TDI)	32	R/W	0000_0000h	30.5.16/ 1326
20E_0054	SW_MUX_CTL_PAD_JTAG_TCK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TCK)	32	R/W	0000_0000h	30.5.17/ 1327
20E_0058	SW_MUX_CTL_PAD_JTAG_TRST_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TRST_B)	32	R/W	0000_0000h	30.5.18/ 1328
20E_005C	SW_MUX_CTL_PAD_GPIO1_IO00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO00)	32	R/W	0000_0005h	30.5.19/ 1329
20E_0060	SW_MUX_CTL_PAD_GPIO1_IO01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO01)	32	R/W	0000_0005h	30.5.20/ 1330
20E_0064	SW_MUX_CTL_PAD_GPIO1_IO02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO02)	32	R/W	0000_0005h	30.5.21/ 1331
20E_0068	SW_MUX_CTL_PAD_GPIO1_IO03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO03)	32	R/W	0000_0005h	30.5.22/ 1332
20E_006C	SW_MUX_CTL_PAD_GPIO1_IO04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO04)	32	R/W	0000_0005h	30.5.23/ 1333
20E_0070	SW_MUX_CTL_PAD_GPIO1_IO05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO05)	32	R/W	0000_0005h	30.5.24/ 1334
20E_0074	SW_MUX_CTL_PAD_GPIO1_IO06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO06)	32	R/W	0000_0005h	30.5.25/ 1335
20E_0078	SW_MUX_CTL_PAD_GPIO1_IO07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO07)	32	R/W	0000_0005h	30.5.26/ 1336
20E_007C	SW_MUX_CTL_PAD_GPIO1_IO08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO08)	32	R/W	0000_0005h	30.5.27/ 1337
20E_0080	SW_MUX_CTL_PAD_GPIO1_IO09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09)	32	R/W	0000_0005h	30.5.28/ 1338
20E_0084	SW_MUX_CTL_PAD_UART1_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA)	32	R/W	0000_0005h	30.5.29/ 1339
20E_0088	SW_MUX_CTL_PAD_UART1_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA)	32	R/W	0000_0005h	30.5.30/ 1340
20E_008C	SW_MUX_CTL_PAD_UART1_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_CTS_B)	32	R/W	0000_0005h	30.5.31/ 1341
20E_0090	SW_MUX_CTL_PAD_UART1_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RTS_B)	32	R/W	0000_0005h	30.5.32/ 1342
20E_0094	SW_MUX_CTL_PAD_UART2_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA)	32	R/W	0000_0005h	30.5.33/ 1343

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0098	SW_MUX_CTL_PAD_UART2_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RX_DATA)	32	R/W	0000_0005h	30.5.34/ 1344
20E_009C	SW_MUX_CTL_PAD_UART2_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_CTS_B)	32	R/W	0000_0005h	30.5.35/ 1345
20E_00A0	SW_MUX_CTL_PAD_UART2_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RTS_B)	32	R/W	0000_0005h	30.5.36/ 1346
20E_00A4	SW_MUX_CTL_PAD_UART3_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_TX_DATA)	32	R/W	0000_0005h	30.5.37/ 1347
20E_00A8	SW_MUX_CTL_PAD_UART3_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA)	32	R/W	0000_0005h	30.5.38/ 1348
20E_00AC	SW_MUX_CTL_PAD_UART3_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B)	32	R/W	0000_0005h	30.5.39/ 1349
20E_00B0	SW_MUX_CTL_PAD_UART3_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B)	32	R/W	0000_0005h	30.5.40/ 1350
20E_00B4	SW_MUX_CTL_PAD_UART4_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART4_TX_DATA)	32	R/W	0000_0005h	30.5.41/ 1351
20E_00B8	SW_MUX_CTL_PAD_UART4_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART4_RX_DATA)	32	R/W	0000_0005h	30.5.42/ 1352
20E_00BC	SW_MUX_CTL_PAD_UART5_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART5_TX_DATA)	32	R/W	0000_0005h	30.5.43/ 1353
20E_00C0	SW_MUX_CTL_PAD_UART5_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART5_RX_DATA)	32	R/W	0000_0005h	30.5.44/ 1354
20E_00C4	SW_MUX_CTL_PAD_ENET1_RX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA0)	32	R/W	0000_0005h	30.5.45/ 1355
20E_00C8	SW_MUX_CTL_PAD_ENET1_RX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA1)	32	R/W	0000_0005h	30.5.46/ 1356
20E_00CC	SW_MUX_CTL_PAD_ENET1_RX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_EN)	32	R/W	0000_0005h	30.5.47/ 1357
20E_00D0	SW_MUX_CTL_PAD_ENET1_TX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA0)	32	R/W	0000_0005h	30.5.48/ 1358
20E_00D4	SW_MUX_CTL_PAD_ENET1_TX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA1)	32	R/W	0000_0005h	30.5.49/ 1359
20E_00D8	SW_MUX_CTL_PAD_ENET1_TX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_EN)	32	R/W	0000_0005h	30.5.50/ 1360

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_00DC	SW_MUX_CTL_PAD_ENET1_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_CLK)	32	R/W	0000_0005h	30.5.51/ 1361
20E_00E0	SW_MUX_CTL_PAD_ENET1_RX_ER SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_ER)	32	R/W	0000_0005h	30.5.52/ 1362
20E_00E4	SW_MUX_CTL_PAD_ENET2_RX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA0)	32	R/W	0000_0005h	30.5.53/ 1363
20E_00E8	SW_MUX_CTL_PAD_ENET2_RX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA1)	32	R/W	0000_0005h	30.5.54/ 1364
20E_00EC	SW_MUX_CTL_PAD_ENET2_RX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_EN)	32	R/W	0000_0005h	30.5.55/ 1365
20E_00F0	SW_MUX_CTL_PAD_ENET2_TX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA0)	32	R/W	0000_0005h	30.5.56/ 1366
20E_00F4	SW_MUX_CTL_PAD_ENET2_TX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA1)	32	R/W	0000_0005h	30.5.57/ 1367
20E_00F8	SW_MUX_CTL_PAD_ENET2_TX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_EN)	32	R/W	0000_0005h	30.5.58/ 1368
20E_00FC	SW_MUX_CTL_PAD_ENET2_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_CLK)	32	R/W	0000_0005h	30.5.59/ 1369
20E_0100	SW_MUX_CTL_PAD_ENET2_RX_ER SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_ER)	32	R/W	0000_0005h	30.5.60/ 1370
20E_0104	SW_MUX_CTL_PAD_LCD_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_CLK)	32	R/W	0000_0005h	30.5.61/ 1371
20E_0108	SW_MUX_CTL_PAD_LCD_ENABLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_ENABLE)	32	R/W	0000_0005h	30.5.62/ 1372
20E_010C	SW_MUX_CTL_PAD_LCD_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC)	32	R/W	0000_0005h	30.5.63/ 1373
20E_0110	SW_MUX_CTL_PAD_LCD_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC)	32	R/W	0000_0005h	30.5.64/ 1374
20E_0114	SW_MUX_CTL_PAD_LCD_RESET SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_RESET)	32	R/W	0000_0005h	30.5.65/ 1375
20E_0118	SW_MUX_CTL_PAD_LCD_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00)	32	R/W	0000_0005h	30.5.66/ 1376
20E_011C	SW_MUX_CTL_PAD_LCD_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01)	32	R/W	0000_0005h	30.5.67/ 1377
20E_0120	SW_MUX_CTL_PAD_LCD_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA02)	32	R/W	0000_0005h	30.5.68/ 1378
20E_0124	SW_MUX_CTL_PAD_LCD_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03)	32	R/W	0000_0005h	30.5.69/ 1379
20E_0128	SW_MUX_CTL_PAD_LCD_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04)	32	R/W	0000_0005h	30.5.70/ 1380

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_012C	SW_MUX_CTL_PAD_LCD_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA05)	32	R/W	0000_0005h	30.5.71/ 1381
20E_0130	SW_MUX_CTL_PAD_LCD_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06)	32	R/W	0000_0005h	30.5.72/ 1382
20E_0134	SW_MUX_CTL_PAD_LCD_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07)	32	R/W	0000_0005h	30.5.73/ 1383
20E_0138	SW_MUX_CTL_PAD_LCD_DATA08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08)	32	R/W	0000_0005h	30.5.74/ 1384
20E_013C	SW_MUX_CTL_PAD_LCD_DATA09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA09)	32	R/W	0000_0005h	30.5.75/ 1385
20E_0140	SW_MUX_CTL_PAD_LCD_DATA10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10)	32	R/W	0000_0005h	30.5.76/ 1386
20E_0144	SW_MUX_CTL_PAD_LCD_DATA11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11)	32	R/W	0000_0005h	30.5.77/ 1387
20E_0148	SW_MUX_CTL_PAD_LCD_DATA12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12)	32	R/W	0000_0005h	30.5.78/ 1388
20E_014C	SW_MUX_CTL_PAD_LCD_DATA13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA13)	32	R/W	0000_0005h	30.5.79/ 1389
20E_0150	SW_MUX_CTL_PAD_LCD_DATA14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14)	32	R/W	0000_0005h	30.5.80/ 1390
20E_0154	SW_MUX_CTL_PAD_LCD_DATA15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15)	32	R/W	0000_0005h	30.5.81/ 1391
20E_0158	SW_MUX_CTL_PAD_LCD_DATA16 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16)	32	R/W	0000_0005h	30.5.82/ 1392
20E_015C	SW_MUX_CTL_PAD_LCD_DATA17 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA17)	32	R/W	0000_0005h	30.5.83/ 1393
20E_0160	SW_MUX_CTL_PAD_LCD_DATA18 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18)	32	R/W	0000_0005h	30.5.84/ 1394
20E_0164	SW_MUX_CTL_PAD_LCD_DATA19 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19)	32	R/W	0000_0005h	30.5.85/ 1395
20E_0168	SW_MUX_CTL_PAD_LCD_DATA20 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA20)	32	R/W	0000_0005h	30.5.86/ 1396
20E_016C	SW_MUX_CTL_PAD_LCD_DATA21 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21)	32	R/W	0000_0005h	30.5.87/ 1397
20E_0170	SW_MUX_CTL_PAD_LCD_DATA22 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22)	32	R/W	0000_0005h	30.5.88/ 1398
20E_0174	SW_MUX_CTL_PAD_LCD_DATA23 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA23)	32	R/W	0000_0005h	30.5.89/ 1399
20E_0178	SW_MUX_CTL_PAD_NAND_RE_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_RE_B)	32	R/W	0000_0005h	30.5.90/ 1400
20E_017C	SW_MUX_CTL_PAD_NAND_WE_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_WE_B)	32	R/W	0000_0005h	30.5.91/ 1401
20E_0180	SW_MUX_CTL_PAD_NAND_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA00)	32	R/W	0000_0005h	30.5.92/ 1402

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0184	SW_MUX_CTL_PAD_NAND_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA01)	32	R/W	0000_0005h	30.5.93/ 1403
20E_0188	SW_MUX_CTL_PAD_NAND_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA02)	32	R/W	0000_0005h	30.5.94/ 1404
20E_018C	SW_MUX_CTL_PAD_NAND_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA03)	32	R/W	0000_0005h	30.5.95/ 1405
20E_0190	SW_MUX_CTL_PAD_NAND_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA04)	32	R/W	0000_0005h	30.5.96/ 1406
20E_0194	SW_MUX_CTL_PAD_NAND_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA05)	32	R/W	0000_0005h	30.5.97/ 1407
20E_0198	SW_MUX_CTL_PAD_NAND_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA06)	32	R/W	0000_0005h	30.5.98/ 1408
20E_019C	SW_MUX_CTL_PAD_NAND_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA07)	32	R/W	0000_0005h	30.5.99/ 1409
20E_01A0	SW_MUX_CTL_PAD_NAND_ALE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_ALE)	32	R/W	0000_0005h	30.5.100/ 1410
20E_01A4	SW_MUX_CTL_PAD_NAND_WP_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_WP_B)	32	R/W	0000_0005h	30.5.101/ 1411
20E_01A8	SW_MUX_CTL_PAD_NAND_READY_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_READY_B)	32	R/W	0000_0005h	30.5.102/ 1412
20E_01AC	SW_MUX_CTL_PAD_NAND_CE0_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CE0_B)	32	R/W	0000_0005h	30.5.103/ 1413
20E_01B0	SW_MUX_CTL_PAD_NAND_CE1_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CE1_B)	32	R/W	0000_0005h	30.5.104/ 1414
20E_01B4	SW_MUX_CTL_PAD_NAND_CLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CLE)	32	R/W	0000_0005h	30.5.105/ 1415
20E_01B8	SW_MUX_CTL_PAD_NAND_DQS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DQS)	32	R/W	0000_0005h	30.5.106/ 1416
20E_01BC	SW_MUX_CTL_PAD_SD1_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CMD)	32	R/W	0000_0005h	30.5.107/ 1417
20E_01C0	SW_MUX_CTL_PAD_SD1_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CLK)	32	R/W	0000_0005h	30.5.108/ 1418
20E_01C4	SW_MUX_CTL_PAD_SD1_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0)	32	R/W	0000_0005h	30.5.109/ 1419
20E_01C8	SW_MUX_CTL_PAD_SD1_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1)	32	R/W	0000_0005h	30.5.110/ 1420
20E_01CC	SW_MUX_CTL_PAD_SD1_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2)	32	R/W	0000_0005h	30.5.111/ 1421
20E_01D0	SW_MUX_CTL_PAD_SD1_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3)	32	R/W	0000_0005h	30.5.112/ 1422
20E_01D4	SW_MUX_CTL_PAD_CSI_MCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK)	32	R/W	0000_0005h	30.5.113/ 1423
20E_01D8	SW_MUX_CTL_PAD_CSI_PIXCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK)	32	R/W	0000_0005h	30.5.114/ 1424

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_01DC	SW_MUX_CTL_PAD_CSI_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC)	32	R/W	0000_0005h	30.5.115/ 1425
20E_01E0	SW_MUX_CTL_PAD_CSI_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC)	32	R/W	0000_0005h	30.5.116/ 1426
20E_01E4	SW_MUX_CTL_PAD_CSI_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA00)	32	R/W	0000_0005h	30.5.117/ 1427
20E_01E8	SW_MUX_CTL_PAD_CSI_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA01)	32	R/W	0000_0005h	30.5.118/ 1428
20E_01EC	SW_MUX_CTL_PAD_CSI_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA02)	32	R/W	0000_0005h	30.5.119/ 1429
20E_01F0	SW_MUX_CTL_PAD_CSI_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA03)	32	R/W	0000_0005h	30.5.120/ 1430
20E_01F4	SW_MUX_CTL_PAD_CSI_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA04)	32	R/W	0000_0005h	30.5.121/ 1431
20E_01F8	SW_MUX_CTL_PAD_CSI_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA05)	32	R/W	0000_0005h	30.5.122/ 1432
20E_01FC	SW_MUX_CTL_PAD_CSI_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA06)	32	R/W	0000_0005h	30.5.123/ 1433
20E_0200	SW_MUX_CTL_PAD_CSI_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA07)	32	R/W	0000_0005h	30.5.124/ 1434
20E_0204	SW_PAD_CTL_PAD_DRAM_ADDR00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR00)	32	R/W	0000_8000h	30.5.125/ 1435
20E_0208	SW_PAD_CTL_PAD_DRAM_ADDR01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR01)	32	R/W	0000_8000h	30.5.126/ 1438
20E_020C	SW_PAD_CTL_PAD_DRAM_ADDR02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR02)	32	R/W	0000_8000h	30.5.127/ 1441
20E_0210	SW_PAD_CTL_PAD_DRAM_ADDR03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR03)	32	R/W	0000_8000h	30.5.128/ 1444
20E_0214	SW_PAD_CTL_PAD_DRAM_ADDR04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR04)	32	R/W	0000_8000h	30.5.129/ 1447
20E_0218	SW_PAD_CTL_PAD_DRAM_ADDR05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR05)	32	R/W	0000_8000h	30.5.130/ 1450
20E_021C	SW_PAD_CTL_PAD_DRAM_ADDR06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR06)	32	R/W	0000_8000h	30.5.131/ 1453
20E_0220	SW_PAD_CTL_PAD_DRAM_ADDR07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR07)	32	R/W	0000_8000h	30.5.132/ 1456
20E_0224	SW_PAD_CTL_PAD_DRAM_ADDR08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR08)	32	R/W	0000_8000h	30.5.133/ 1459
20E_0228	SW_PAD_CTL_PAD_DRAM_ADDR09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR09)	32	R/W	0000_8000h	30.5.134/ 1462
20E_022C	SW_PAD_CTL_PAD_DRAM_ADDR10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR10)	32	R/W	0000_8000h	30.5.135/ 1465
20E_0230	SW_PAD_CTL_PAD_DRAM_ADDR11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR11)	32	R/W	0000_8000h	30.5.136/ 1468

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0234	SW_PAD_CTL_PAD_DRAM_ADDR12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR12)	32	R/W	0000_8000h	30.5.137/ 1471
20E_0238	SW_PAD_CTL_PAD_DRAM_ADDR13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR13)	32	R/W	0000_8000h	30.5.138/ 1474
20E_023C	SW_PAD_CTL_PAD_DRAM_ADDR14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR14)	32	R/W	0000_8000h	30.5.139/ 1477
20E_0240	SW_PAD_CTL_PAD_DRAM_ADDR15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR15)	32	R/W	0000_8000h	30.5.140/ 1480
20E_0244	SW_PAD_CTL_PAD_DRAM_DQM0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM0)	32	R/W	0000_8030h	30.5.141/ 1483
20E_0248	SW_PAD_CTL_PAD_DRAM_DQM1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM1)	32	R/W	0000_8030h	30.5.142/ 1486
20E_024C	SW_PAD_CTL_PAD_DRAM_RAS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_RAS_B)	32	R/W	0000_8030h	30.5.143/ 1489
20E_0250	SW_PAD_CTL_PAD_DRAM_CAS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CAS_B)	32	R/W	0000_8030h	30.5.144/ 1492
20E_0254	SW_PAD_CTL_PAD_DRAM_CS0_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CS0_B)	32	R/W	0000_8000h	30.5.145/ 1495
20E_0258	SW_PAD_CTL_PAD_DRAM_CS1_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CS1_B)	32	R/W	0000_8000h	30.5.146/ 1498
20E_025C	SW_PAD_CTL_PAD_DRAM_SDWE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDWE_B)	32	R/W	0000_8000h	30.5.147/ 1501
20E_0260	SW_PAD_CTL_PAD_DRAM_ODT0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT0)	32	R/W	0000_3030h	30.5.148/ 1504
20E_0264	SW_PAD_CTL_PAD_DRAM_ODT1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT1)	32	R/W	0000_3030h	30.5.149/ 1507
20E_0268	SW_PAD_CTL_PAD_DRAM_SDBA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA0)	32	R/W	0000_8000h	30.5.150/ 1510
20E_026C	SW_PAD_CTL_PAD_DRAM_SDBA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA1)	32	R/W	0000_8000h	30.5.151/ 1513
20E_0270	SW_PAD_CTL_PAD_DRAM_SDBA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA2)	32	R/W	0000_B000h	30.5.152/ 1516
20E_0274	SW_PAD_CTL_PAD_DRAM_SDCKE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE0)	32	R/W	0000_3000h	30.5.153/ 1519
20E_0278	SW_PAD_CTL_PAD_DRAM_SDCKE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE1)	32	R/W	0000_3000h	30.5.154/ 1522
20E_027C	SW_PAD_CTL_PAD_DRAM_SDCLK0_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK0_P)	32	R/W	0000_8030h	30.5.155/ 1525
20E_0280	SW_PAD_CTL_PAD_DRAM_SDQS0_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS0_P)	32	R/W	0000_2030h	30.5.156/ 1528
20E_0284	SW_PAD_CTL_PAD_DRAM_SDQS1_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS1_P)	32	R/W	0000_2030h	30.5.157/ 1531

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0288	SW_PAD_CTL_PAD_DRAM_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET)	32	R/W	0008_3030h	30.5.158/ 1534
20E_028C	SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_TEST_MODE)	32	R/W	0000_30A0h	30.5.159/ 1536
20E_0290	SW_PAD_CTL_PAD_POR_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_POR_B)	32	R/W	0001_B0A0h	30.5.160/ 1538
20E_0294	SW_PAD_CTL_PAD_ONOFF SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ONOFF)	32	R/W	0001_B0A0h	30.5.161/ 1540
20E_0298	SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ)	32	R/W	0000_B8A0h	30.5.162/ 1542
20E_029C	SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ)	32	R/W	0000_20A0h	30.5.163/ 1544
20E_02A0	SW_PAD_CTL_PAD_BOOT_MODE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE0)	32	R/W	0001_30A0h	30.5.164/ 1546
20E_02A4	SW_PAD_CTL_PAD_BOOT_MODE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE1)	32	R/W	0001_30A0h	30.5.165/ 1548
20E_02A8	SW_PAD_CTL_PAD_SNVS_TAMPER0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER0)	32	R/W	0001_10A0h	30.5.166/ 1550
20E_02AC	SW_PAD_CTL_PAD_SNVS_TAMPER1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER1)	32	R/W	0001_10A0h	30.5.167/ 1552
20E_02B0	SW_PAD_CTL_PAD_SNVS_TAMPER2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER2)	32	R/W	0001_10A0h	30.5.168/ 1554
20E_02B4	SW_PAD_CTL_PAD_SNVS_TAMPER3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER3)	32	R/W	0001_10A0h	30.5.169/ 1556
20E_02B8	SW_PAD_CTL_PAD_SNVS_TAMPER4 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER4)	32	R/W	0001_10A0h	30.5.170/ 1558
20E_02BC	SW_PAD_CTL_PAD_SNVS_TAMPER5 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER5)	32	R/W	0001_10A0h	30.5.171/ 1560
20E_02C0	SW_PAD_CTL_PAD_SNVS_TAMPER6 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER6)	32	R/W	0001_10A0h	30.5.172/ 1562
20E_02C4	SW_PAD_CTL_PAD_SNVS_TAMPER7 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER7)	32	R/W	0001_10A0h	30.5.173/ 1564
20E_02C8	SW_PAD_CTL_PAD_SNVS_TAMPER8 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER8)	32	R/W	0001_10A0h	30.5.174/ 1566

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_02CC	SW_PAD_CTL_PAD_SNVS_TAMPER9 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER9)	32	R/W	0001_10A0h	30.5.175/ 1568
20E_02D0	SW_PAD_CTL_PAD_JTAG_MOD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD)	32	R/W	0000_B0A0h	30.5.176/ 1570
20E_02D4	SW_PAD_CTL_PAD_JTAG_TMS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS)	32	R/W	0000_70A0h	30.5.177/ 1572
20E_02D8	SW_PAD_CTL_PAD_JTAG_TDO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO)	32	R/W	0000_B0B1h	30.5.178/ 1574
20E_02DC	SW_PAD_CTL_PAD_JTAG_TDI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI)	32	R/W	0000_70A0h	30.5.179/ 1576
20E_02E0	SW_PAD_CTL_PAD_JTAG_TCK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK)	32	R/W	0000_70A0h	30.5.180/ 1578
20E_02E4	SW_PAD_CTL_PAD_JTAG_TRST_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B)	32	R/W	0000_70A0h	30.5.181/ 1580
20E_02E8	SW_PAD_CTL_PAD_GPIO1_IO00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO00)	32	R/W	0000_10B0h	30.5.182/ 1582
20E_02EC	SW_PAD_CTL_PAD_GPIO1_IO01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO01)	32	R/W	0000_10B0h	30.5.183/ 1584
20E_02F0	SW_PAD_CTL_PAD_GPIO1_IO02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO02)	32	R/W	0000_10B0h	30.5.184/ 1586
20E_02F4	SW_PAD_CTL_PAD_GPIO1_IO03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO03)	32	R/W	0000_10B0h	30.5.185/ 1588
20E_02F8	SW_PAD_CTL_PAD_GPIO1_IO04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO04)	32	R/W	0000_10B0h	30.5.186/ 1590
20E_02FC	SW_PAD_CTL_PAD_GPIO1_IO05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO05)	32	R/W	0000_10B0h	30.5.187/ 1592
20E_0300	SW_PAD_CTL_PAD_GPIO1_IO06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO06)	32	R/W	0000_10B0h	30.5.188/ 1594
20E_0304	SW_PAD_CTL_PAD_GPIO1_IO07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO07)	32	R/W	0000_10B0h	30.5.189/ 1596
20E_0308	SW_PAD_CTL_PAD_GPIO1_IO08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08)	32	R/W	0000_10B0h	30.5.190/ 1598
20E_030C	SW_PAD_CTL_PAD_GPIO1_IO09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09)	32	R/W	0000_10B0h	30.5.191/ 1600
20E_0310	SW_PAD_CTL_PAD_UART1_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA)	32	R/W	0000_10B0h	30.5.192/ 1602
20E_0314	SW_PAD_CTL_PAD_UART1_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA)	32	R/W	0000_10B0h	30.5.193/ 1604
20E_0318	SW_PAD_CTL_PAD_UART1_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_CTS_B)	32	R/W	0000_10B0h	30.5.194/ 1606
20E_031C	SW_PAD_CTL_PAD_UART1_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RTS_B)	32	R/W	0000_10B0h	30.5.195/ 1608

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0320	SW_PAD_CTL_PAD_UART2_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA)	32	R/W	0000_10B0h	30.5.196/ 1610
20E_0324	SW_PAD_CTL_PAD_UART2_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA)	32	R/W	0000_10B0h	30.5.197/ 1612
20E_0328	SW_PAD_CTL_PAD_UART2_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_CTS_B)	32	R/W	0000_10B0h	30.5.198/ 1614
20E_032C	SW_PAD_CTL_PAD_UART2_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RTS_B)	32	R/W	0000_10B0h	30.5.199/ 1616
20E_0330	SW_PAD_CTL_PAD_UART3_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA)	32	R/W	0000_10B0h	30.5.200/ 1618
20E_0334	SW_PAD_CTL_PAD_UART3_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA)	32	R/W	0000_10B0h	30.5.201/ 1620
20E_0338	SW_PAD_CTL_PAD_UART3_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B)	32	R/W	0000_10B0h	30.5.202/ 1622
20E_033C	SW_PAD_CTL_PAD_UART3_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B)	32	R/W	0000_10B0h	30.5.203/ 1624
20E_0340	SW_PAD_CTL_PAD_UART4_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART4_TX_DATA)	32	R/W	0000_10B0h	30.5.204/ 1626
20E_0344	SW_PAD_CTL_PAD_UART4_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART4_RX_DATA)	32	R/W	0000_10B0h	30.5.205/ 1628
20E_0348	SW_PAD_CTL_PAD_UART5_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART5_TX_DATA)	32	R/W	0000_10B0h	30.5.206/ 1630
20E_034C	SW_PAD_CTL_PAD_UART5_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART5_RX_DATA)	32	R/W	0000_10B0h	30.5.207/ 1632
20E_0350	SW_PAD_CTL_PAD_ENET1_RX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA0)	32	R/W	0000_10B0h	30.5.208/ 1634
20E_0354	SW_PAD_CTL_PAD_ENET1_RX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA1)	32	R/W	0000_10B0h	30.5.209/ 1636
20E_0358	SW_PAD_CTL_PAD_ENET1_RX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_EN)	32	R/W	0000_10B0h	30.5.210/ 1638
20E_035C	SW_PAD_CTL_PAD_ENET1_TX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA0)	32	R/W	0000_10B0h	30.5.211/ 1640
20E_0360	SW_PAD_CTL_PAD_ENET1_TX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA1)	32	R/W	0000_10B0h	30.5.212/ 1642

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0364	SW_PAD_CTL_PAD_ENET1_TX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_EN)	32	R/W	0000_10B0h	30.5.213/ 1644
20E_0368	SW_PAD_CTL_PAD_ENET1_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK)	32	R/W	0000_10B0h	30.5.214/ 1646
20E_036C	SW_PAD_CTL_PAD_ENET1_RX_ER SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_ER)	32	R/W	0000_10B0h	30.5.215/ 1648
20E_0370	SW_PAD_CTL_PAD_ENET2_RX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA0)	32	R/W	0000_10B0h	30.5.216/ 1650
20E_0374	SW_PAD_CTL_PAD_ENET2_RX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA1)	32	R/W	0000_10B0h	30.5.217/ 1652
20E_0378	SW_PAD_CTL_PAD_ENET2_RX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_EN)	32	R/W	0000_10B0h	30.5.218/ 1654
20E_037C	SW_PAD_CTL_PAD_ENET2_TX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA0)	32	R/W	0000_10B0h	30.5.219/ 1656
20E_0380	SW_PAD_CTL_PAD_ENET2_TX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA1)	32	R/W	0000_10B0h	30.5.220/ 1658
20E_0384	SW_PAD_CTL_PAD_ENET2_TX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_EN)	32	R/W	0000_10B0h	30.5.221/ 1660
20E_0388	SW_PAD_CTL_PAD_ENET2_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_CLK)	32	R/W	0000_10B0h	30.5.222/ 1662
20E_038C	SW_PAD_CTL_PAD_ENET2_RX_ER SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_ER)	32	R/W	0000_10B0h	30.5.223/ 1664
20E_0390	SW_PAD_CTL_PAD_LCD_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_CLK)	32	R/W	0000_10B0h	30.5.224/ 1666
20E_0394	SW_PAD_CTL_PAD_LCD_ENABLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE)	32	R/W	0000_10B0h	30.5.225/ 1668
20E_0398	SW_PAD_CTL_PAD_LCD_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC)	32	R/W	0000_10B0h	30.5.226/ 1670
20E_039C	SW_PAD_CTL_PAD_LCD_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC)	32	R/W	0000_10B0h	30.5.227/ 1672
20E_03A0	SW_PAD_CTL_PAD_LCD_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_RESET)	32	R/W	0000_10B0h	30.5.228/ 1674
20E_03A4	SW_PAD_CTL_PAD_LCD_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00)	32	R/W	0000_10B0h	30.5.229/ 1676
20E_03A8	SW_PAD_CTL_PAD_LCD_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01)	32	R/W	0000_10B0h	30.5.230/ 1678
20E_03AC	SW_PAD_CTL_PAD_LCD_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02)	32	R/W	0000_10B0h	30.5.231/ 1680
20E_03B0	SW_PAD_CTL_PAD_LCD_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03)	32	R/W	0000_10B0h	30.5.232/ 1682

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_03B4	SW_PAD_CTL_PAD_LCD_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04)	32	R/W	0000_10B0h	30.5.233/ 1684
20E_03B8	SW_PAD_CTL_PAD_LCD_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05)	32	R/W	0000_10B0h	30.5.234/ 1686
20E_03BC	SW_PAD_CTL_PAD_LCD_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06)	32	R/W	0000_10B0h	30.5.235/ 1688
20E_03C0	SW_PAD_CTL_PAD_LCD_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07)	32	R/W	0000_10B0h	30.5.236/ 1690
20E_03C4	SW_PAD_CTL_PAD_LCD_DATA08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08)	32	R/W	0000_10B0h	30.5.237/ 1692
20E_03C8	SW_PAD_CTL_PAD_LCD_DATA09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09)	32	R/W	0000_10B0h	30.5.238/ 1694
20E_03CC	SW_PAD_CTL_PAD_LCD_DATA10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10)	32	R/W	0000_10B0h	30.5.239/ 1696
20E_03D0	SW_PAD_CTL_PAD_LCD_DATA11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11)	32	R/W	0000_10B0h	30.5.240/ 1698
20E_03D4	SW_PAD_CTL_PAD_LCD_DATA12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12)	32	R/W	0000_10B0h	30.5.241/ 1700
20E_03D8	SW_PAD_CTL_PAD_LCD_DATA13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13)	32	R/W	0000_10B0h	30.5.242/ 1702
20E_03DC	SW_PAD_CTL_PAD_LCD_DATA14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14)	32	R/W	0000_10B0h	30.5.243/ 1704
20E_03E0	SW_PAD_CTL_PAD_LCD_DATA15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15)	32	R/W	0000_10B0h	30.5.244/ 1706
20E_03E4	SW_PAD_CTL_PAD_LCD_DATA16 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16)	32	R/W	0000_10B0h	30.5.245/ 1708
20E_03E8	SW_PAD_CTL_PAD_LCD_DATA17 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17)	32	R/W	0000_10B0h	30.5.246/ 1710
20E_03EC	SW_PAD_CTL_PAD_LCD_DATA18 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18)	32	R/W	0000_10B0h	30.5.247/ 1712
20E_03F0	SW_PAD_CTL_PAD_LCD_DATA19 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19)	32	R/W	0000_10B0h	30.5.248/ 1714
20E_03F4	SW_PAD_CTL_PAD_LCD_DATA20 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20)	32	R/W	0000_10B0h	30.5.249/ 1716
20E_03F8	SW_PAD_CTL_PAD_LCD_DATA21 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21)	32	R/W	0000_10B0h	30.5.250/ 1718
20E_03FC	SW_PAD_CTL_PAD_LCD_DATA22 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22)	32	R/W	0000_10B0h	30.5.251/ 1720
20E_0400	SW_PAD_CTL_PAD_LCD_DATA23 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23)	32	R/W	0000_10B0h	30.5.252/ 1722
20E_0404	SW_PAD_CTL_PAD_NAND_RE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_RE_B)	32	R/W	0000_10B0h	30.5.253/ 1724
20E_0408	SW_PAD_CTL_PAD_NAND_WE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_WE_B)	32	R/W	0000_10B0h	30.5.254/ 1726

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_040C	SW_PAD_CTL_PAD_NAND_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA00)	32	R/W	0000_10B0h	30.5.255/ 1728
20E_0410	SW_PAD_CTL_PAD_NAND_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA01)	32	R/W	0000_10B0h	30.5.256/ 1730
20E_0414	SW_PAD_CTL_PAD_NAND_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA02)	32	R/W	0000_10B0h	30.5.257/ 1732
20E_0418	SW_PAD_CTL_PAD_NAND_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA03)	32	R/W	0000_10B0h	30.5.258/ 1734
20E_041C	SW_PAD_CTL_PAD_NAND_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA04)	32	R/W	0000_10B0h	30.5.259/ 1736
20E_0420	SW_PAD_CTL_PAD_NAND_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA05)	32	R/W	0000_10B0h	30.5.260/ 1738
20E_0424	SW_PAD_CTL_PAD_NAND_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA06)	32	R/W	0000_10B0h	30.5.261/ 1740
20E_0428	SW_PAD_CTL_PAD_NAND_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA07)	32	R/W	0000_10B0h	30.5.262/ 1742
20E_042C	SW_PAD_CTL_PAD_NAND_ALE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_ALE)	32	R/W	0000_10B0h	30.5.263/ 1744
20E_0430	SW_PAD_CTL_PAD_NAND_WP_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_WP_B)	32	R/W	0000_10B0h	30.5.264/ 1746
20E_0434	SW_PAD_CTL_PAD_NAND_READY_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_READY_B)	32	R/W	0000_10B0h	30.5.265/ 1748
20E_0438	SW_PAD_CTL_PAD_NAND_CE0_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CE0_B)	32	R/W	0000_10B0h	30.5.266/ 1750
20E_043C	SW_PAD_CTL_PAD_NAND_CE1_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CE1_B)	32	R/W	0000_10B0h	30.5.267/ 1752
20E_0440	SW_PAD_CTL_PAD_NAND_CLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CLE)	32	R/W	0000_10B0h	30.5.268/ 1754
20E_0444	SW_PAD_CTL_PAD_NAND_DQS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DQS)	32	R/W	0000_10B0h	30.5.269/ 1756
20E_0448	SW_PAD_CTL_PAD_SD1_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CMD)	32	R/W	0000_10B0h	30.5.270/ 1758
20E_044C	SW_PAD_CTL_PAD_SD1_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CLK)	32	R/W	0000_10B0h	30.5.271/ 1760
20E_0450	SW_PAD_CTL_PAD_SD1_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0)	32	R/W	0000_10B0h	30.5.272/ 1762
20E_0454	SW_PAD_CTL_PAD_SD1_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1)	32	R/W	0000_10B0h	30.5.273/ 1764
20E_0458	SW_PAD_CTL_PAD_SD1_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2)	32	R/W	0000_10B0h	30.5.274/ 1766
20E_045C	SW_PAD_CTL_PAD_SD1_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3)	32	R/W	0000_10B0h	30.5.275/ 1768
20E_0460	SW_PAD_CTL_PAD_CSI_MCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK)	32	R/W	0000_10B0h	30.5.276/ 1770

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0464	SW_PAD_CTL_PAD_CSI_PIXCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK)	32	R/W	0000_10B0h	30.5.277/ 1772
20E_0468	SW_PAD_CTL_PAD_CSI_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC)	32	R/W	0000_10B0h	30.5.278/ 1774
20E_046C	SW_PAD_CTL_PAD_CSI_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC)	32	R/W	0000_10B0h	30.5.279/ 1776
20E_0470	SW_PAD_CTL_PAD_CSI_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA00)	32	R/W	0000_10B0h	30.5.280/ 1778
20E_0474	SW_PAD_CTL_PAD_CSI_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA01)	32	R/W	0000_10B0h	30.5.281/ 1780
20E_0478	SW_PAD_CTL_PAD_CSI_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA02)	32	R/W	0000_10B0h	30.5.282/ 1782
20E_047C	SW_PAD_CTL_PAD_CSI_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA03)	32	R/W	0000_10B0h	30.5.283/ 1784
20E_0480	SW_PAD_CTL_PAD_CSI_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA04)	32	R/W	0000_10B0h	30.5.284/ 1786
20E_0484	SW_PAD_CTL_PAD_CSI_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA05)	32	R/W	0000_10B0h	30.5.285/ 1788
20E_0488	SW_PAD_CTL_PAD_CSI_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA06)	32	R/W	0000_10B0h	30.5.286/ 1790
20E_048C	SW_PAD_CTL_PAD_CSI_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA07)	32	R/W	0000_10B0h	30.5.287/ 1792
20E_0490	SW_PAD_CTL_GRP_ADDDS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_ADDDS)	32	R/W	0000_0030h	30.5.288/ 1793
20E_0494	SW_PAD_CTL_GRP_DDRMODE_CTL SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRMODE_CTL)	32	R/W	0000_0000h	30.5.289/ 1794
20E_0498	SW_PAD_CTL_GRP_B0DS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_B0DS)	32	R/W	0000_0030h	30.5.290/ 1795
20E_049C	SW_PAD_CTL_GRP_DDRPK SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRPK)	32	R/W	0000_2000h	30.5.291/ 1796
20E_04A0	SW_PAD_CTL_GRP_CTLDS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_CTLDS)	32	R/W	0000_0030h	30.5.292/ 1796
20E_04A4	SW_PAD_CTL_GRP_B1DS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_B1DS)	32	R/W	0000_0030h	30.5.293/ 1797
20E_04A8	SW_PAD_CTL_GRP_DDRHYS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRHYS)	32	R/W	0000_0000h	30.5.294/ 1798
20E_04AC	SW_PAD_CTL_GRP_DDRPKE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRPKE)	32	R/W	0000_1000h	30.5.295/ 1799
20E_04B0	SW_PAD_CTL_GRP_DDRMODE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRMODE)	32	R/W	0000_0000h	30.5.296/ 1800
20E_04B4	SW_PAD_CTL_GRP_DDR_TYPE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDR_TYPE)	32	R/W	0008_0000h	30.5.297/ 1801
20E_04B8	USB_OTG1_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_ID_SELECT_INPUT)	32	R/W	0000_0000h	30.5.298/ 1802

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_04BC	USB_OTG2_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_ID_SELECT_INPUT)	32	R/W	0000_0000h	30.5.299/ 1802
20E_04C0	CCM_PMIC_READY_SELECT_INPUT DAISY Register (IOMUXC_CCM_PMIC_READY_SELECT_INPUT)	32	R/W	0000_0000h	30.5.300/ 1803
20E_04C4	CSI_DATA02_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA02_SELECT_INPUT)	32	R/W	0000_0000h	30.5.301/ 1804
20E_04C8	CSI_DATA03_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA03_SELECT_INPUT)	32	R/W	0000_0000h	30.5.302/ 1805
20E_04CC	CSI_DATA05_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA05_SELECT_INPUT)	32	R/W	0000_0000h	30.5.303/ 1806
20E_04D0	CSI_DATA00_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA00_SELECT_INPUT)	32	R/W	0000_0000h	30.5.304/ 1807
20E_04D4	CSI_DATA01_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA01_SELECT_INPUT)	32	R/W	0000_0000h	30.5.305/ 1808
20E_04D8	CSI_DATA04_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA04_SELECT_INPUT)	32	R/W	0000_0000h	30.5.306/ 1809
20E_04DC	CSI_DATA06_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA06_SELECT_INPUT)	32	R/W	0000_0000h	30.5.307/ 1810
20E_04E0	CSI_DATA07_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA07_SELECT_INPUT)	32	R/W	0000_0000h	30.5.308/ 1811
20E_04E4	CSI_DATA08_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA08_SELECT_INPUT)	32	R/W	0000_0000h	30.5.309/ 1812
20E_04E8	CSI_DATA09_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA09_SELECT_INPUT)	32	R/W	0000_0000h	30.5.310/ 1813
20E_04EC	CSI_DATA10_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA10_SELECT_INPUT)	32	R/W	0000_0000h	30.5.311/ 1814
20E_04F0	CSI_DATA11_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA11_SELECT_INPUT)	32	R/W	0000_0000h	30.5.312/ 1815
20E_04F4	CSI_DATA12_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA12_SELECT_INPUT)	32	R/W	0000_0000h	30.5.313/ 1816
20E_04F8	CSI_DATA13_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA13_SELECT_INPUT)	32	R/W	0000_0000h	30.5.314/ 1817
20E_04FC	CSI_DATA14_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA14_SELECT_INPUT)	32	R/W	0000_0000h	30.5.315/ 1818
20E_0500	CSI_DATA15_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA15_SELECT_INPUT)	32	R/W	0000_0000h	30.5.316/ 1819
20E_0504	CSI_DATA16_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA16_SELECT_INPUT)	32	R/W	0000_0000h	30.5.317/ 1820
20E_0508	CSI_DATA17_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA17_SELECT_INPUT)	32	R/W	0000_0000h	30.5.318/ 1821
20E_050C	CSI_DATA18_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA18_SELECT_INPUT)	32	R/W	0000_0000h	30.5.319/ 1822
20E_0510	CSI_DATA19_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA19_SELECT_INPUT)	32	R/W	0000_0000h	30.5.320/ 1823

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0514	CSI_DATA20_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA20_SELECT_INPUT)	32	R/W	0000_0000h	30.5.321/ 1824
20E_0518	CSI_DATA21_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA21_SELECT_INPUT)	32	R/W	0000_0000h	30.5.322/ 1825
20E_051C	CSI_DATA22_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA22_SELECT_INPUT)	32	R/W	0000_0000h	30.5.323/ 1826
20E_0520	CSI_DATA23_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA23_SELECT_INPUT)	32	R/W	0000_0000h	30.5.324/ 1827
20E_0524	CSI_HSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_HSYNC_SELECT_INPUT)	32	R/W	0000_0000h	30.5.325/ 1828
20E_0528	CSI_PIXCLK_SELECT_INPUT DAISY Register (IOMUXC_CSI_PIXCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.326/ 1829
20E_052C	CSI_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_VSYNC_SELECT_INPUT)	32	R/W	0000_0000h	30.5.327/ 1830
20E_0530	CSI_FIELD_SELECT_INPUT DAISY Register (IOMUXC_CSI_FIELD_SELECT_INPUT)	32	R/W	0000_0000h	30.5.328/ 1831
20E_0534	ECSPI1_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.329/ 1832
20E_0538	ECSPI1_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MISO_SELECT_INPUT)	32	R/W	0000_0000h	30.5.330/ 1833
20E_053C	ECSPI1_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	30.5.331/ 1834
20E_0540	ECSPI1_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.332/ 1835
20E_0544	ECSPI2_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.333/ 1836
20E_0548	ECSPI2_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MISO_SELECT_INPUT)	32	R/W	0000_0000h	30.5.334/ 1837
20E_054C	ECSPI2_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	30.5.335/ 1838
20E_0550	ECSPI2_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.336/ 1839
20E_0554	ECSPI3_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.337/ 1840
20E_0558	ECSPI3_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MISO_SELECT_INPUT)	32	R/W	0000_0000h	30.5.338/ 1841
20E_055C	ECSPI3_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	30.5.339/ 1842
20E_0560	ECSPI3_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.340/ 1843
20E_0564	ECSPI4_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.341/ 1844
20E_0568	ECSPI4_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MISO_SELECT_INPUT)	32	R/W	0000_0000h	30.5.342/ 1845

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_056C	ECSPI4_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	30.5.343/ 1846
20E_0570	ECSPI4_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.344/ 1847
20E_0574	ENET1_REF_CLK1_SELECT_INPUT DAISY Register (IOMUXC_ENET1_REF_CLK1_SELECT_INPUT)	32	R/W	0000_0000h	30.5.345/ 1847
20E_0578	ENET1_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET1_MAC0_MDIO_SELECT_INPUT)	32	R/W	0000_0000h	30.5.346/ 1848
20E_057C	ENET2_REF_CLK2_SELECT_INPUT DAISY Register (IOMUXC_ENET2_REF_CLK2_SELECT_INPUT)	32	R/W	0000_0000h	30.5.347/ 1849
20E_0580	ENET2_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET2_MAC0_MDIO_SELECT_INPUT)	32	R/W	0000_0000h	30.5.348/ 1850
20E_0584	FLEXCAN1_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN1_RX_SELECT_INPUT)	32	R/W	0000_0000h	30.5.349/ 1850
20E_0588	FLEXCAN2_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN2_RX_SELECT_INPUT)	32	R/W	0000_0000h	30.5.350/ 1851
20E_058C	GPT1_CAPTURE1_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CAPTURE1_SELECT_INPUT)	32	R/W	0000_0000h	30.5.351/ 1852
20E_0590	GPT1_CAPTURE2_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CAPTURE2_SELECT_INPUT)	32	R/W	0000_0000h	30.5.352/ 1853
20E_0594	GPT1_CLK_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.353/ 1854
20E_0598	GPT2_CAPTURE1_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CAPTURE1_SELECT_INPUT)	32	R/W	0000_0000h	30.5.354/ 1855
20E_059C	GPT2_CAPTURE2_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CAPTURE2_SELECT_INPUT)	32	R/W	0000_0000h	30.5.355/ 1856
20E_05A0	GPT2_CLK_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.356/ 1857
20E_05A4	I2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SCL_SELECT_INPUT)	32	R/W	0000_0000h	30.5.357/ 1857
20E_05A8	I2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SDA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.358/ 1858
20E_05AC	I2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SCL_SELECT_INPUT)	32	R/W	0000_0000h	30.5.359/ 1859
20E_05B0	I2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SDA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.360/ 1859
20E_05B4	I2C3_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SCL_SELECT_INPUT)	32	R/W	0000_0000h	30.5.361/ 1860
20E_05B8	I2C3_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SDA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.362/ 1861
20E_05BC	I2C4_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SCL_SELECT_INPUT)	32	R/W	0000_0000h	30.5.363/ 1861
20E_05C0	I2C4_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SDA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.364/ 1862

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_05C4	KPP_COL0_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL0_SELECT_INPUT)	32	R/W	0000_0000h	30.5.365/ 1863
20E_05C8	KPP_COL1_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL1_SELECT_INPUT)	32	R/W	0000_0000h	30.5.366/ 1864
20E_05CC	KPP_COL2_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL2_SELECT_INPUT)	32	R/W	0000_0000h	30.5.367/ 1865
20E_05D0	KPP_ROW0_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW0_SELECT_INPUT)	32	R/W	0000_0000h	30.5.368/ 1866
20E_05D4	KPP_ROW1_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW1_SELECT_INPUT)	32	R/W	0000_0000h	30.5.369/ 1867
20E_05D8	KPP_ROW2_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW2_SELECT_INPUT)	32	R/W	0000_0000h	30.5.370/ 1868
20E_05DC	LCD_BUSY_SELECT_INPUT DAISY Register (IOMUXC_LCD_BUSY_SELECT_INPUT)	32	R/W	0000_0000h	30.5.371/ 1869
20E_05E0	SAI1_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_MCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.372/ 1870
20E_05E4	SAI1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.373/ 1871
20E_05E8	SAI1_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.374/ 1872
20E_05EC	SAI1_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	30.5.375/ 1873
20E_05F0	SAI2_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_MCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.376/ 1874
20E_05F4	SAI2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.377/ 1875
20E_05F8	SAI2_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.378/ 1876
20E_05FC	SAI2_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	30.5.379/ 1877
20E_0600	SAI3_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_MCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.380/ 1878
20E_0604	SAI3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.381/ 1879
20E_0608	SAI3_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.382/ 1880
20E_060C	SAI3_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	30.5.383/ 1881
20E_0610	SDMA_EVENTS0_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS0_SELECT_INPUT)	32	R/W	0000_0000h	30.5.384/ 1881
20E_0614	SDMA_EVENTS1_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS1_SELECT_INPUT)	32	R/W	0000_0000h	30.5.385/ 1882
20E_0618	SPDIF_IN_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_IN_SELECT_INPUT)	32	R/W	0000_0000h	30.5.386/ 1883

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_061C	SPDIF_EXT_CLK_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_EXT_CLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.387/ 1884
20E_0620	UART1_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART1_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.388/ 1884
20E_0624	UART1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART1_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.389/ 1885
20E_0628	UART2_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART2_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.390/ 1886
20E_062C	UART2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART2_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.391/ 1886
20E_0630	UART3_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART3_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.392/ 1887
20E_0634	UART3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART3_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.393/ 1888
20E_0638	UART4_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART4_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.394/ 1888
20E_063C	UART4_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART4_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.395/ 1889
20E_0640	UART5_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART5_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.396/ 1890
20E_0644	UART5_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART5_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.397/ 1890
20E_0648	UART6_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART6_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.398/ 1891
20E_064C	UART6_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART6_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.399/ 1892
20E_0650	UART7_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART7_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.400/ 1892
20E_0654	UART7_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART7_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.401/ 1893
20E_0658	UART8_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART8_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.402/ 1894
20E_065C	UART8_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART8_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	30.5.403/ 1894
20E_0660	USB_OTG2_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_OC_SELECT_INPUT)	32	R/W	0000_0000h	30.5.404/ 1895
20E_0664	USB_OTG_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG_OC_SELECT_INPUT)	32	R/W	0000_0000h	30.5.405/ 1896
20E_0668	USDHC1_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_CD_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.406/ 1896
20E_066C	USDHC1_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_WP_SELECT_INPUT)	32	R/W	0000_0000h	30.5.407/ 1897
20E_0670	USDHC2_CLK_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CLK_SELECT_INPUT)	32	R/W	0000_0000h	30.5.408/ 1898

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_0674	USDHC2_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CD_B_SELECT_INPUT)	32	R/W	0000_0000h	30.5.409/ 1898
20E_0678	USDHC2_CMD_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CMD_SELECT_INPUT)	32	R/W	0000_0000h	30.5.410/ 1899
20E_067C	USDHC2_DATA0_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA0_SELECT_INPUT)	32	R/W	0000_0000h	30.5.411/ 1900
20E_0680	USDHC2_DATA1_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA1_SELECT_INPUT)	32	R/W	0000_0000h	30.5.412/ 1900
20E_0684	USDHC2_DATA2_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA2_SELECT_INPUT)	32	R/W	0000_0000h	30.5.413/ 1901
20E_0688	USDHC2_DATA3_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA3_SELECT_INPUT)	32	R/W	0000_0000h	30.5.414/ 1902
20E_068C	USDHC2_DATA4_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA4_SELECT_INPUT)	32	R/W	0000_0000h	30.5.415/ 1902
20E_0690	USDHC2_DATA5_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA5_SELECT_INPUT)	32	R/W	0000_0000h	30.5.416/ 1903
20E_0694	USDHC2_DATA6_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA6_SELECT_INPUT)	32	R/W	0000_0000h	30.5.417/ 1904
20E_0698	USDHC2_DATA7_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA7_SELECT_INPUT)	32	R/W	0000_0000h	30.5.418/ 1904
20E_069C	USDHC2_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_WP_SELECT_INPUT)	32	R/W	0000_0000h	30.5.419/ 1905

30.5.1 SW_MUX_CTL_PAD_BOOT_MODE0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0)

SW_MUX_CTL Register

Address: 20E_0000h base + 14h offset = 20E_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad BOOT_MODE0 0 **DISABLED** — Input Path is determined by functionality
-	This field is reserved. Reserved

30.5.2 SW_MUX_CTL_PAD_BOOT_MODE1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1)

SW_MUX_CTL Register

Address: 20E_0000h base + 18h offset = 20E_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad BOOT_MODE1 0 **DISABLED** — Input Path is determined by functionality
-	This field is reserved. Reserved

30.5.3 SW_MUX_CTL_PAD_SNVS_TAMPER0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER0)

SW_MUX_CTL Register

Address: 20E_0000h base + 1Ch offset = 20E_001Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER0 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field **NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER0 ==> GPIO5_00 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO00 of instance - gpio5 Other Reserved

30.5.4 SW_MUX_CTL_PAD_SNVS_TAMPER1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER1)

SW_MUX_CTL Register

Address: 20E_0000h base + 20h offset = 20E_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER1 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	**NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER1 ==> GPIO5_01 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO01 of instance - gpio5 Other Reserved

30.5.5 SW_MUX_CTL_PAD_SNVS_TAMPER2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER2)

SW_MUX_CTL Register

Address: 20E_0000h base + 24h offset = 20E_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER2 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	**NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER2 ==> GPIO5_02 101 **ALT5** — Slect mux mode: ALT5 mux port, GPIO5_IO02 of instance - gpio5 Other Reserved

30.5.6 SW_MUX_CTL_PAD_SNVS_TAMPER3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER3)

SW_MUX_CTL Register

Address: 20E_0000h base + 28h offset = 20E_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER3 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	Mux Mode Select Field **NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER3 ==> GPIO5_03 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO03 of instance - gpio Other Reserved

30.5.7 SW_MUX_CTL_PAD_SNVS_TAMPER4 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER4)

SW_MUX_CTL Register

Address: 20E_0000h base + 2Ch offset = 20E_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER4 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER4 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	**NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER4 ==> GPIO5_04 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO04 of instance - gpio5 Other Reserved

30.5.8 SW_MUX_CTL_PAD_SNVS_TAMPER5 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER5)

SW_MUX_CTL Register

Address: 20E_0000h base + 30h offset = 20E_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER5 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER5 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field **NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER5 ==> GPIO5_05 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO05 of instance - gpio5 Other Reserved

30.5.9 SW_MUX_CTL_PAD_SNVS_TAMPER6 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER6)

SW_MUX_CTL Register

Address: 20E_0000h base + 34h offset = 20E_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER6 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER6 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	**NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER6 ==> GPIO5_06 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO06 of instance - gpio5 Other Reserved

30.5.10 SW_MUX_CTL_PAD_SNVS_TAMPER7 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER7)

SW_MUX_CTL Register

Address: 20E_0000h base + 38h offset = 20E_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER7 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER7 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	**NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER7 ==> GPIO5_07 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO07 of instance - gpio5 Other Reserved

30.5.11 SW_MUX_CTL_PAD_SNVS_TAMPER8 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER8)

SW_MUX_CTL Register

Address: 20E_0000h base + 3Ch offset = 20E_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER8 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER8 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	**NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER8 ==> GPIO5_08 101 **ALT5** — Select mux moe: ALT5 mux port, GPIO05_IO08 of instance - gpio5 Other Reserved

30.5.12 SW_MUX_CTL_PAD_SNVS_TAMPER9 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER9)

SW_MUX_CTL Register

Address: 20E_0000h base + 40h offset = 20E_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

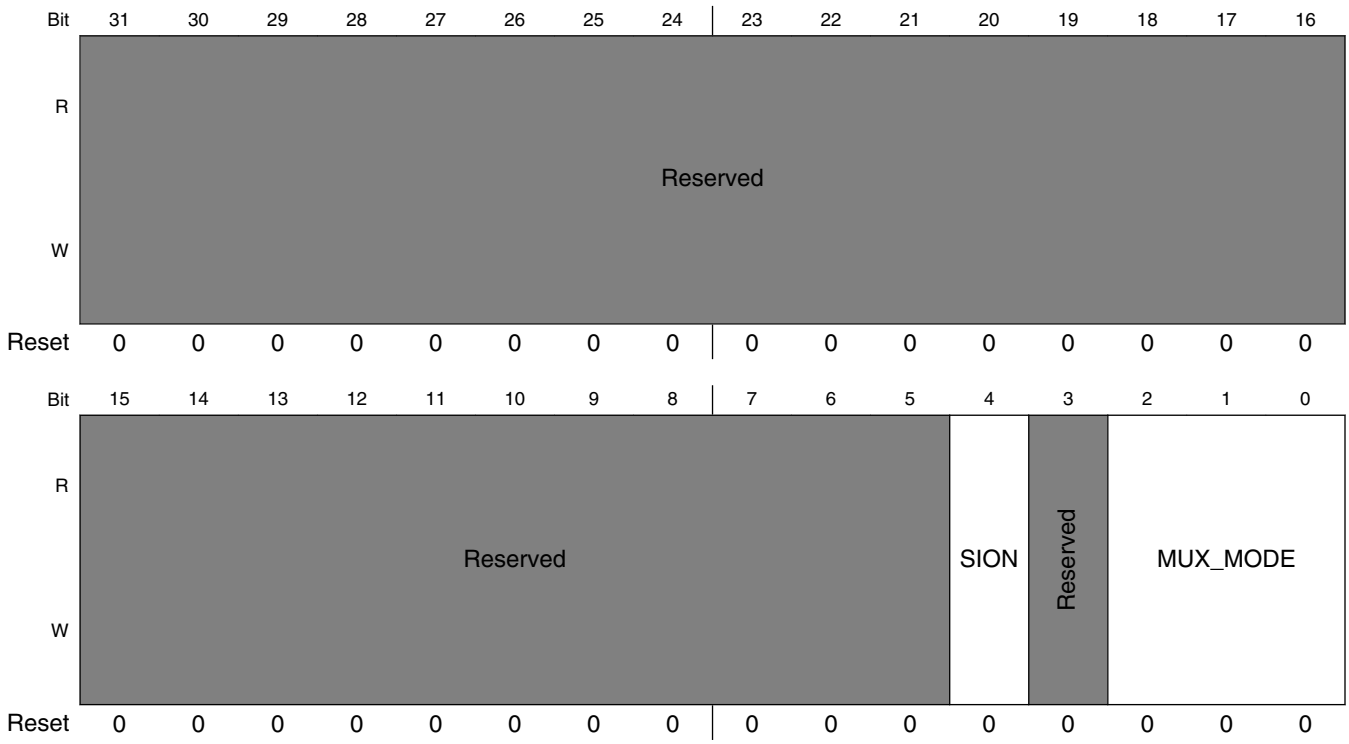
IOMUXC_SW_MUX_CTL_PAD_SNVS_TAMPER9 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SNVS_TAMPER9 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	**NOTE:** ALT5 mode is only valid when TAMPER PIN is used as GPIO. This depends on FUSE setting "TAMPER_PIN_DISABLE[1:0]". Following is the mux information when TAMPER PIN is used as GPIO: SNVS_TAMPER9 ==> GPIO5_09 101 **ALT5** — Select mux mode: ALT5 mux port, GPIO5_IO09 of instance - gpio9 Other Reserved

30.5.13 SW_MUX_CTL_PAD_JTAG_MOD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_MOD)

SW_MUX_CTL Register

Address: 20E_0000h base + 44h offset = 20E_0044h



IOMUXC_SW_MUX_CTL_PAD_JTAG_MOD field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad JTAG_MOD 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: JTAG_MOD. 000 **ALT0** — Select mux mode: ALT0 mux port: SJC_MOD of instance: sjc 001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CLK of instance: gpt2 010 **ALT2** — Select mux mode: ALT2 mux port: SPDIF_OUT of instance: spdif

Table continues on the next page...

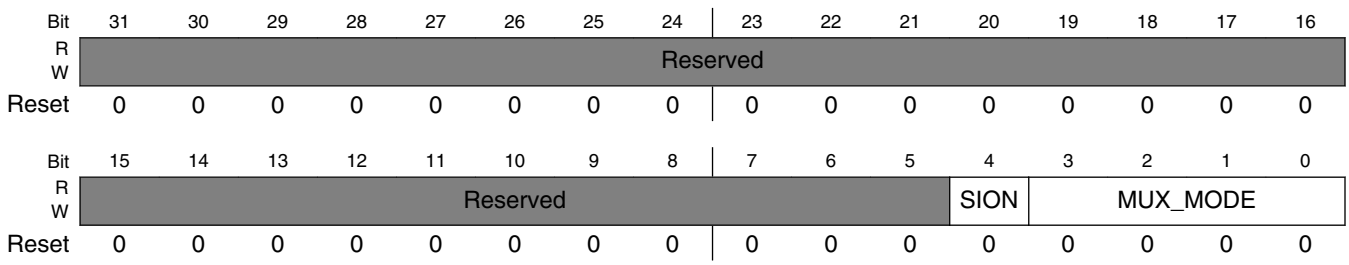
IOMUXC_SW_MUX_CTL_PAD_JTAG_MOD field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: ENET1_REF_CLK_25M of instance: enet1 100 **ALT4** — Select mux mode: ALT4 mux port: CCM_PMIC_RDY of instance: ccm 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO10 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: SDMA_EXT_EVENT00 of instance: sdma

30.5.14 SW_MUX_CTL_PAD_JTAG_TMS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TMS)

SW_MUX_CTL Register

Address: 20E_0000h base + 48h offset = 20E_0048h



IOMUXC_SW_MUX_CTL_PAD_JTAG_TMS field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad JTAG_TMS 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: JTAG_TMS. 0000 **ALT0** — Select mux mode: ALT0 mux port: SJC_TMS of instance: sjc 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CAPTURE1 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_MCLK of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: CCM_CLKO1 of instance: ccm 0100 **ALT4** — Select mux mode: ALT4 mux port: CCM_WAIT of instance: ccm 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO11 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: SDMA_EXT_EVENT01 of instance: sdma 1000 **ALT8** — Select mux mode: ALT8 mux port: EPIT1_OUT of instance: epit1

30.5.15 SW_MUX_CTL_PAD_JTAG_TDO SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TDO)

SW_MUX_CTL Register

Address: 20E_0000h base + 4Ch offset = 20E_004Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_JTAG_TDO field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad JTAG_TDO 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: JTAG_TDO. 0000 **ALT0** — Select mux mode: ALT0 mux port: SJC_TDO of instance: sjc 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CAPTURE2 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_SYNC of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: CCM_CLKO2 of instance: ccm 0100 **ALT4** — Select mux mode: ALT4 mux port: CCM_STOP of instance: ccm 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO12 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: MQS_RIGHT of instance: mqs 1000 **ALT8** — Select mux mode: ALT8 mux port: EPIT2_OUT of instance: epit2

30.5.16 SW_MUX_CTL_PAD_JTAG_TDI SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TDI)

SW_MUX_CTL Register

Address: 20E_0000h base + 50h offset = 20E_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_JTAG_TDI field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad JTAG_TDI 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: JTAG_TDI. 0000 **ALT0** — Select mux mode: ALT0 mux port: SJC_TDI of instance: sjc 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE1 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_BCLK of instance: sai2 0100 **ALT4** — Select mux mode: ALT4 mux port: PWM6_OUT of instance: pwm6 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO13 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: MQS_LEFT of instance: mqs 1000 **ALT8** — Select mux mode: ALT8 mux port: SIM1_POWER_FAIL of instance: sim1

30.5.17 SW_MUX_CTL_PAD_JTAG_TCK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TCK)

SW_MUX_CTL Register

Address: 20E_0000h base + 54h offset = 20E_0054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_JTAG_TCK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad JTAG_TCK 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: JTAG_TCK. 0000 **ALT0** — Select mux mode: ALT0 mux port: SJC_TCK of instance: sjc 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE2 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_RX_DATA of instance: sai2 0100 **ALT4** — Select mux mode: ALT4 mux port: PWM7_OUT of instance: pwm7 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO14 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: OSC32K_32K_OUT of instance: osc32k 1000 **ALT8** — Select mux mode: ALT8 mux port: SIM2_POWER_FAIL of instance: sim2

30.5.18 SW_MUX_CTL_PAD_JTAG_TRST_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_JTAG_TRST_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 58h offset = 20E_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_JTAG_TRST_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad JTAG_TRST_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: JTAG_TRST_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: SJC_TRSTB of instance: sjc 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE3 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_DATA of instance: sai2 0100 **ALT4** — Select mux mode: ALT4 mux port: PWM8_OUT of instance: pwm8 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO15 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: ANATOP_24M_OUT of instance: anatop 1000 **ALT8** — Select mux mode: ALT8 mux port: CAAM_RNG_OSC_OBS of instance: caam

30.5.19 SW_MUX_CTL_PAD_GPIO1_IO00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO00)

SW_MUX_CTL Register

Address: 20E_0000h base + 5Ch offset = 20E_005Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO00. 0000 **ALT0** — Select mux mode: ALT0 mux port: I2C2_SCL of instance: i2c2 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_CAPTURE1 of instance: gpt1 0010 **ALT2** — Select mux mode: ALT2 mux port: ANATOP_OTG1_ID of instance: anatop 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET1_REF_CLK1 of instance: enet1 0100 **ALT4** — Select mux mode: ALT4 mux port: MQS_RIGHT of instance: mqs 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO00 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: ENET1_1588_EVENT0_IN of instance: enet1 0111 **ALT7** — Select mux mode: ALT7 mux port: SRC_SYSTEM_RESET of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: WDOG3_WDOG_B of instance: wdog3

30.5.20 SW_MUX_CTL_PAD_GPIO1_IO01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO01)

SW_MUX_CTL Register

Address: 20E_0000h base + 60h offset = 20E_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO01. 0000 **ALT0** — Select mux mode: ALT0 mux port: I2C2_SDA of instance: i2c2 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_COMPARE1 of instance: gpt1 0010 **ALT2** — Select mux mode: ALT2 mux port: USB_OTG1_OC of instance: usb 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET2_REF_CLK2 of instance: enet2 0100 **ALT4** — Select mux mode: ALT4 mux port: MQS_LEFT of instance: mqs 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO01 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: ENET1_1588_EVENT0_OUT of instance: enet1 0111 **ALT7** — Select mux mode: ALT7 mux port: SRC_EARLY_RESET of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: WDOG1_WDOG_B of instance: wdog1

30.5.21 SW_MUX_CTL_PAD_GPIO1_IO02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO02)

SW_MUX_CTL Register

Address: 20E_0000h base + 64h offset = 20E_0064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO02. 0000 **ALT0** — Select mux mode: ALT0 mux port: I2C1_SCL of instance: i2c1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_COMPARE2 of instance: gpt1 0010 **ALT2** — Select mux mode: ALT2 mux port: USB_OTG2_PWR of instance: usb 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET1_REF_CLK_25M of instance: enet1 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC1_WP of instance: usdhc1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO02 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: SDMA_EXT_EVENT00 of instance: sdma 0111 **ALT7** — Select mux mode: ALT7 mux port: SRC_ANY_PU_RESET of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: UART1_TX of instance: uart1

30.5.22 SW_MUX_CTL_PAD_GPIO1_IO03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO03)

SW_MUX_CTL Register

Address: 20E_0000h base + 68h offset = 20E_0068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO03. 0000 **ALT0** — Select mux mode: ALT0 mux port: I2C1_SDA of instance: i2c1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_COMPARE3 of instance: gpt1 0010 **ALT2** — Select mux mode: ALT2 mux port: USB_OTG2_OC of instance: usb 0011 **ALT3** — Select mux mode: ALT3 mux port: OSC32K_32K_OUT of instance: osc32k 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC1_CD_B of instance: usdhc1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO03 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: CCM_DI0_EXT_CLK of instance: ccm 0111 **ALT7** — Select mux mode: ALT7 mux port: SRC_TESTER_ACK of instance: src **NOTE:** ALT7 mode will be automatically active when system reset. The PAD setting will be 100 K pull down and input enable during reset period. Once system reset is completed, the state of GPIO1_IO03 will be output keeper and input enable. 1000 **ALT8** — Select mux mode: ALT8 mux port: UART1_RX of instance: uart1

30.5.23 SW_MUX_CTL_PAD_GPIO1_IO04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO04)

SW_MUX_CTL Register

Address: 20E_0000h base + 6Ch offset = 20E_006Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO04. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_REF_CLK1 of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM3_OUT of instance: pwm3 0010 **ALT2** — Select mux mode: ALT2 mux port: USB_OTG1_PWR of instance: usb 0011 **ALT3** — Select mux mode: ALT3 mux port: ANATOP_24M_OUT of instance: anatop 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC1_RESET_B of instance: usdhc1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO04 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: ENET2_1588_EVENT0_IN of instance: enet2 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_TX of instance: uart5

30.5.24 SW_MUX_CTL_PAD_GPIO1_IO05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO05)

SW_MUX_CTL Register

Address: 20E_0000h base + 70h offset = 20E_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO05. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_REF_CLK2 of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM4_OUT of instance: pwm4 0010 **ALT2** — Select mux mode: ALT2 mux port: ANATOP_OTG2_ID of instance: anatop 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_FIELD of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC1_VSELECT of instance: usdhc1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO05 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: ENET2_1588_EVENT0_OUT of instance: enet2 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_RX of instance: uart5

30.5.25 SW_MUX_CTL_PAD_GPIO1_IO06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO06)

SW_MUX_CTL Register

Address: 20E_0000h base + 74h offset = 20E_0074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO06. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_MDIO of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_MDIO of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: USB_OTG_PWR_WAKE of instance: usb 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_MCLK of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC2_WP of instance: usdhc2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO06 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: CCM_WAIT of instance: ccm 0111 **ALT7** — Select mux mode: ALT7 mux port: CCM_REF_EN_B of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: UART1_CTS_B of instance: uart1

30.5.26 SW_MUX_CTL_PAD_GPIO1_IO07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO07)

SW_MUX_CTL Register

Address: 20E_0000h base + 78h offset = 20E_0078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO07 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO07. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_MDC of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_MDC of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: USB_OTG_HOST_MODE of instance: usb 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_PIXCLK of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC2_CD_B of instance: usdhc2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO07 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: CCM_STOP of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: UART1_RTS_B of instance: uart1

30.5.27 SW_MUX_CTL_PAD_GPIO1_IO08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO08)

SW_MUX_CTL Register

Address: 20E_0000h base + 7Ch offset = 20E_007Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO08 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO08. 0000 **ALT0** — Select mux mode: ALT0 mux port: PWM1_OUT of instance: pwm1 0001 **ALT1** — Select mux mode: ALT1 mux port: WDOG1_WDOG_B of instance: wdog1 0010 **ALT2** — Select mux mode: ALT2 mux port: SPDIF_OUT of instance: spdif 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_VSYNC of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC2_VSELECT of instance: usdhc2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO08 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: CCM_PMIC_RDY of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_RTS_B of instance: uart5

30.5.28 SW_MUX_CTL_PAD_GPIO1_IO09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09)

SW_MUX_CTL Register

Address: 20E_0000h base + 80h offset = 20E_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: GPIO1_IO09. 0000 **ALT0** — Select mux mode: ALT0 mux port: PWM2_OUT of instance: pwm2 0001 **ALT1** — Select mux mode: ALT1 mux port: WDOG1_WDOG_ANY of instance: wdog1 0010 **ALT2** — Select mux mode: ALT2 mux port: SPDIF_IN of instance: spdif 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_HSYNC of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: USDHC2_RESET_B of instance: usdhc2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO09 of instance: gpio1 0110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_RESET_B of instance: usdhc1 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_CTS_B of instance: uart5

30.5.29 SW_MUX_CTL_PAD_UART1_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + 84h offset = 20E_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART1_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART1_TX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART1_TX of instance: uart1 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_RDATA02 of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C3_SCL of instance: i2c3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA02 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_COMPARE1 of instance: gpt1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO16 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: SPDIF_OUT of instance: spdif

30.5.30 SW_MUX_CTL_PAD_UART1_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + 88h offset = 20E_0088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART1_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART1_RX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART1_RX of instance: uart1 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_RDATA03 of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C3_SDA of instance: i2c3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA03 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_CLK of instance: gpt1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO17 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: SPDIF_IN of instance: spdif

30.5.31 SW_MUX_CTL_PAD_UART1_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_CTS_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 8Ch offset = 20E_008Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART1_CTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART1_CTS_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART1_CTS_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART1_CTS_B of instance: uart1 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_RX_CLK of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: USDHC1_WP of instance: usdhc1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA04 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET2_1588_EVENT1_IN of instance: enet2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO18 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_WP of instance: usdhc2

30.5.32 SW_MUX_CTL_PAD_UART1_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RTS_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 90h offset = 20E_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART1_RTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART1_RTS_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART1_RTS_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART1_RTS_B of instance: uart1 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_TX_ER of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: USDHC1_CD_B of instance: usdhc1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA05 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET2_1588_EVENT1_OUT of instance: enet2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO19 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_CD_B of instance: usdhc2

30.5.33 SW_MUX_CTL_PAD_UART2_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + 94h offset = 20E_0094h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART2_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART2_TX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART2_TX of instance: uart2 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_TDATA02 of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C4_SCL of instance: i2c4 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA06 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_CAPTURE1 of instance: gpt1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO20 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_SS0 of instance: ecspi3

30.5.34 SW_MUX_CTL_PAD_UART2_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + 98h offset = 20E_0098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART2_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART2_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART2_RX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART2_RX of instance: uart2 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_TDATA03 of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C4_SDA of instance: i2c4 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA07 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_CAPTURE2 of instance: gpt1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO21 of instance: gpio1 0111 **ALT7** — Select mux mode: ALT7 mux port: SJC_DONE of instance: sjc 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_SCLK of instance: ecspi3

30.5.35 SW_MUX_CTL_PAD_UART2_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_CTS_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 9Ch offset = 20E_009Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART2_CTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART2_CTS_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART2_CTS_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART2_CTS_B of instance: uart2 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_CRS of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN2_TX of instance: flexcan2 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA08 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_COMPARE2 of instance: gpt1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO22 of instance: gpio1 0111 **ALT7** — Select mux mode: ALT7 mux port: SJC_DE_B of instance: sjc 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_MOSI of instance: ecspi3

30.5.36 SW_MUX_CTL_PAD_UART2_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RTS_B)

SW_MUX_CTL Register

Address: 20E_0000h base + A0h offset = 20E_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART2_RTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART2_RTS_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART2_RTS_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART2_RTS_B of instance: uart2 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET1_COL of instance: enet1 0010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN2_RX of instance: flexcan2 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA09 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_COMPARE3 of instance: gpt1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO23 of instance: gpio1 0111 **ALT7** — Select mux mode: ALT7 mux port: SJC_FAIL of instance: sjc 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_MISO of instance: ecspi3

30.5.37 SW_MUX_CTL_PAD_UART3_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_TX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + A4h offset = 20E_00A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART3_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART3_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART3_TX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART3_TX of instance: uart3 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_RDATA02 of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT0_PD of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA01 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: UART2_CTS_B of instance: uart2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO24 of instance: gpio1 0111 **ALT7** — Select mux mode: ALT7 mux port: SJC_JTAG_ACT of instance: sjc **NOTE:** ALT7 mode will be automatically active (output SJC.SJC_JTAG_ACT) when system reset. Once system reset is completed, the state of UART3_TX_DATA will be output keeper and input enenable. 1000 **ALT8** — Select mux mode: ALT8 mux port: ANATOP_OTG1_ID of instance: anatop

30.5.38 SW_MUX_CTL_PAD_UART3_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + A8h offset = 20E_00A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART3_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART3_RX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART3_RX of instance: uart3 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_RDATA03 of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT0_PD of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA00 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: UART2_RTS_B of instance: uart2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO25 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: EPIT1_OUT of instance: epit1

30.5.39 SW_MUX_CTL_PAD_UART3_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B)

SW_MUX_CTL Register

Address: 20E_0000h base + ACh offset = 20E_00ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART3_CTS_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART3_CTS_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART3_CTS_B of instance: uart3 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_RX_CLK of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN1_TX of instance: flexcan1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA10 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET1_1588_EVENT1_IN of instance: enet1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO26 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: EPIT2_OUT of instance: epit2

30.5.40 SW_MUX_CTL_PAD_UART3_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B)

SW_MUX_CTL Register

Address: 20E_0000h base + B0h offset = 20E_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART3_RTS_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART3_RTS_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART3_RTS_B of instance: uart3 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_TX_ER of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN1_RX of instance: flexcan1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA11 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET1_1588_EVENT1_OUT of instance: enet1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO27 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: WDOG1_WDOG_B of instance: wdog1

30.5.41 SW_MUX_CTL_PAD_UART4_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART4_TX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + B4h offset = 20E_00B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART4_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART4_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART4_TX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART4_TX of instance: uart4 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_TDATA02 of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C1_SCL of instance: i2c1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA12 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: CSU_CSU_ALARM_AUT02 of instance: csu 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO28 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_SCLK of instance: ecspi2

30.5.42 SW_MUX_CTL_PAD_UART4_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART4_RX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + B8h offset = 20E_00B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART4_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART4_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART4_RX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART4_RX of instance: uart4 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_TDATA03 of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C1_SDA of instance: i2c1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA13 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: CSU_CSU_ALARM_AUT01 of instance: csu 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO29 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_SS0 of instance: ecspi2

30.5.43 SW_MUX_CTL_PAD_UART5_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART5_TX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + BCh offset = 20E_00BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART5_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART5_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART5_TX_DATA. 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO30 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_MOSI of instance: ecspi2 0000 **ALT0** — Select mux mode: ALT0 mux port: UART5_TX of instance: uart5 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_CRS of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C2_SCL of instance: i2c2 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA14 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: CSU_CSU_ALARM_AUT00 of instance: csu

30.5.44 SW_MUX_CTL_PAD_UART5_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART5_RX_DATA)

SW_MUX_CTL Register

Address: 20E_0000h base + C0h offset = 20E_00C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_UART5_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART5_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: UART5_RX_DATA. 0000 **ALT0** — Select mux mode: ALT0 mux port: UART5_RX of instance: uart5 0001 **ALT1** — Select mux mode: ALT1 mux port: ENET2_COL of instance: enet2 0010 **ALT2** — Select mux mode: ALT2 mux port: I2C2_SDA of instance: i2c2 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA15 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: CSU_CSU_INT_DEB of instance: csu 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO31 of instance: gpio1 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_MISO of instance: ecspi2

30.5.45 SW_MUX_CTL_PAD_ENET1_RX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA0)

SW_MUX_CTL Register

Address: 20E_0000h base + C4h offset = 20E_00C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RX_DATA0 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_RX_DATA0. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_RDATA00 of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART4_RTS_B of instance: uart4 0010 **ALT2** — Select mux mode: ALT2 mux port: PWM1_OUT of instance: pwm1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA16 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: FLEXCAN1_TX of instance: flexcan1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO00 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW00 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC1_LCTL of instance: usdhc1

30.5.46 SW_MUX_CTL_PAD_ENET1_RX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA1)

SW_MUX_CTL Register

Address: 20E_0000h base + C8h offset = 20E_00C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RX_DATA1 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_RX_DATA1. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_RDATA01 of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART4_CTS_B of instance: uart4 0010 **ALT2** — Select mux mode: ALT2 mux port: PWM2_OUT of instance: pwm2 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA17 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: FLEXCAN1_RX of instance: flexcan1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO01 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL00 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_LCTL of instance: usdhc2

30.5.47 SW_MUX_CTL_PAD_ENET1_RX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_EN)

SW_MUX_CTL Register

Address: 20E_0000h base + CCh offset = 20E_00CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_EN field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RX_EN 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_RX_EN. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_RX_EN of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART5_RTS_B of instance: uart5 0010 **ALT2** — Select mux mode: ALT2 mux port: OSC32K_32K_OUT of instance: osc32k 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA18 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: FLEXCAN2_TX of instance: flexcan2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO02 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW01 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC1_VSELECT of instance: usdhc1

30.5.48 SW_MUX_CTL_PAD_ENET1_TX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA0)

SW_MUX_CTL Register

Address: 20E_0000h base + D0h offset = 20E_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_TX_DATA0 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_TX_DATA0. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_TDATA00 of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART5_CTS_B of instance: uart5 0010 **ALT2** — Select mux mode: ALT2 mux port: ANATOP_24M_OUT of instance: anatop 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA19 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: FLEXCAN2_RX of instance: flexcan2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO03 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL01 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_VSELECT of instance: usdhc2

30.5.49 SW_MUX_CTL_PAD_ENET1_TX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA1)

SW_MUX_CTL Register

Address: 20E_0000h base + D4h offset = 20E_00D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_TX_DATA1 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_TX_DATA1. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_TDATA01 of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART6_CTS_B of instance: uart6 0010 **ALT2** — Select mux mode: ALT2 mux port: PWM5_OUT of instance: pwm5 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA20 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET2_MDIO of instance: enet2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO04 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW02 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: WDOG1_WDOG_RST_B_DEB of instance: wdog1

30.5.50 SW_MUX_CTL_PAD_ENET1_TX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_EN)

SW_MUX_CTL Register

Address: 20E_0000h base + D8h offset = 20E_00D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_EN field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_TX_EN 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_TX_EN. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_TX_EN of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART6_RTS_B of instance: uart6 0010 **ALT2** — Select mux mode: ALT2 mux port: PWM6_OUT of instance: pwm6 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA21 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET2_MDC of instance: enet2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO05 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL02 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: WDOG2_WDOG_RST_B_DEB of instance: wdog2

30.5.51 SW_MUX_CTL_PAD_ENET1_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_CLK)

SW_MUX_CTL Register

Address: 20E_0000h base + DCh offset = 20E_00DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_TX_CLK 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_TX_CLK. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_TX_CLK of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_CTS_B of instance: uart7 0010 **ALT2** — Select mux mode: ALT2 mux port: PWM7_OUT of instance: pwm7 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA22 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET1_REF_CLK1 of instance: enet1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO06 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW03 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_CLK of instance: gpt1

30.5.52 SW_MUX_CTL_PAD_ENET1_RX_ER SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_ER)

SW_MUX_CTL Register

Address: 20E_0000h base + E0h offset = 20E_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_ER field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RX_ER 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET1_RX_ER. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET1_RX_ER of instance: enet1 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_RTS_B of instance: uart7 0010 **ALT2** — Select mux mode: ALT2 mux port: PWM8_OUT of instance: pwm8 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA23 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_CRE of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO07 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL03 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_CAPTURE2 of instance: gpt1

30.5.53 SW_MUX_CTL_PAD_ENET2_RX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA0)

SW_MUX_CTL Register

Address: 20E_0000h base + E4h offset = 20E_00E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_RX_DATA0 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_RX_DATA0. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_RDATA00 of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART6_TX of instance: uart6 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT0_TRXD of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C3_SCL of instance: i2c3 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET1_MDIO of instance: enet1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO08 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW04 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG1_PWR of instance: usb

30.5.54 SW_MUX_CTL_PAD_ENET2_RX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA1)

SW_MUX_CTL Register

Address: 20E_0000h base + E8h offset = 20E_00E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_RX_DATA1 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_RX_DATA1. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_RDATA01 of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART6_RX of instance: uart6 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT0_CLK of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C3_SDA of instance: i2c3 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET1_MDC of instance: enet1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO09 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL04 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG1_OC of instance: usb

30.5.55 SW_MUX_CTL_PAD_ENET2_RX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_EN)

SW_MUX_CTL Register

Address: 20E_0000h base + ECh offset = 20E_00ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_EN field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_RX_EN 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_RX_EN. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_RX_EN of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_TX of instance: uart7 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT0_RST_B of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C4_SCL of instance: i2c4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR26 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO10 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW05 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET1_REF_CLK_25M of instance: enet1

30.5.56 SW_MUX_CTL_PAD_ENET2_TX_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA0)

SW_MUX_CTL Register

Address: 20E_0000h base + F0h offset = 20E_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_TX_DATA0 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_TX_DATA0. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_TDATA00 of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_RX of instance: uart7 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT0_SVEN of instance: sim1 anatop 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C4_SDA of instance: i2c4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_EB_B02 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO11 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL05 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ANATOP_24M_OUT of instance: anatop

30.5.57 SW_MUX_CTL_PAD_ENET2_TX_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA1)

SW_MUX_CTL Register

Address: 20E_0000h base + F4h offset = 20E_00F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_TX_DATA1 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_TX_DATA1. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_TDATA01 of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_TX of instance: uart8 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT0_TRXD of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_SCLK of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_EB_B03 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO12 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW06 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG2_PWR of instance: usb

30.5.58 SW_MUX_CTL_PAD_ENET2_TX_EN SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_EN)

SW_MUX_CTL Register

Address: 20E_0000h base + F8h offset = 20E_00F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_EN field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_TX_EN 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_TX_EN. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_TX_EN of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_RX of instance: uart8 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT0_CLK of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_MOSI of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ACLK_FREERUN of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO13 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL06 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG2_OC of instance: usb

30.5.59 SW_MUX_CTL_PAD_ENET2_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_CLK)

SW_MUX_CTL Register

Address: 20E_0000h base + FCh offset = 20E_00FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_TX_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_TX_CLK 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_TX_CLK. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_TX_CLK of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_CTS_B of instance: uart8 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT0_RST_B of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_MISO of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: ENET2_REF_CLK2 of instance: enet2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO14 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_ROW07 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ANATOP_OTG2_ID of instance: anatop

30.5.60 SW_MUX_CTL_PAD_ENET2_RX_ER SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_ER)

SW_MUX_CTL Register

Address: 20E_0000h base + 100h offset = 20E_0100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_ENET2_RX_ER field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET2_RX_ER 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: ENET2_RX_ER. 0000 **ALT0** — Select mux mode: ALT0 mux port: ENET2_RX_ER of instance: enet2 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_RTS_B of instance: uart8 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT0_SVEN of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_SS0 of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR25 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO15 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: KPP_COL07 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: WDOG1_WDOG_ANY of instance: wdog1

30.5.61 SW_MUX_CTL_PAD_LCD_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_CLK)

SW_MUX_CTL Register

Address: 20E_0000h base + 104h offset = 20E_0104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_CLK 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_CLK. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_CLK of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: LCDIF_WR_RWN of instance: lcdif 0010 **ALT2** — Select mux mode: ALT2 mux port: UART4_TX of instance: uart4 0011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_MCLK of instance: sai3 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_CS2_B of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO00 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: WDOG1_WDOG_RST_B_DEB of instance: wdog1

30.5.62 SW_MUX_CTL_PAD_LCD_ENABLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_ENABLE)

SW_MUX_CTL Register

Address: 20E_0000h base + 108h offset = 20E_0108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_ENABLE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_ENABLE 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_ENABLE. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_ENABLE of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: LCDIF_RD_E of instance: lcdif 0010 **ALT2** — Select mux mode: ALT2 mux port: UART4_RX of instance: uart4 0011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_TX_SYNC of instance: sai3 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_CS3_B of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO01 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_RDY of instance: ecspi2

30.5.63 SW_MUX_CTL_PAD_LCD_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC)

SW_MUX_CTL Register

Address: 20E_0000h base + 10Ch offset = 20E_010Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_HSYNC 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_HSYNC. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_HSYNC of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: LCDIF_RS of instance: lcdif 0010 **ALT2** — Select mux mode: ALT2 mux port: UART4_CTS_B of instance: uart4 0011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_TX_BCLK of instance: sai3 0100 **ALT4** — Select mux mode: ALT4 mux port: WDOG3_WDOG_RST_B_DEB of instance: wdog3 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO02 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_SS1 of instance: ecspi2

30.5.64 SW_MUX_CTL_PAD_LCD_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC)

SW_MUX_CTL Register

Address: 20E_0000h base + 110h offset = 20E_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_VSYNC 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_VSYNC. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_VSYNC of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: LCDIF_BUSY of instance: lcdif 0010 **ALT2** — Select mux mode: ALT2 mux port: UART4_RTS_B of instance: uart4 0011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_RX_DATA of instance: sai3 0100 **ALT4** — Select mux mode: ALT4 mux port: WDOG2_WDOG_B of instance: wdog2 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO03 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_SS2 of instance: ecspi2

30.5.65 SW_MUX_CTL_PAD_LCD_RESET SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_RESET)

SW_MUX_CTL Register

Address: 20E_0000h base + 114h offset = 20E_0114h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_RESET field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_RESET 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_RESET. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_RESET of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: LCDIF_CS of instance: lcdif 0010 **ALT2** — Select mux mode: ALT2 mux port: CA7_MX6UL_EVENTI of instance: ca7_mx6ul 0011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_TX_DATA of instance: sai3 0100 **ALT4** — Select mux mode: ALT4 mux port: WDOG1_WDOG_ANY of instance: wdog1 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO04 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI2_SS3 of instance: ecspi2

30.5.66 SW_MUX_CTL_PAD_LCD_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00)

SW_MUX_CTL Register

Address: 20E_0000h base + 118h offset = 20E_0118h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA00. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA00 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM1_OUT of instance: pwm1 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET1_1588_EVENT2_IN of instance: enet1 0100 **ALT4** — Select mux mode: ALT4 mux port: I2C3_SDA of instance: i2c3 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO05 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG00 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI1_MCLK of instance: sai1

30.5.67 SW_MUX_CTL_PAD_LCD_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01)

SW_MUX_CTL Register

Address: 20E_0000h base + 11Ch offset = 20E_011Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA01. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA01 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM2_OUT of instance: pwm2 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET1_1588_EVENT2_OUT of instance: enet1 0100 **ALT4** — Select mux mode: ALT4 mux port: I2C3_SCL of instance: i2c3 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO06 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG01 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI1_TX_SYNC of instance: sai1

30.5.68 SW_MUX_CTL_PAD_LCD_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA02)

SW_MUX_CTL Register

Address: 20E_0000h base + 120h offset = 20E_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA02. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA02 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM3_OUT of instance: pwm3 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET1_1588_EVENT3_IN of instance: enet1 0100 **ALT4** — Select mux mode: ALT4 mux port: I2C4_SDA of instance: i2c4 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO07 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG02 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI1_TX_BCLK of instance: sai1

30.5.69 SW_MUX_CTL_PAD_LCD_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03)

SW_MUX_CTL Register

Address: 20E_0000h base + 124h offset = 20E_0124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA03. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA03 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM4_OUT of instance: pwm4 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET1_1588_EVENT3_OUT of instance: enet1 0100 **ALT4** — Select mux mode: ALT4 mux port: I2C4_SCL of instance: i2c4 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO08 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG03 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI1_RX_DATA of instance: sai1

30.5.70 SW_MUX_CTL_PAD_LCD_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04)

SW_MUX_CTL Register

Address: 20E_0000h base + 128h offset = 20E_0128h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA04. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA04 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_CTS_B of instance: uart8 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET2_1588_EVENT2_IN of instance: enet2 0100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_SR_CLK of instance: spdif 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO09 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG04 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI1_TX_DATA of instance: sai1

30.5.71 SW_MUX_CTL_PAD_LCD_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA05)

SW_MUX_CTL Register

Address: 20E_0000h base + 12Ch offset = 20E_012Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA05. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA05 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_RTS_B of instance: uart8 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET2_1588_EVENT2_OUT of instance: enet2 0100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_OUT of instance: spdif 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO10 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG05 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI1_SS1 of instance: ecspi1

30.5.72 SW_MUX_CTL_PAD_LCD_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06)

SW_MUX_CTL Register

Address: 20E_0000h base + 130h offset = 20E_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA06. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA06 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_CTS_B of instance: uart7 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET2_1588_EVENT3_IN of instance: enet2 0100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_LOCK of instance: spdif 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO11 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG06 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI1_SS2 of instance: ecspi1

30.5.73 SW_MUX_CTL_PAD_LCD_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07)

SW_MUX_CTL Register

Address: 20E_0000h base + 134h offset = 20E_0134h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA07 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA07. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA07 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_RTS_B of instance: uart7 0011 **ALT3** — Select mux mode: ALT3 mux port: ENET2_1588_EVENT3_OUT of instance: enet2 0100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_EXT_CLK of instance: spdif 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO12 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG07 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI1_SS3 of instance: ecspi1

30.5.74 SW_MUX_CTL_PAD_LCD_DATA08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08)

SW_MUX_CTL Register

Address: 20E_0000h base + 138h offset = 20E_0138h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA08 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA08. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA08 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SPDIF_IN of instance: spdif 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA16 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA00 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO13 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG08 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN1_TX of instance: flexcan1

30.5.75 SW_MUX_CTL_PAD_LCD_DATA09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA09)

SW_MUX_CTL Register

Address: 20E_0000h base + 13Ch offset = 20E_013Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA09. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA09 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_MCLK of instance: sai3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA17 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA01 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO14 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG09 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN1_RX of instance: flexcan1

30.5.76 SW_MUX_CTL_PAD_LCD_DATA10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10)

SW_MUX_CTL Register

Address: 20E_0000h base + 140h offset = 20E_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA10 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA10. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA10 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_RX_SYNC of instance: sai3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA18 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA02 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO15 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG10 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN2_TX of instance: flexcan2

30.5.77 SW_MUX_CTL_PAD_LCD_DATA11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11)

SW_MUX_CTL Register

Address: 20E_0000h base + 144h offset = 20E_0144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA11 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA11. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA11 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_RX_BCLK of instance: sai3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA19 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA03 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO16 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG11 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN2_RX of instance: flexcan2

30.5.78 SW_MUX_CTL_PAD_LCD_DATA12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12)

SW_MUX_CTL Register

Address: 20E_0000h base + 148h offset = 20E_0148h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA12 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA12. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA12 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_TX_SYNC of instance: sai3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA20 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA04 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO17 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG12 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI1_RDY of instance: ecspi1

30.5.79 SW_MUX_CTL_PAD_LCD_DATA13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA13)

SW_MUX_CTL Register

Address: 20E_0000h base + 14Ch offset = 20E_014Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA13 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA13. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA13 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_TX_BCLK of instance: sai3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA21 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA05 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO18 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG13 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_RESET_B of instance: usdhc2

30.5.80 SW_MUX_CTL_PAD_LCD_DATA14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14)

SW_MUX_CTL Register

Address: 20E_0000h base + 150h offset = 20E_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA14 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA14. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA14 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_RX_DATA of instance: sai3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA22 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA06 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO19 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG14 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA4 of instance: usdhc2

30.5.81 SW_MUX_CTL_PAD_LCD_DATA15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15)

SW_MUX_CTL Register

Address: 20E_0000h base + 154h offset = 20E_0154h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA15 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA15. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA15 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_TX_DATA of instance: sai3 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA23 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA07 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO20 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG15 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA5 of instance: usdhc2

30.5.82 SW_MUX_CTL_PAD_LCD_DATA16 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16)

SW_MUX_CTL Register

Address: 20E_0000h base + 158h offset = 20E_0158h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA16 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA16. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA16 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_TX of instance: uart7 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA01 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA08 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO21 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG24 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA6 of instance: usdhc2

30.5.83 SW_MUX_CTL_PAD_LCD_DATA17 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA17)

SW_MUX_CTL Register

Address: 20E_0000h base + 15Ch offset = 20E_015Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA17 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA17 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA17. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA17 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART7_RX of instance: uart7 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA00 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA09 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO22 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG25 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA7 of instance: usdhc2

30.5.84 SW_MUX_CTL_PAD_LCD_DATA18 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18)

SW_MUX_CTL Register

Address: 20E_0000h base + 160h offset = 20E_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA18 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA18. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA18 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM5_OUT of instance: pwm5 0010 **ALT2** — Select mux mode: ALT2 mux port: CA7_MX6UL_EVENTO of instance: ca7_mx6ul 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA10 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA10 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO23 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG26 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_CMD of instance: usdhc2

30.5.85 SW_MUX_CTL_PAD_LCD_DATA19 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19)

SW_MUX_CTL Register

Address: 20E_0000h base + 164h offset = 20E_0164h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA19 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA19. 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA11 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO24 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG27 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_CLK of instance: usdhc2 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA19 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: PWM6_OUT of instance: pwm6 0010 **ALT2** — Select mux mode: ALT2 mux port: WDOG1_WDOG_ANY of instance: wdog1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA11 of instance: csi

30.5.86 SW_MUX_CTL_PAD_LCD_DATA20 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA20)

SW_MUX_CTL Register

Address: 20E_0000h base + 168h offset = 20E_0168h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA20 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA20 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA20. 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA12 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO25 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG28 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA0 of instance: usdhc2 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA20 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_TX of instance: uart8 0010 **ALT2** — Select mux mode: ALT2 mux port: ECSPI1_SCLK of instance: ecspi1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA12 of instance: csi

30.5.87 SW_MUX_CTL_PAD_LCD_DATA21 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21)

SW_MUX_CTL Register

Address: 20E_0000h base + 16Ch offset = 20E_016Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA21 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA21. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA21 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: UART8_RX of instance: uart8 0010 **ALT2** — Select mux mode: ALT2 mux port: ECSPI1_SS0 of instance: ecspi1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA13 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA13 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO26 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG29 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA1 of instance: usdhc2

30.5.88 SW_MUX_CTL_PAD_LCD_DATA22 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22)

SW_MUX_CTL Register

Address: 20E_0000h base + 170h offset = 20E_0170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA22 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA22. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA22 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: MQS_RIGHT of instance: mqs 0010 **ALT2** — Select mux mode: ALT2 mux port: ECSPI1_MOSI of instance: ecspi1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA14 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA14 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO27 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG30 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA2 of instance: usdhc2

30.5.89 SW_MUX_CTL_PAD_LCD_DATA23 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA23)

SW_MUX_CTL Register

Address: 20E_0000h base + 174h offset = 20E_0174h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA23 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA23 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: LCD_DATA23. 0000 **ALT0** — Select mux mode: ALT0 mux port: LCDIF_DATA23 of instance: lcdif 0001 **ALT1** — Select mux mode: ALT1 mux port: MQS_LEFT of instance: mqs 0010 **ALT2** — Select mux mode: ALT2 mux port: ECSPI1_MISO of instance: ecspi1 0011 **ALT3** — Select mux mode: ALT3 mux port: CSI_DATA15 of instance: csi 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DATA15 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO28 of instance: gpio3 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG31 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC2_DATA3 of instance: usdhc2

30.5.90 SW_MUX_CTL_PAD_NAND_RE_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_RE_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 178h offset = 20E_0178h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_RE_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_RE_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_RE_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_RE_B of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_CLK of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_SCLK of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: KPP_ROW00 of instance: kpp 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_EB_B00 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO00 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_SS2 of instance: ecspi3

30.5.91 SW_MUX_CTL_PAD_NAND_WE_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_WE_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 17Ch offset = 20E_017Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_WE_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_WE_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_WE_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_WE_B of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_CMD of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_SS0_B of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: KPP_COL00 of instance: kpp 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_EB_B01 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO01 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_SS3 of instance: ecspi3

30.5.92 SW_MUX_CTL_PAD_NAND_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA00)

SW_MUX_CTL Register

Address: 20E_0000h base + 180h offset = 20E_0180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA00. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA00 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA0 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_SS1_B of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: KPP_ROW01 of instance: kpp 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD08 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO02 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI4_RDY of instance: ecspi4

30.5.93 SW_MUX_CTL_PAD_NAND_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA01)

SW_MUX_CTL Register

Address: 20E_0000h base + 184h offset = 20E_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA01. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA01 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA1 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_DQS of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: KPP_COL01 of instance: kpp 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD09 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO03 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI4_SS1 of instance: ecspi4

30.5.94 SW_MUX_CTL_PAD_NAND_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA02)

SW_MUX_CTL Register

Address: 20E_0000h base + 188h offset = 20E_0188h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA02. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA02 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA2 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_DATA00 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: KPP_ROW02 of instance: kpp 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD10 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO04 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI4_SS2 of instance: ecspi4

30.5.95 SW_MUX_CTL_PAD_NAND_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA03)

SW_MUX_CTL Register

Address: 20E_0000h base + 18Ch offset = 20E_018Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA03. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA03 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA3 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_DATA01 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: KPP_COL02 of instance: kpp 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD11 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO05 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI4_SS3 of instance: ecspi4

30.5.96 SW_MUX_CTL_PAD_NAND_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA04)

SW_MUX_CTL Register

Address: 20E_0000h base + 190h offset = 20E_0190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA04. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA04 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA4 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_DATA02 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_SCLK of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD12 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO06 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART2_TX of instance: uart2

30.5.97 SW_MUX_CTL_PAD_NAND_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA05)

SW_MUX_CTL Register

Address: 20E_0000h base + 194h offset = 20E_0194h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA05. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA05 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA5 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_B_DATA03 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_MOSI of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD13 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO07 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART2_RX of instance: uart2

30.5.98 SW_MUX_CTL_PAD_NAND_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA06)

SW_MUX_CTL Register

Address: 20E_0000h base + 198h offset = 20E_0198h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA06. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA06 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA6 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_RX_BCLK of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_MISO of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD14 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO08 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART2_CTS_B of instance: uart2

30.5.99 SW_MUX_CTL_PAD_NAND_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DATA07)

SW_MUX_CTL Register

Address: 20E_0000h base + 19Ch offset = 20E_019Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DATA07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DATA07 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DATA07. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DATA07 of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA7 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_SS1_B of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI4_SS0 of instance: ecspi4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD15 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO09 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART2_RTS_B of instance: uart2

30.5.100 SW_MUX_CTL_PAD_NAND_ALE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_ALE)

SW_MUX_CTL Register

Address: 20E_0000h base + 1A0h offset = 20E_01A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_ALE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_ALE 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_ALE. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_ALE of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_RESET_B of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_DQS of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: PWM3_OUT of instance: pwm3 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR17 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO10 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_SS1 of instance: ecspi3

30.5.101 SW_MUX_CTL_PAD_NAND_WP_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_WP_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 1A4h offset = 20E_01A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_WP_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_WP_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_WP_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_WP_B of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC1_RESET_B of instance: usdhc1 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_SCLK of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: PWM4_OUT of instance: pwm4 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_BCLK of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO11 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ECSPI3_RDY of instance: ecspi3

30.5.102 SW_MUX_CTL_PAD_NAND_READY_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_READY_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 1A8h offset = 20E_01A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_READY_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_READY_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_READY_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_READY_B of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC1_DATA4 of instance: usdhc1 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_DATA00 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI3_SS0 of instance: ecspi3 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_CS1_B of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO12 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART3_TX of instance: uart3

30.5.103 SW_MUX_CTL_PAD_NAND_CE0_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CE0_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 1ACh offset = 20E_01ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_CE0_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_CE0_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_CE0_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_CE0_B of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC1_DATA5 of instance: usdhc1 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_DATA01 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI3_SCLK of instance: ecspi3 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_DTACK_B of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO13 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART3_RX of instance: uart3

30.5.104 SW_MUX_CTL_PAD_NAND_CE1_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CE1_B)

SW_MUX_CTL Register

Address: 20E_0000h base + 1B0h offset = 20E_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_CE1_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_CE1_B 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_CE1_B. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_CE1_B of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC1_DATA6 of instance: usdhc1 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_DATA02 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI3_MOSI of instance: ecspi3 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR18 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO14 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART3_CTS_B of instance: uart3

30.5.105 SW_MUX_CTL_PAD_NAND_CLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_CLE)

SW_MUX_CTL Register

Address: 20E_0000h base + 1B4h offset = 20E_01B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_CLE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_CLE 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_CLE. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_CLE of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC1_DATA7 of instance: usdhc1 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_DATA03 of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI3_MISO of instance: ecspi3 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR16 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO15 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: UART3_RTS_B of instance: uart3

30.5.106 SW_MUX_CTL_PAD_NAND_DQS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_NAND_DQS)

SW_MUX_CTL Register

Address: 20E_0000h base + 1B8h offset = 20E_01B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_NAND_DQS field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad NAND_DQS 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: NAND_DQS. 0000 **ALT0** — Select mux mode: ALT0 mux port: RAWNAND_DQS of instance: rawnand 0001 **ALT1** — Select mux mode: ALT1 mux port: CSI_FIELD of instance: csi 0010 **ALT2** — Select mux mode: ALT2 mux port: QSPI_A_SS0_B of instance: qspi 0011 **ALT3** — Select mux mode: ALT3 mux port: PWM5_OUT of instance: pwm5 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_WAIT of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO16 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SDMA_EXT_EVENT01 of instance: sdma 1000 **ALT8** — Select mux mode: ALT8 mux port: SPDIF_EXT_CLK of instance: spdif

30.5.107 SW_MUX_CTL_PAD_SD1_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CMD)

SW_MUX_CTL Register

Address: 20E_0000h base + 1BCh offset = 20E_01BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SD1_CMD field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_CMD 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: SD1_CMD. 0000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_CMD of instance: usdhc1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE1 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_RX_SYNC of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_OUT of instance: spdif 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR19 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO16 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: SDMA_EXT_EVENT00 of instance: sdma 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG1_PWR of instance: usb

30.5.108 SW_MUX_CTL_PAD_SD1_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CLK)

SW_MUX_CTL Register

Address: 20E_0000h base + 1C0h offset = 20E_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SD1_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_CLK 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: SD1_CLK. 0000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_CLK of instance: usdhc1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE2 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_MCLK of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_IN of instance: spdif 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR20 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO17 of instance: gpio2 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG1_OC of instance: usb

30.5.109 SW_MUX_CTL_PAD_SD1_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0)

SW_MUX_CTL Register

Address: 20E_0000h base + 1C4h offset = 20E_01C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA0 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: SD1_DATA0. 0000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA0 of instance: usdhc1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE3 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_SYNC of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN1_TX of instance: flexcan1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR21 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO18 of instance: gpio2 1000 **ALT8** — Select mux mode: ALT8 mux port: ANATOP_OTG1_ID of instance: anatop

30.5.110 SW_MUX_CTL_PAD_SD1_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1)

SW_MUX_CTL Register

Address: 20E_0000h base + 1C8h offset = 20E_01C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA1 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: SD1_DATA1. 0000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA1 of instance: usdhc1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CLK of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_BCLK of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN1_RX of instance: flexcan1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR22 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO19 of instance: gpio2 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG2_PWR of instance: usb

30.5.111 SW_MUX_CTL_PAD_SD1_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2)

SW_MUX_CTL Register

Address: 20E_0000h base + 1CCh offset = 20E_01CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA2 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: SD1_DATA2. 0000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA2 of instance: usdhc1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CAPTURE1 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_RX_DATA of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN2_TX of instance: flexcan2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR23 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO20 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: CCM_CLKO1 of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: USB_OTG2_OC of instance: usb

30.5.112 SW_MUX_CTL_PAD_SD1_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3)

SW_MUX_CTL Register

Address: 20E_0000h base + 1D0h offset = 20E_01D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA3 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: SD1_DATA3. 0000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA3 of instance: usdhc1 0001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CAPTURE2 of instance: gpt2 0010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_DATA of instance: sai2 0011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN2_RX of instance: flexcan2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_ADDR24 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO21 of instance: gpio2 0110 **ALT6** — Select mux mode: ALT6 mux port: CCM_CLKO2 of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: ANATOP_OTG2_ID of instance: anatop

30.5.113 SW_MUX_CTL_PAD_CSI_MCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK)

SW_MUX_CTL Register

Address: 20E_0000h base + 1D4h offset = 20E_01D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_MCLK 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_MCLK. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_MCLK of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_CD_B of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: RAWNAND_CE2_B of instance: rawnand 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C1_SDA of instance: i2c1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_CS0_B of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO17 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SNVS_HP_VIO_5_CTL of instance: snvs_hp 1000 **ALT8** — Select mux mode: ALT8 mux port: UART6_TX of instance: uart6

30.5.114 SW_MUX_CTL_PAD_CSI_PIXCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK)

SW_MUX_CTL Register

Address: 20E_0000h base + 1D8h offset = 20E_01D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_PIXCLK 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_PIXCLK. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_PIXCLK of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_WP of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: RAWNAND_CE3_B of instance: rawnand 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C1_SCL of instance: i2c1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_OE of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO18 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SNVS_HP_VIO_5 of instance: snvs_hp 1000 **ALT8** — Select mux mode: ALT8 mux port: UART6_RX of instance: uart6

30.5.115 SW_MUX_CTL_PAD_CSI_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC)

SW_MUX_CTL Register

Address: 20E_0000h base + 1DCh offset = 20E_01DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_VSYNC 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_VSYNC. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_VSYNC of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_CLK of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT1_CLK of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C2_SDA of instance: i2c2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_RW of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO19 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: PWM7_OUT of instance: pwm7 1000 **ALT8** — Select mux mode: ALT8 mux port: UART6_RTS_B of instance: uart6

30.5.116 SW_MUX_CTL_PAD_CSI_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC)

SW_MUX_CTL Register

Address: 20E_0000h base + 1E0h offset = 20E_01E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_HSYNC 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_HSYNC. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_HSYNC of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_CMD of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT1_PD of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: I2C2_SCL of instance: i2c2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_LBA_B of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO20 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: PWM8_OUT of instance: pwm8 1000 **ALT8** — Select mux mode: ALT8 mux port: UART6_CTS_B of instance: uart6

30.5.117 SW_MUX_CTL_PAD_CSI_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA00)

SW_MUX_CTL Register

Address: 20E_0000h base + 1E4h offset = 20E_01E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_DATA00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA00. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA02 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA0 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT1_RST_B of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI2_SCLK of instance: ecspi2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD00 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO21 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SRC_INT_BOOT of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_TX of instance: uart5

30.5.118 SW_MUX_CTL_PAD_CSI_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA01)

SW_MUX_CTL Register

Address: 20E_0000h base + 1E8h offset = 20E_01E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_DATA01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA01. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA03 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA1 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT1_SVEN of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI2_SS0 of instance: ecspi2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD01 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO22 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SAI1_MCLK of instance: sai1 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_RX of instance: uart5

30.5.119 SW_MUX_CTL_PAD_CSI_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA02)

SW_MUX_CTL Register

Address: 20E_0000h base + 1ECh offset = 20E_01ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_DATA02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA02. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA04 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA2 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM1_PORT1_TRXD of instance: sim1 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI2_MOSI of instance: ecspi2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD02 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO23 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SAI1_RX_SYNC of instance: sai1 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_RTS_B of instance: uart5

30.5.120 SW_MUX_CTL_PAD_CSI_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA03)

SW_MUX_CTL Register

Address: 20E_0000h base + 1F0h offset = 20E_01F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_DATA03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA03. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA05 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA3 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT1_PD of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI2_MISO of instance: ecspi2 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD03 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO24 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SAI1_RX_BCLK of instance: sai1 1000 **ALT8** — Select mux mode: ALT8 mux port: UART5_CTS_B of instance: uart5

30.5.121 SW_MUX_CTL_PAD_CSI_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA04)

SW_MUX_CTL Register

Address: 20E_0000h base + 1F4h offset = 20E_01F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_DATA04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA04. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA06 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA4 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT1_CLK of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI1_SCLK of instance: ecspi1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD04 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO25 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SAI1_TX_SYNC of instance: sai1 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC1_WP of instance: usdhc1

30.5.122 SW_MUX_CTL_PAD_CSI_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA05)

SW_MUX_CTL Register

Address: 20E_0000h base + 1F8h offset = 20E_01F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_DATA05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA05. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA07 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA5 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT1_RST_B of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI1_SS0 of instance: ecspi1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD05 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO26 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SAI1_TX_BCLK of instance: sai1 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC1_CD_B of instance: usdhc1

30.5.123 SW_MUX_CTL_PAD_CSI_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA06)

SW_MUX_CTL Register

Address: 20E_0000h base + 1FCh offset = 20E_01FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_MUX_CTL_PAD_CSI_DATA06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA06. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA08 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA6 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT1_SVEN of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI1_MOSI of instance: ecspi1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD06 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO27 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SAI1_RX_DATA of instance: sai1 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC1_RESET_B of instance: usdhc1

30.5.124 SW_MUX_CTL_PAD_CSI_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_CSI_DATA07)

SW_MUX_CTL Register

Address: 20E_0000h base + 200h offset = 20E_0200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

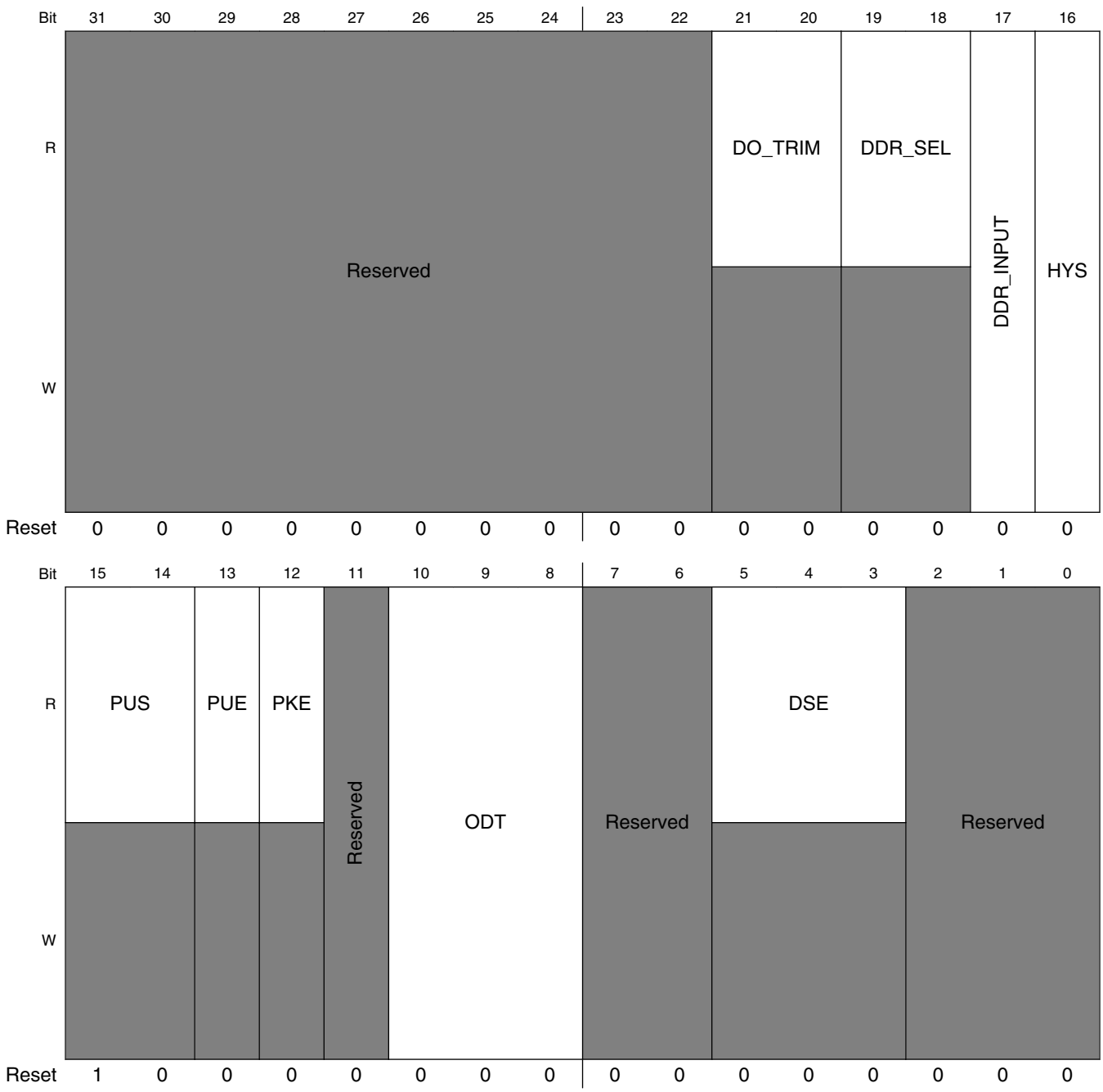
IOMUXC_SW_MUX_CTL_PAD_CSI_DATA07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad CSI_DATA07 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: CSI_DATA07. 0000 **ALT0** — Select mux mode: ALT0 mux port: CSI_DATA09 of instance: csi 0001 **ALT1** — Select mux mode: ALT1 mux port: USDHC2_DATA7 of instance: usdhc2 0010 **ALT2** — Select mux mode: ALT2 mux port: SIM2_PORT1_TRXD of instance: sim2 0011 **ALT3** — Select mux mode: ALT3 mux port: ECSPI1_MISO of instance: ecspi1 0100 **ALT4** — Select mux mode: ALT4 mux port: EIM_AD07 of instance: eim 0101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO28 of instance: gpio4 0110 **ALT6** — Select mux mode: ALT6 mux port: SAI1_TX_DATA of instance: sai1 1000 **ALT8** — Select mux mode: ALT8 mux port: USDHC1_VSELECT of instance: usdhc1

30.5.125 SW_PAD_CTL_PAD_DRAM_ADDR00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR00)

SW_PAD_CTL Register

Address: 20E_0000h base + 204h offset = 20E_0204h



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR00 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR00 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR00 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR00 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.126 SW_PAD_CTL_PAD_DRAM_ADDR01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR01)

SW_PAD_CTL Register

Address: 20E_0000h base + 208h offset = 20E_0208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR01 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR01 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR01 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

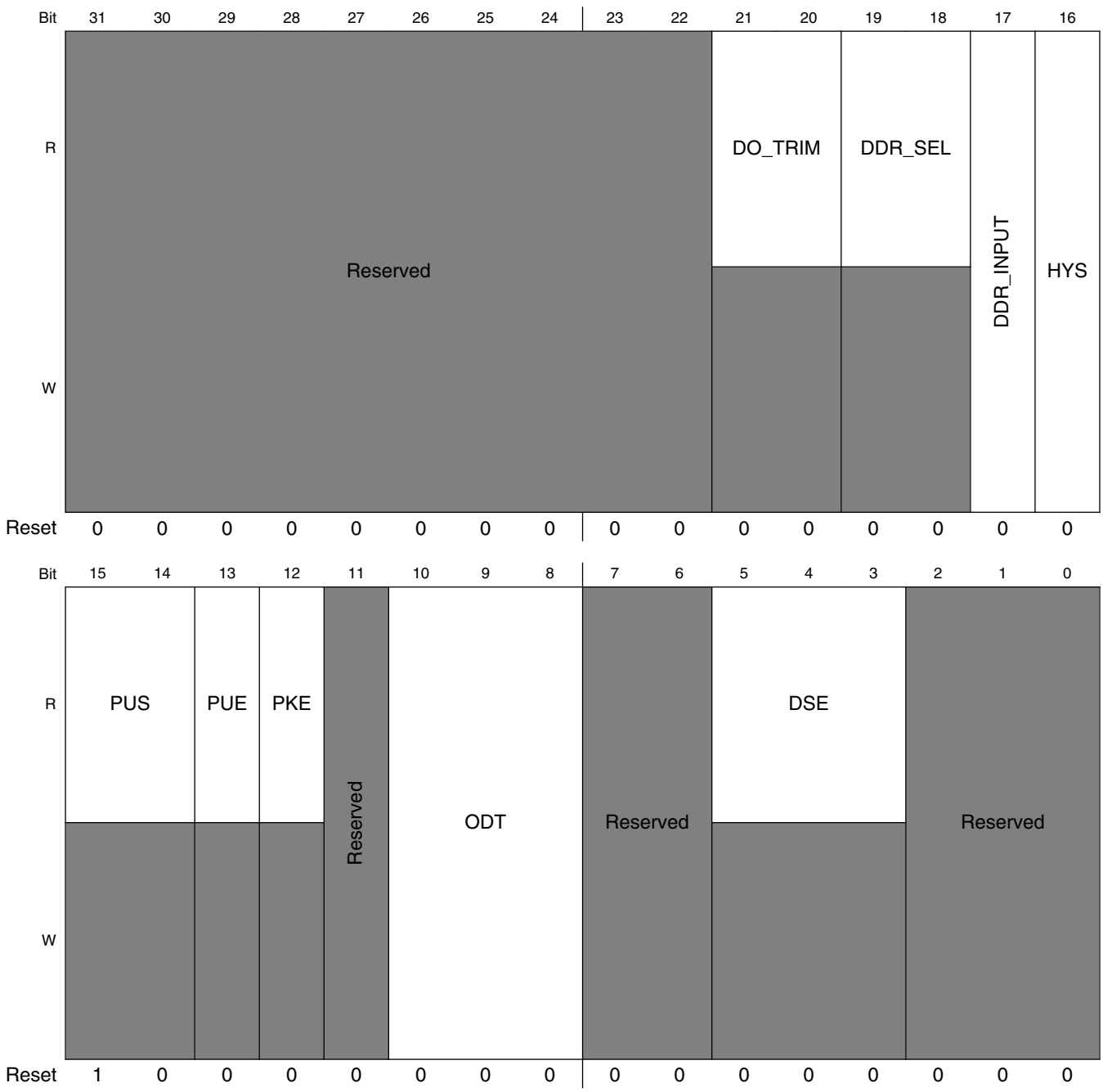
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR01 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.127 SW_PAD_CTL_PAD_DRAM_ADDR02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR02)

SW_PAD_CTL Register

Address: 20E_0000h base + 20Ch offset = 20E_020Ch



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR02 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR02 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR02 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

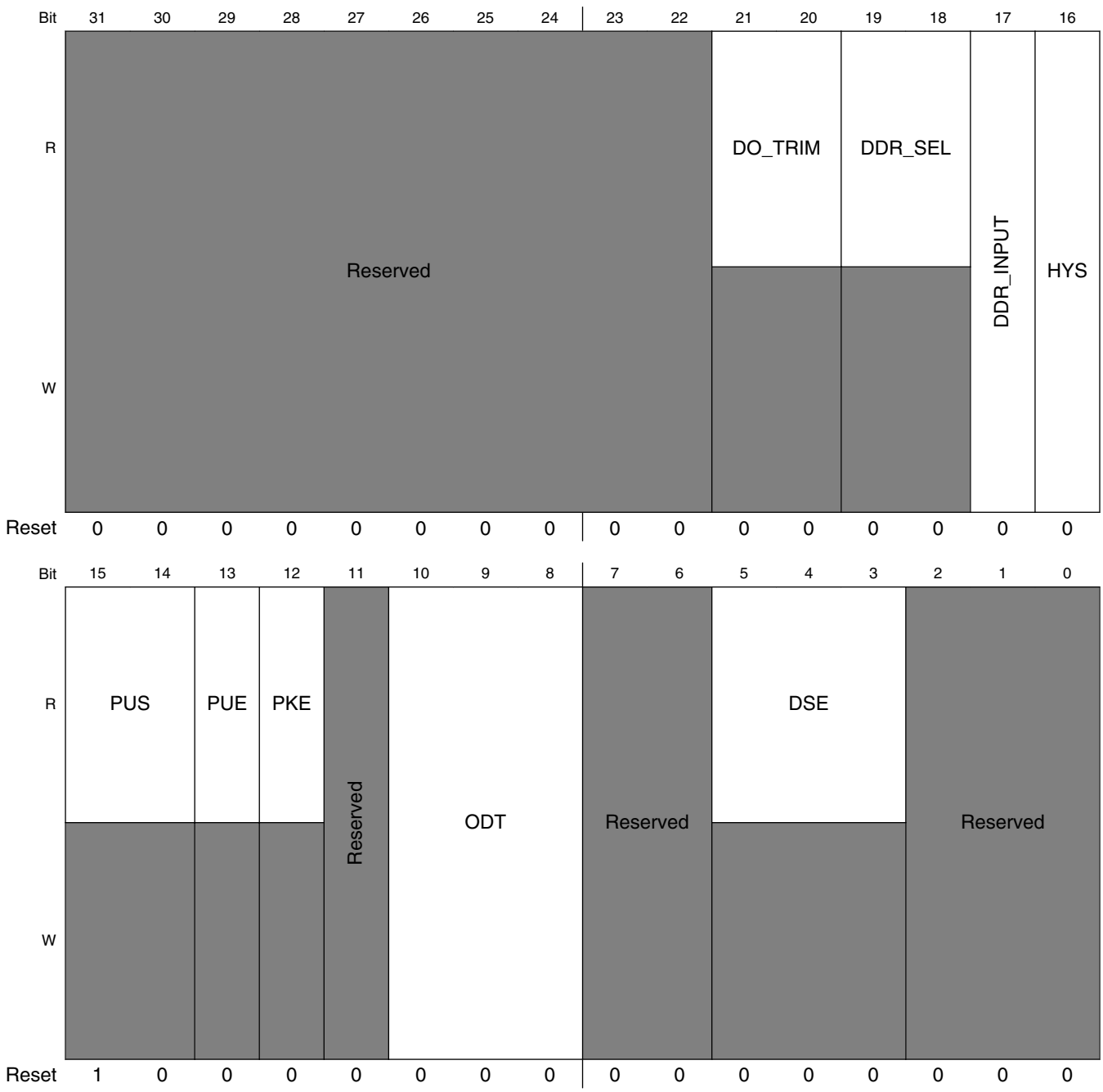
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR02 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.128 SW_PAD_CTL_PAD_DRAM_ADDR03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR03)

SW_PAD_CTL Register

Address: 20E_0000h base + 210h offset = 20E_0210h



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR03 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR03 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR03 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR03 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.129 SW_PAD_CTL_PAD_DRAM_ADDR04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR04)

SW_PAD_CTL Register

Address: 20E_0000h base + 214h offset = 20E_0214h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR04 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR04 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR04 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR04 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.130 SW_PAD_CTL_PAD_DRAM_ADDR05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR05)

SW_PAD_CTL Register

Address: 20E_0000h base + 218h offset = 20E_0218h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR05 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR05 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR05 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

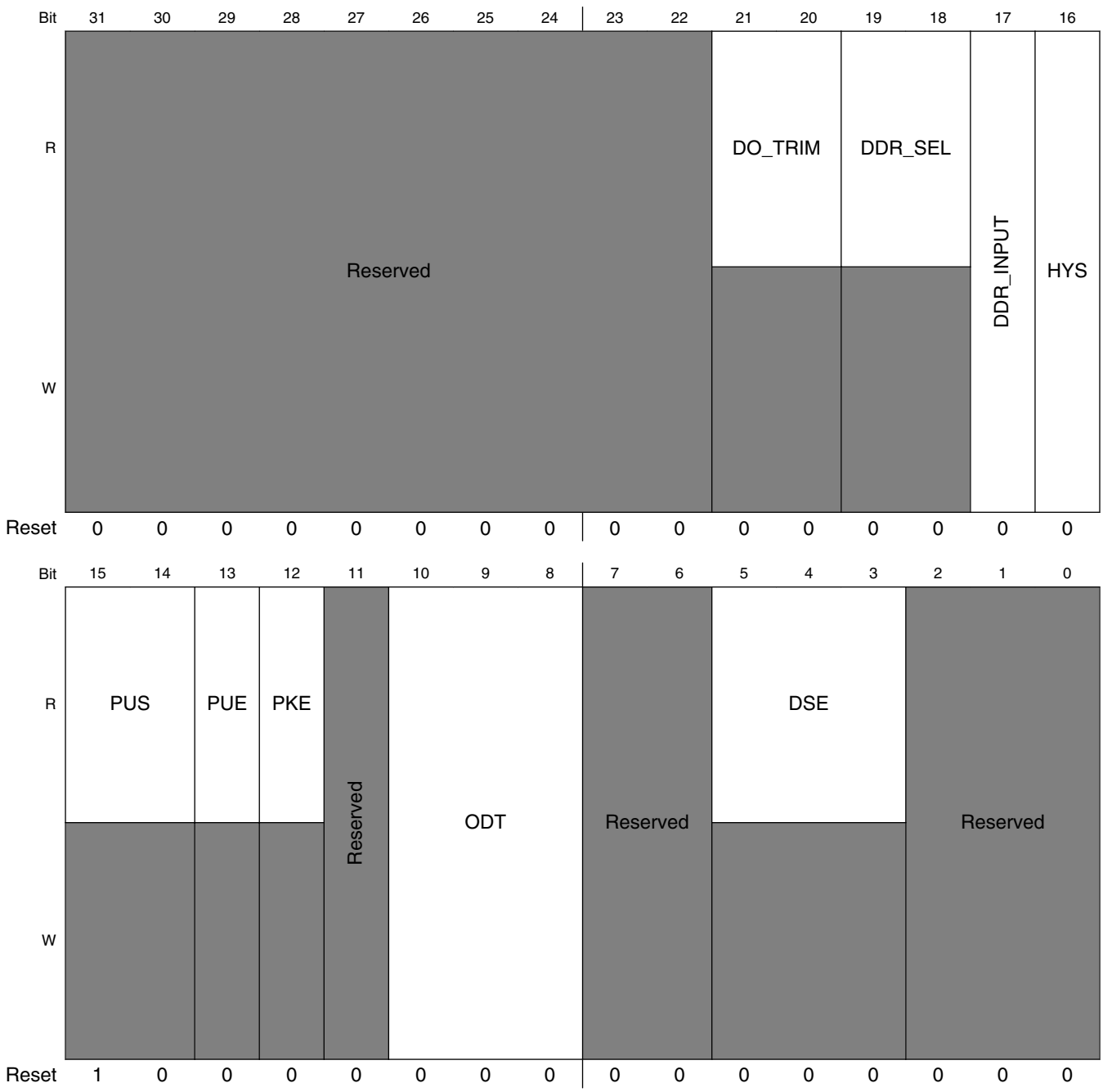
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR05 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.131 SW_PAD_CTL_PAD_DRAM_ADDR06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR06)

SW_PAD_CTL Register

Address: 20E_0000h base + 21Ch offset = 20E_021Ch



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR06 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR06 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR06 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

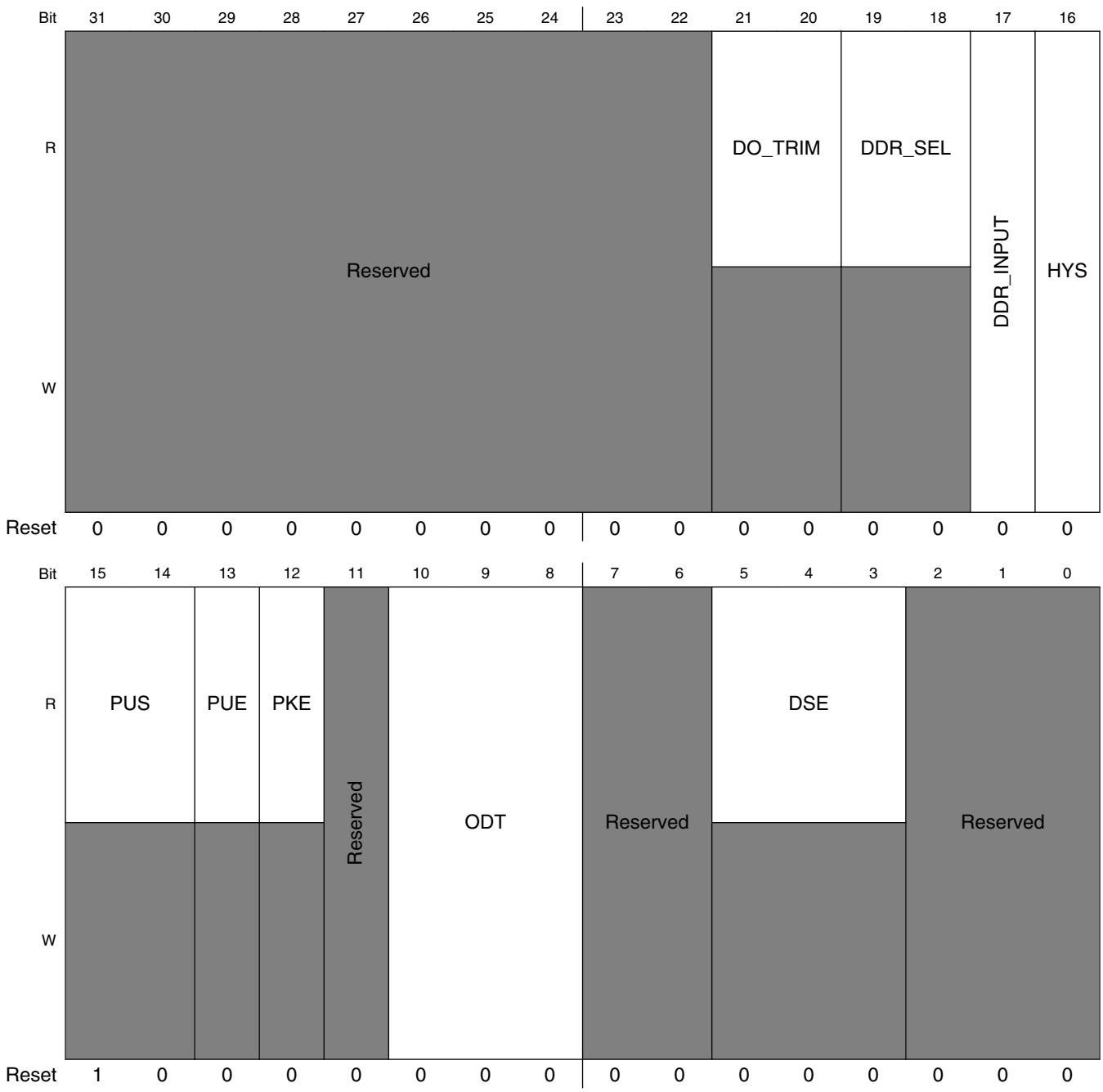
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR06 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.132 SW_PAD_CTL_PAD_DRAM_ADDR07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR07)

SW_PAD_CTL Register

Address: 20E_0000h base + 220h offset = 20E_0220h



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR07 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR07 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR07 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

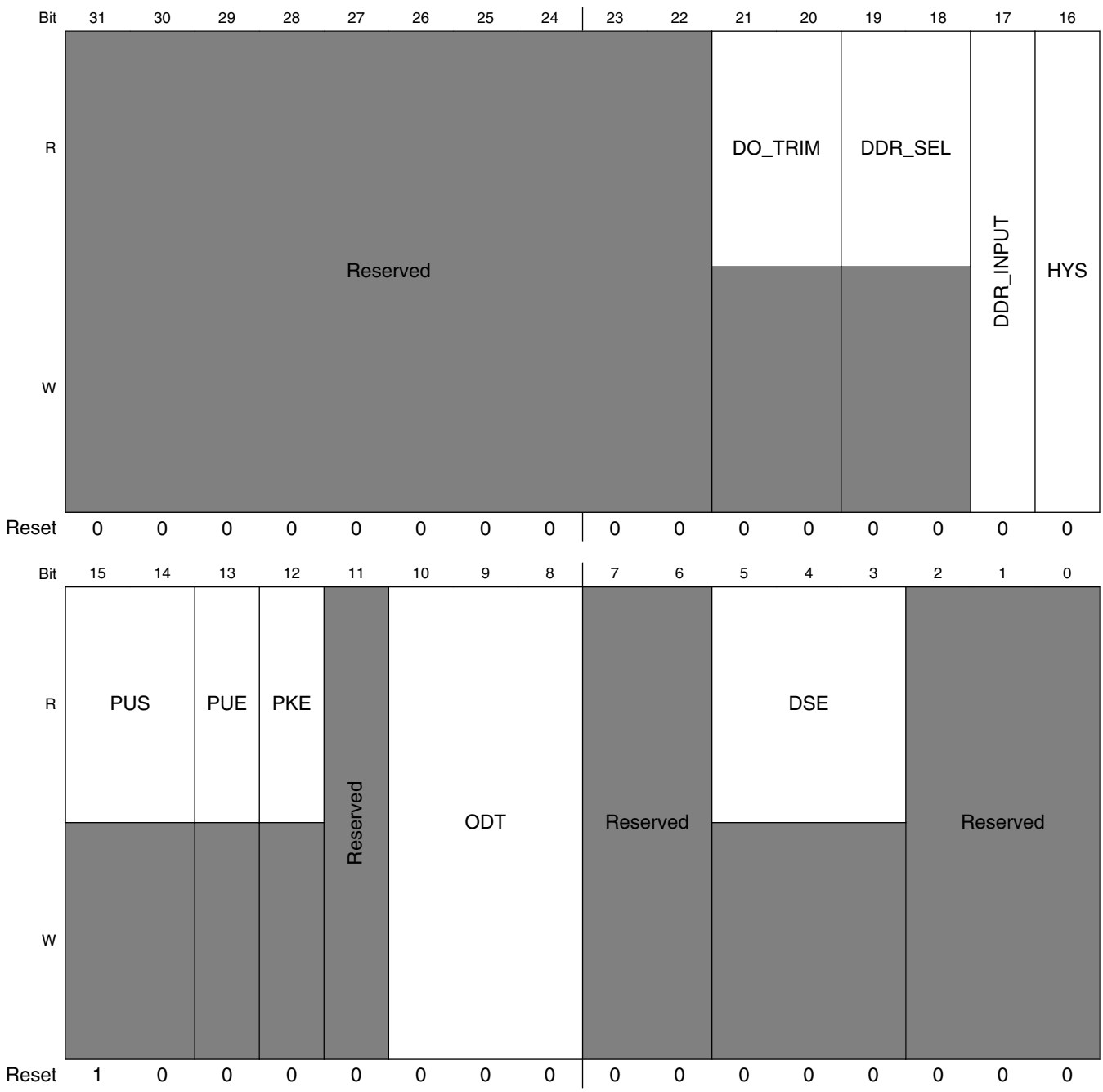
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR07 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.133 SW_PAD_CTL_PAD_DRAM_ADDR08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR08)

SW_PAD_CTL Register

Address: 20E_0000h base + 224h offset = 20E_0224h



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR08 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR08 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR08 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR08 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.134 SW_PAD_CTL_PAD_DRAM_ADDR09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR09)

SW_PAD_CTL Register

Address: 20E_0000h base + 228h offset = 20E_0228h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR09 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR09 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR09 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR09 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.135 SW_PAD_CTL_PAD_DRAM_ADDR10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR10)

SW_PAD_CTL Register

Address: 20E_0000h base + 22Ch offset = 20E_022Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR10 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ADDR10 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR10 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR10 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR10 field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.136 SW_PAD_CTL_PAD_DRAM_ADDR11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR11)

SW_PAD_CTL Register

Address: 20E_0000h base + 230h offset = 20E_0230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR11 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ADDR11 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR11 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR11 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

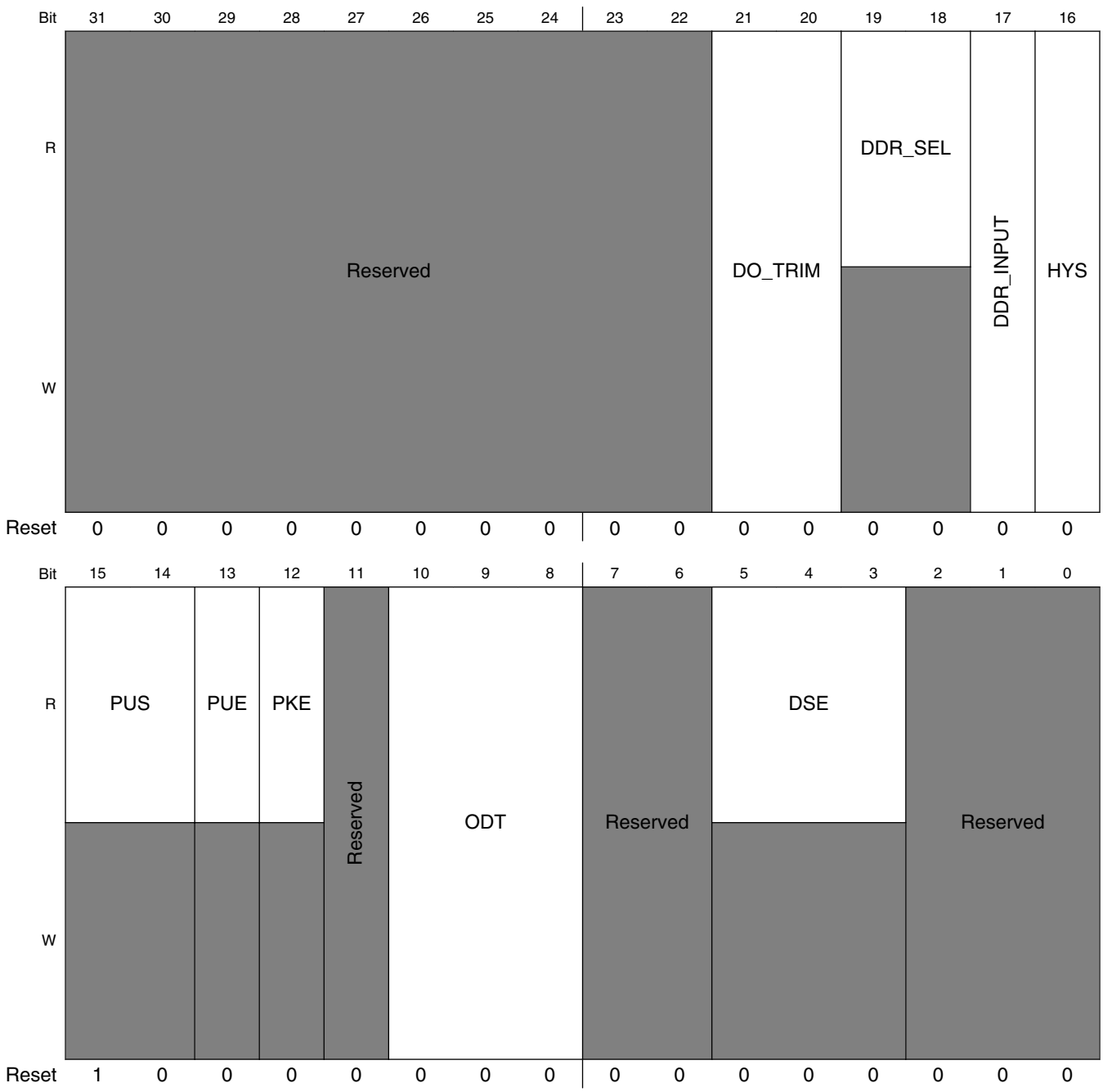
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR11 field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.137 SW_PAD_CTL_PAD_DRAM_ADDR12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR12)

SW_PAD_CTL Register

Address: 20E_0000h base + 234h offset = 20E_0234h



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR12 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ADDR12 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR12 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR12 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

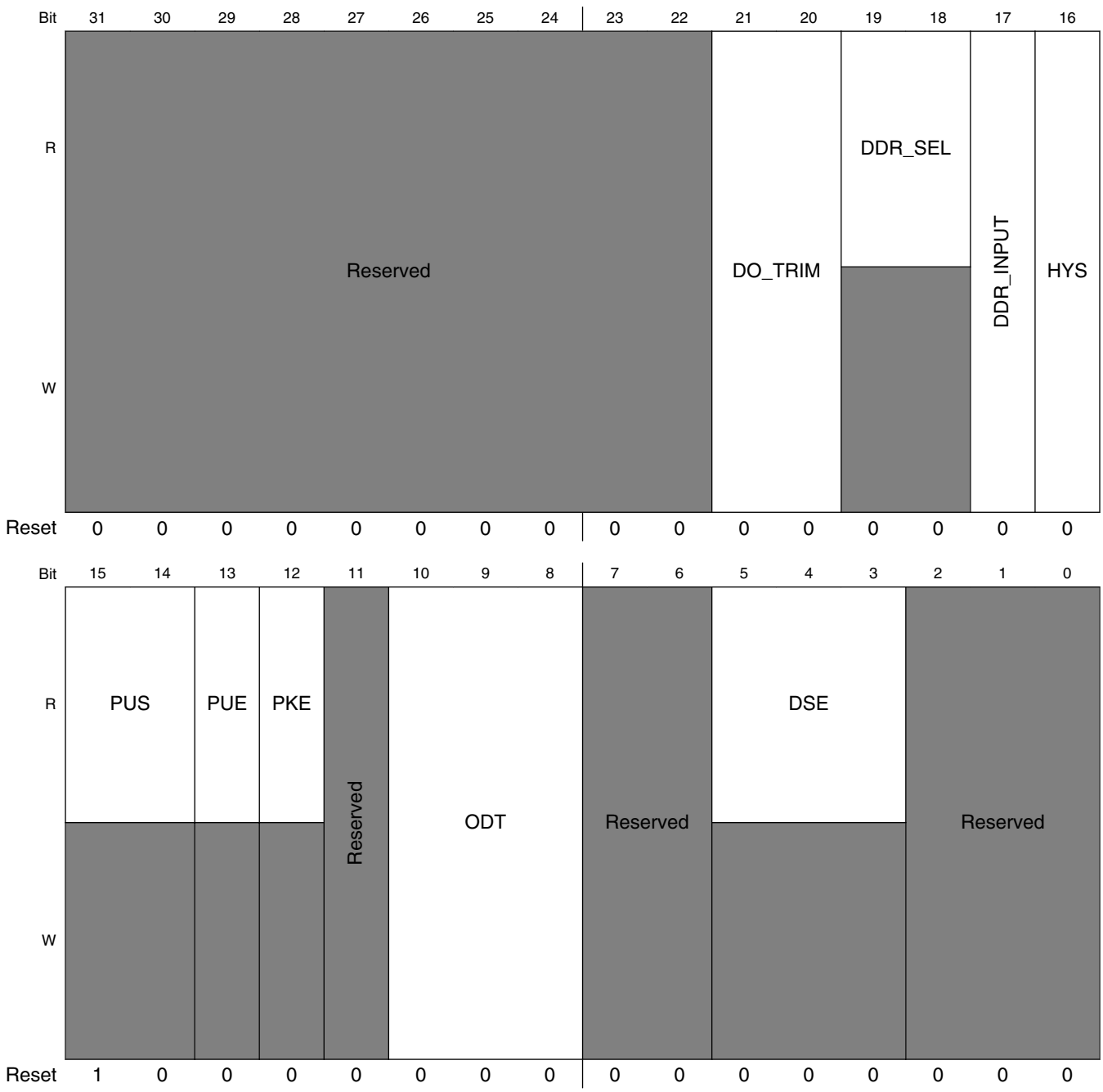
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR12 field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.138 SW_PAD_CTL_PAD_DRAM_ADDR13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR13)

SW_PAD_CTL Register

Address: 20E_0000h base + 238h offset = 20E_0238h



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR13 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ADDR13 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR13 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR13 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR13 field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.139 SW_PAD_CTL_PAD_DRAM_ADDR14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR14)

SW_PAD_CTL Register

Address: 20E_0000h base + 23Ch offset = 20E_023Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR14 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ADDR14 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR14 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR14 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

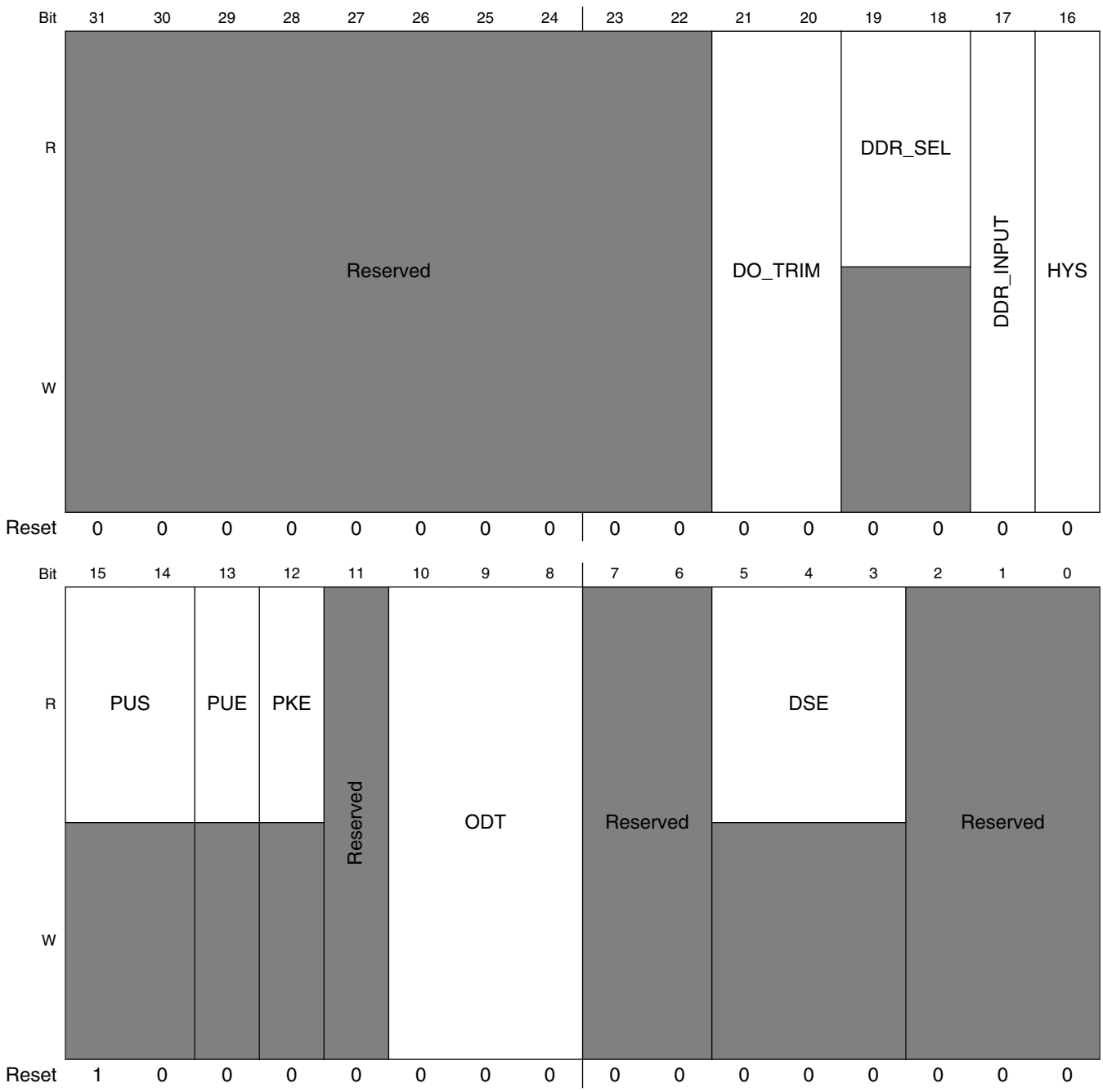
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR14 field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.140 SW_PAD_CTL_PAD_DRAM_ADDR15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR15)

SW_PAD_CTL Register

Address: 20E_0000h base + 240h offset = 20E_0240h



IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR15 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ADDR15 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ADDR15 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ADDR15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_ADDR15 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

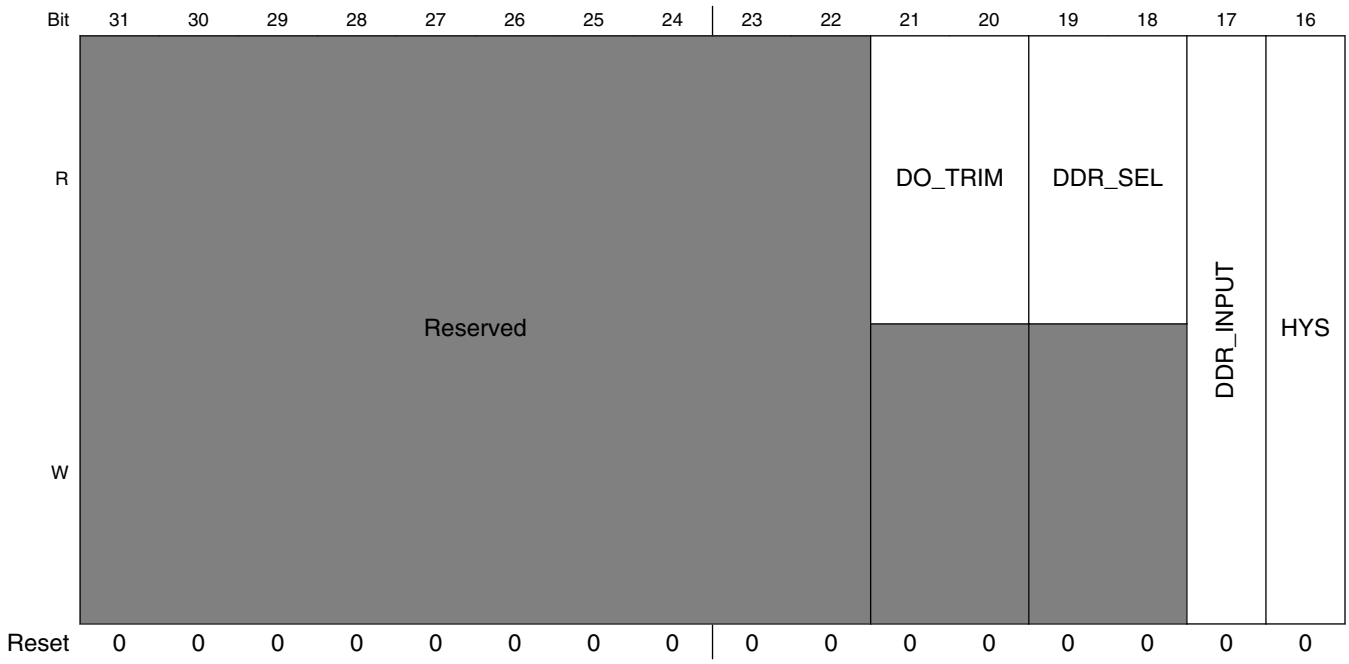
IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDR15 field descriptions (continued)

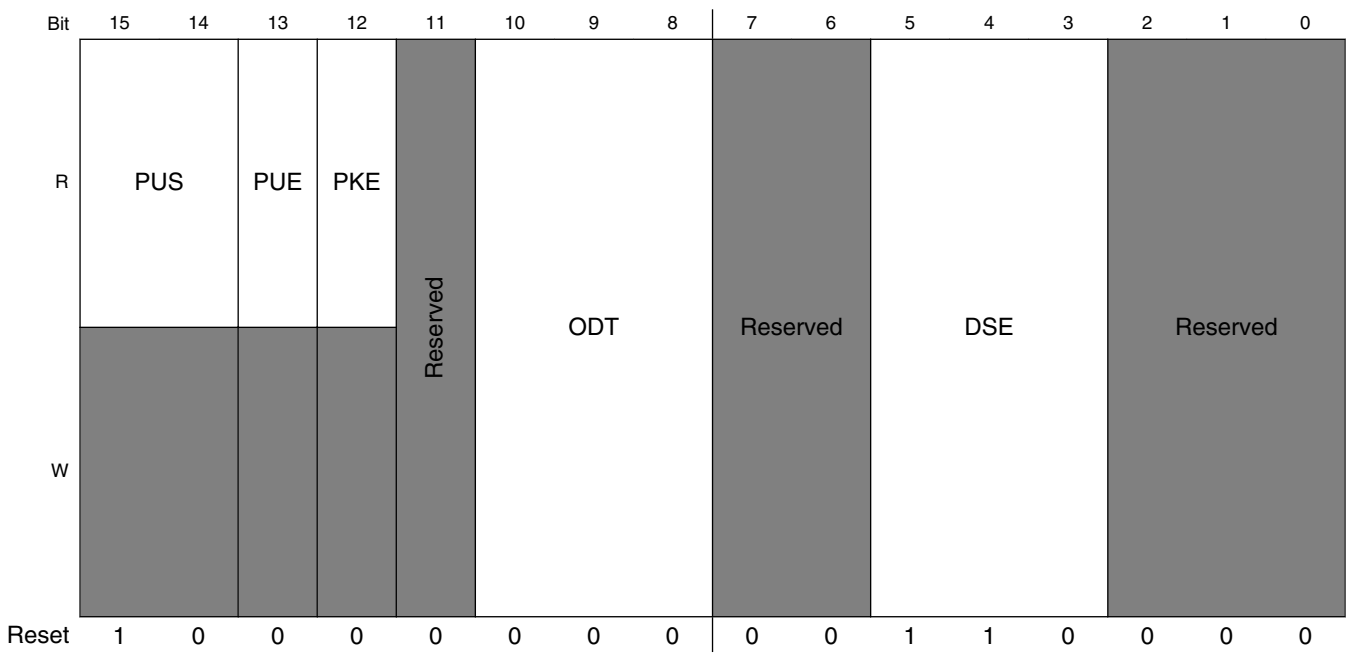
Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.141 SW_PAD_CTL_PAD_DRAM_DQM0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM0)

SW_PAD_CTL Register

Address: 20E_0000h base + 244h offset = 20E_0244h





IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM0 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_DQM0 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_DQM0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_DQM0 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM0 field descriptions (continued)

Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_DQM0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.142 SW_PAD_CTL_PAD_DRAM_DQM1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM1)

SW_PAD_CTL Register

Address: 20E_0000h base + 248h offset = 20E_0248h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM1 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Read Only Field 0 **DO_TRIM** — min delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_DQM1 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_DQM1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_DQM1 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT

Table continues on the next page...

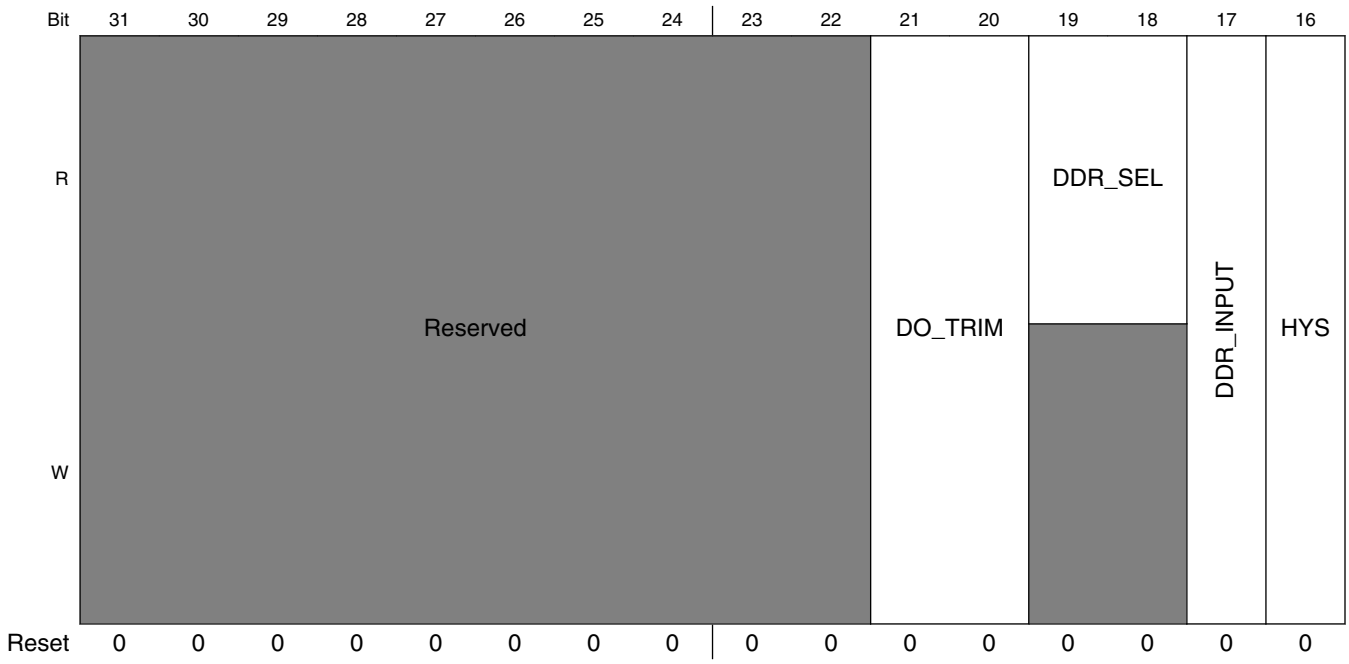
IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM1 field descriptions (continued)

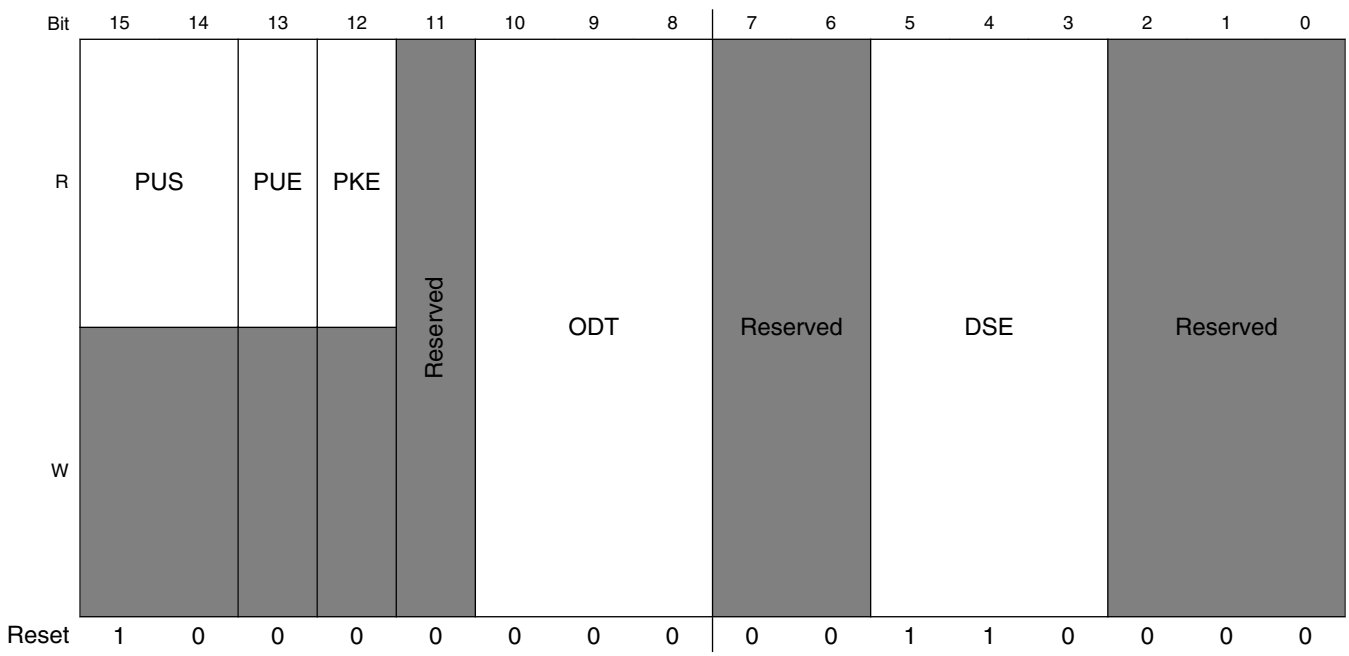
Field	Description
	110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_DQM1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.143 SW_PAD_CTL_PAD_DRAM_RAS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_RAS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 24Ch offset = 20E_024Ch





IOMUXC_SW_PAD_CTL_PAD_DRAM_RAS_B field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_RAS_B 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_RAS_B 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_RAS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_RAS_B 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

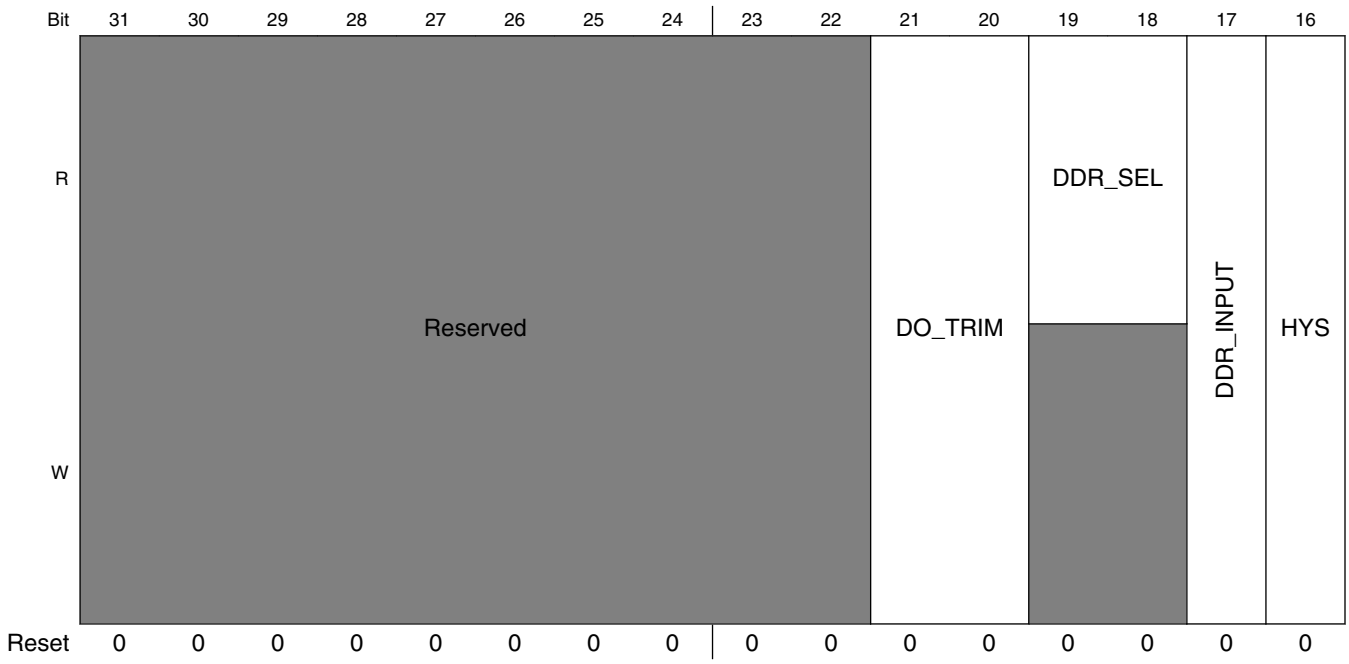
IOMUXC_SW_PAD_CTL_PAD_DRAM_RAS_B field descriptions (continued)

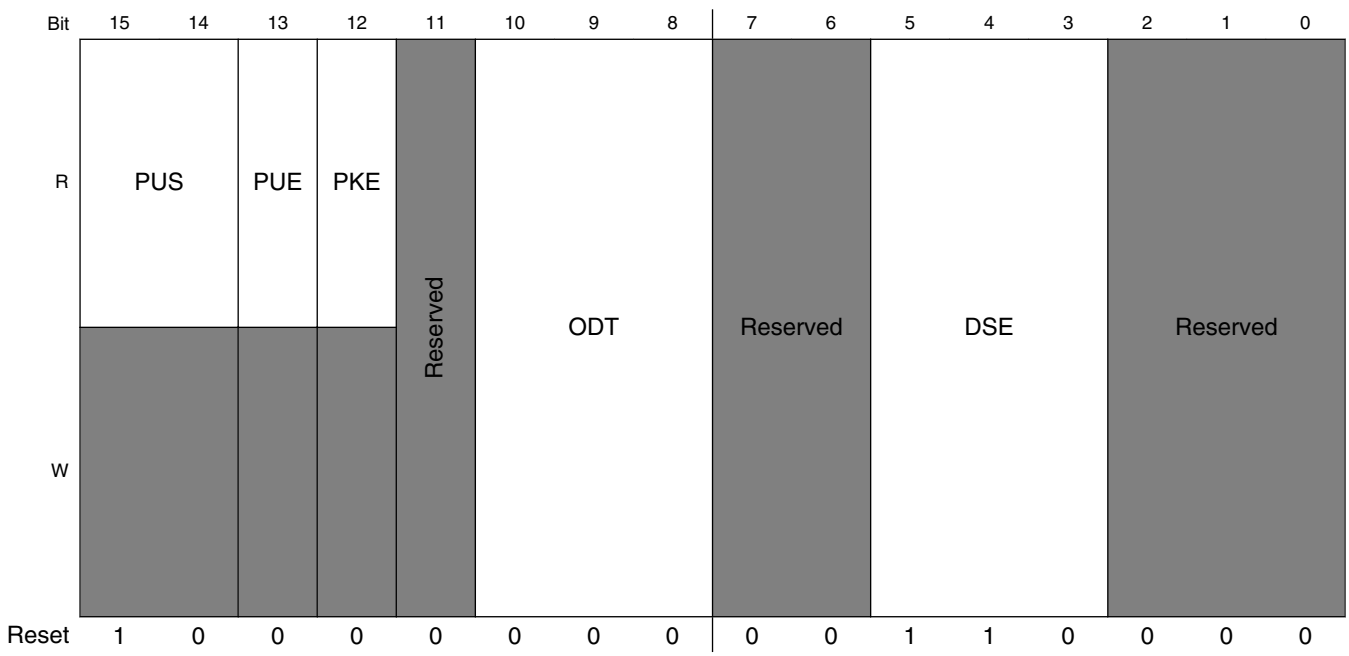
Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_RAS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.144 SW_PAD_CTL_PAD_DRAM_CAS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CAS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 250h offset = 20E_0250h





IOMUXC_SW_PAD_CTL_PAD_DRAM_CAS_B field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_CAS_B 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_CAS_B 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_CAS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_CAS_B 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

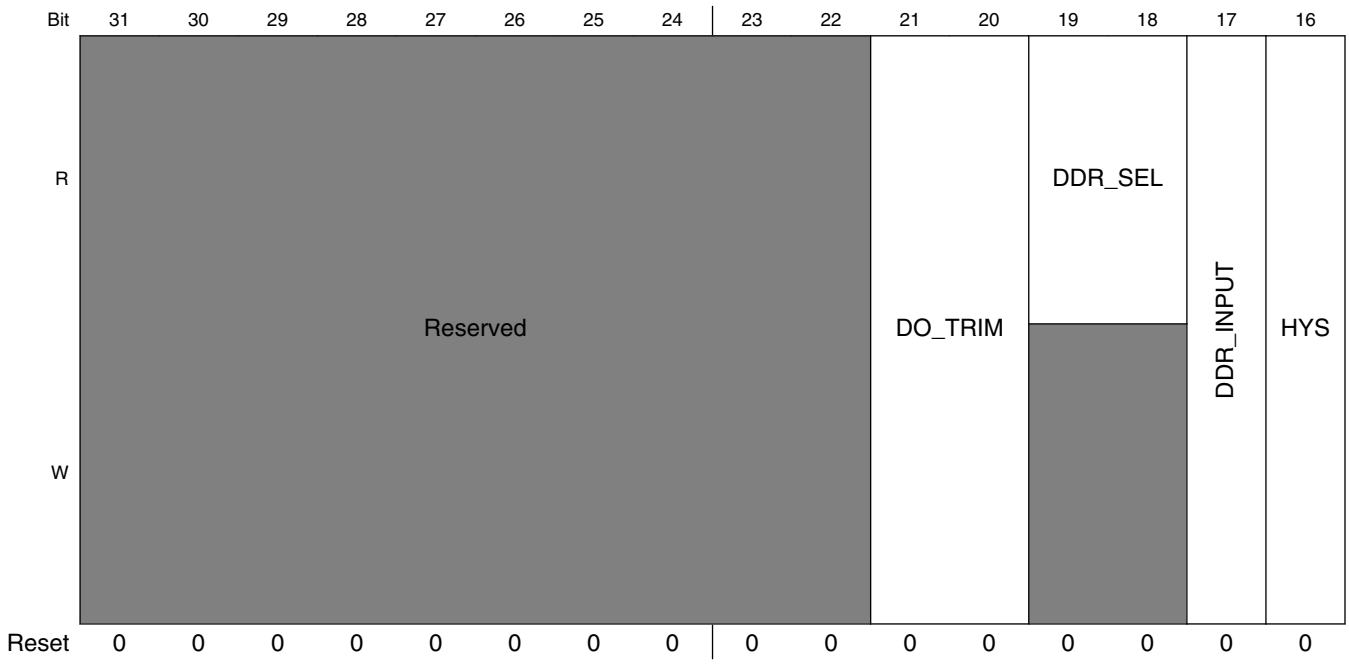
IOMUXC_SW_PAD_CTL_PAD_DRAM_CAS_B field descriptions (continued)

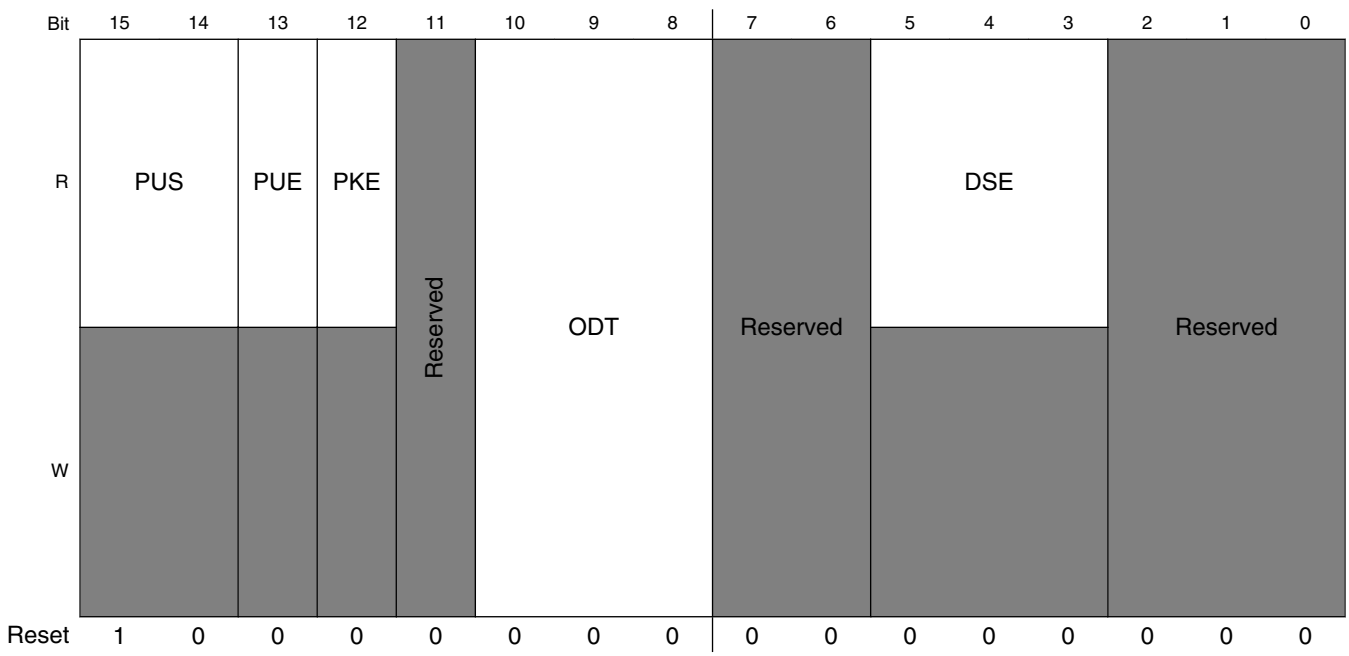
Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_CAS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.145 SW_PAD_CTL_PAD_DRAM_CS0_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CS0_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 254h offset = 20E_0254h





IOMUXC_SW_PAD_CTL_PAD_DRAM_CS0_B field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_CS0_B 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_CS0_B 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_CS0_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_CS0_B 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_CS0_B field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.146 SW_PAD_CTL_PAD_DRAM_CS1_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_CS1_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 258h offset = 20E_0258h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_CS1_B field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_CS1_B 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_CS1_B 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_CS1_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_CS1_B 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

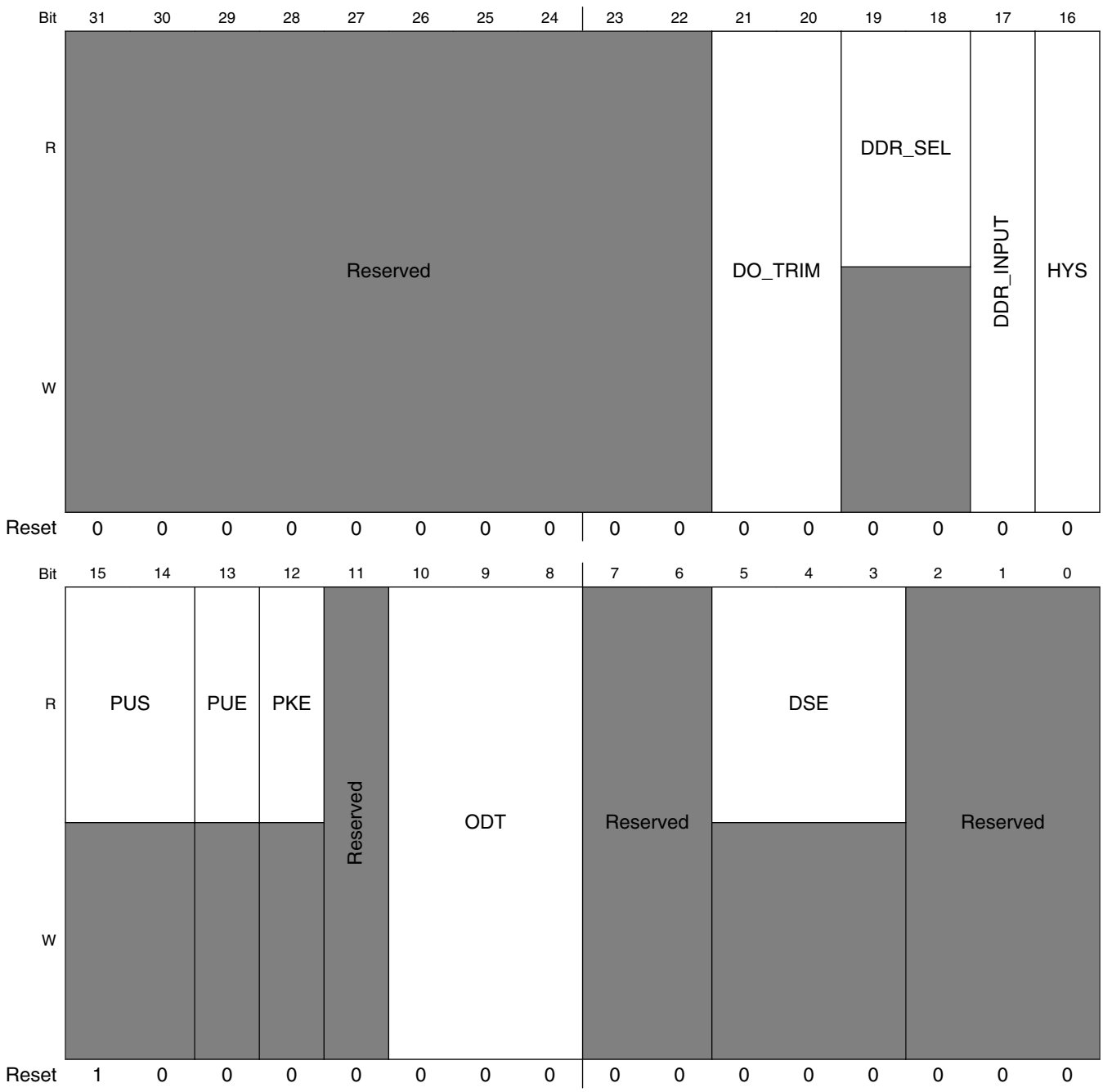
IOMUXC_SW_PAD_CTL_PAD_DRAM_CS1_B field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.147 SW_PAD_CTL_PAD_DRAM_SDWE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDWE_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 25Ch offset = 20E_025Ch



IOMUXC_SW_PAD_CTL_PAD_DRAM_SDWE_B field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDWE_B 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_SDWE_B 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_SDWE_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_SDWE_B 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_SDWE_B field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.148 SW_PAD_CTL_PAD_DRAM_ODT0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT0)

SW_PAD_CTL Register

Address: 20E_0000h base + 260h offset = 20E_0260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT0 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ODT0 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ODT0 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ODT0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DRAM_ODT0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_ODT0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_ODT0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field

Table continues on the next page...

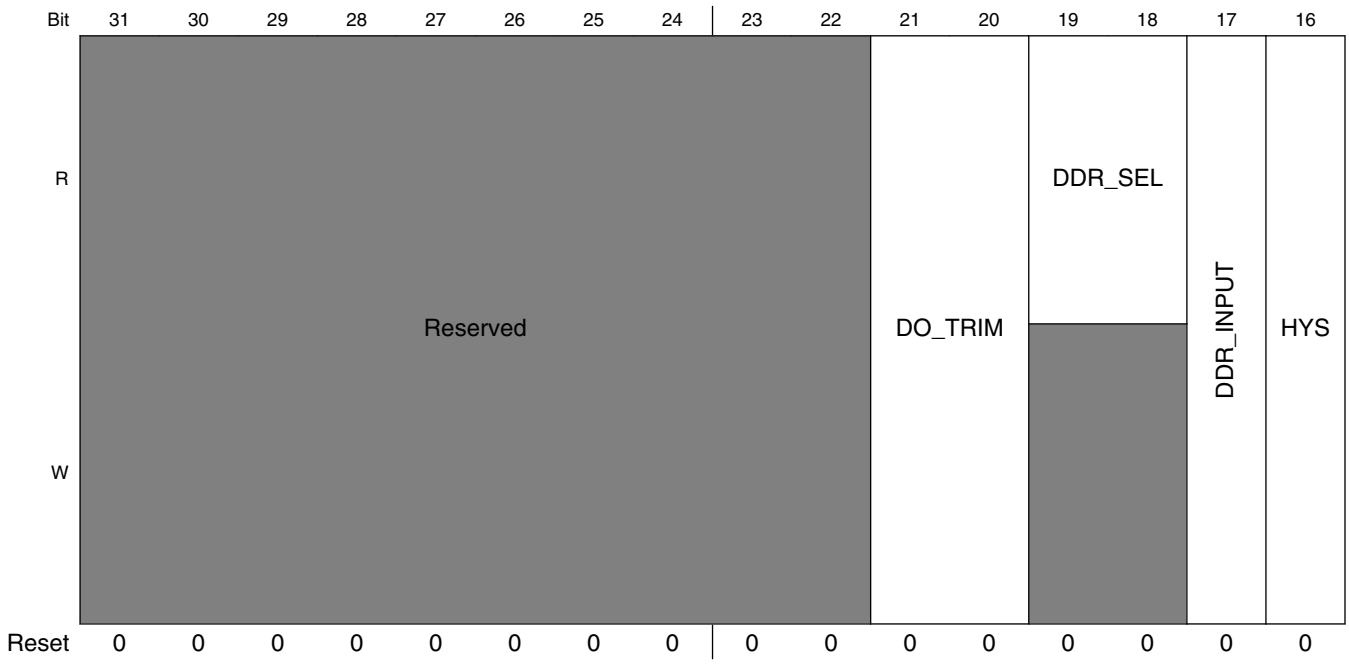
IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT0 field descriptions (continued)

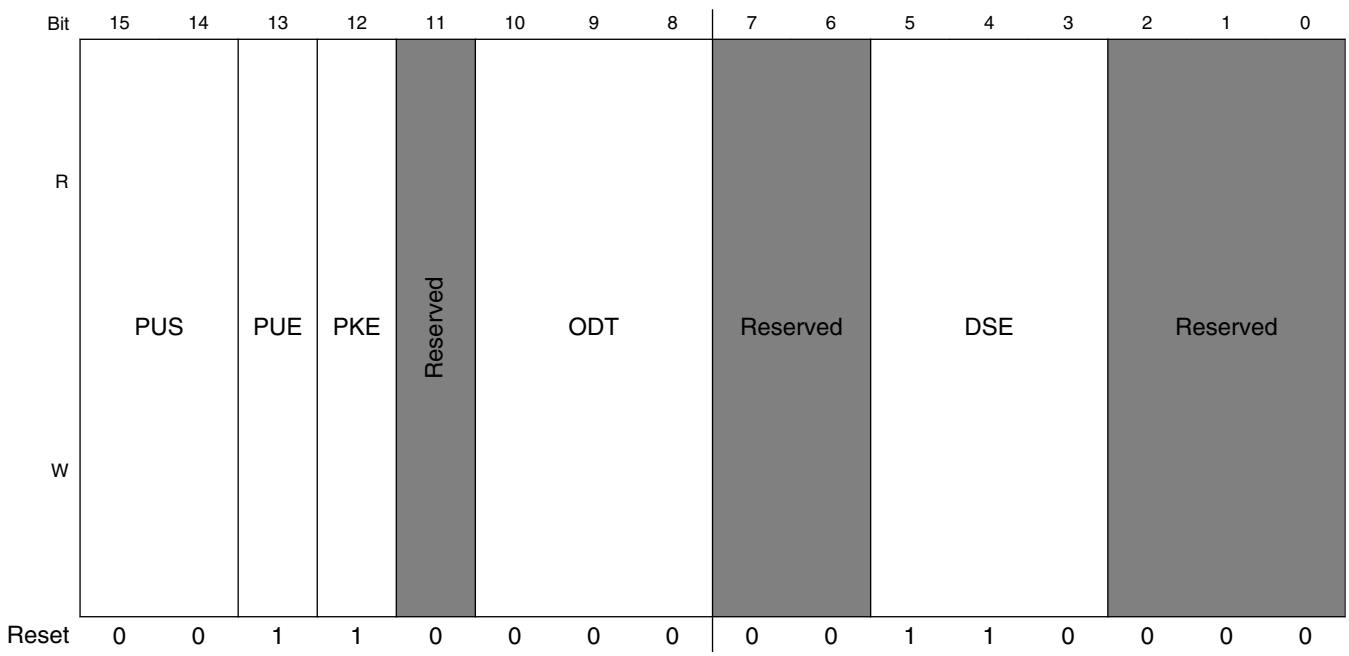
Field	Description
	Select one out of next values for pad: DRAM_ODT0 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_ODT0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.149 SW_PAD_CTL_PAD_DRAM_ODT1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT1)

SW_PAD_CTL Register

Address: 20E_0000h base + 264h offset = 20E_0264h





IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT1 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_ODT1 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_ODT1 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_ODT1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DRAM_ODT1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_ODT1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_ODT1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_ODT1 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DRAM_ODT1 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_ODT1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.150 SW_PAD_CTL_PAD_DRAM_SDBA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA0)

SW_PAD_CTL Register

Address: 20E_0000h base + 268h offset = 20E_0268h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA0 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDBA0 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_SDBA0 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_SDBA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_SDBA0 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

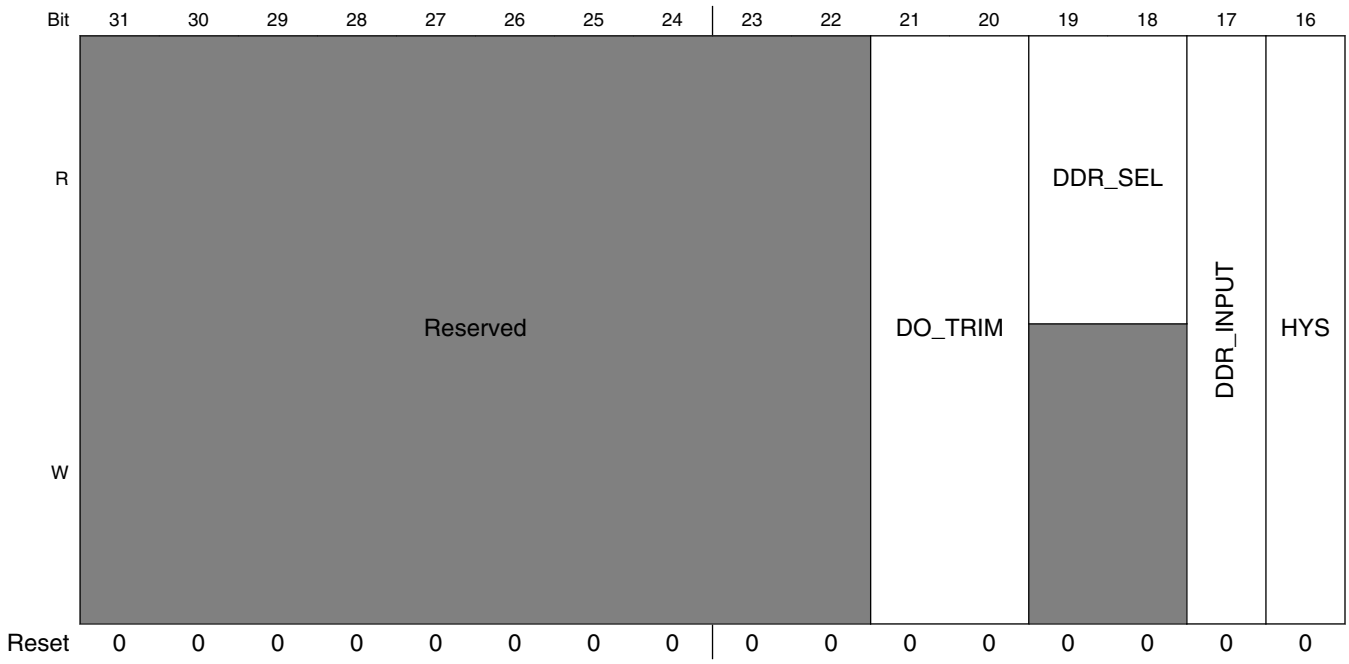
IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA0 field descriptions (continued)

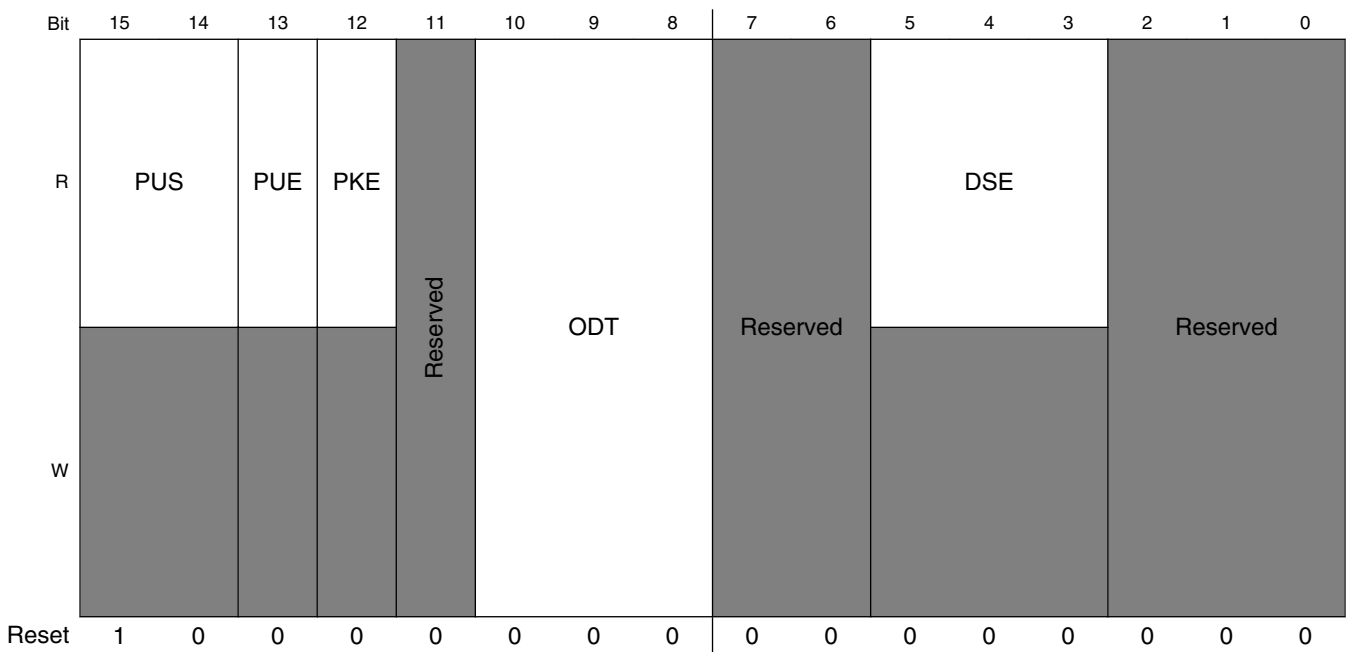
Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.151 SW_PAD_CTL_PAD_DRAM_SDBA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA1)

SW_PAD_CTL Register

Address: 20E_0000h base + 26Ch offset = 20E_026Ch





IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA1 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDBA1 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_SDBA1 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_SDBA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field 0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_SDBA1 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA1 field descriptions (continued)

Field	Description
	011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.152 SW_PAD_CTL_PAD_DRAM_SDBA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA2)

SW_PAD_CTL Register

Address: 20E_0000h base + 270h offset = 20E_0270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA2 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDBA2 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_SDBA2 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_SDBA2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDBA2 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDBA2 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_SDBA2 000 **ODT_0_off** — off

Table continues on the next page...

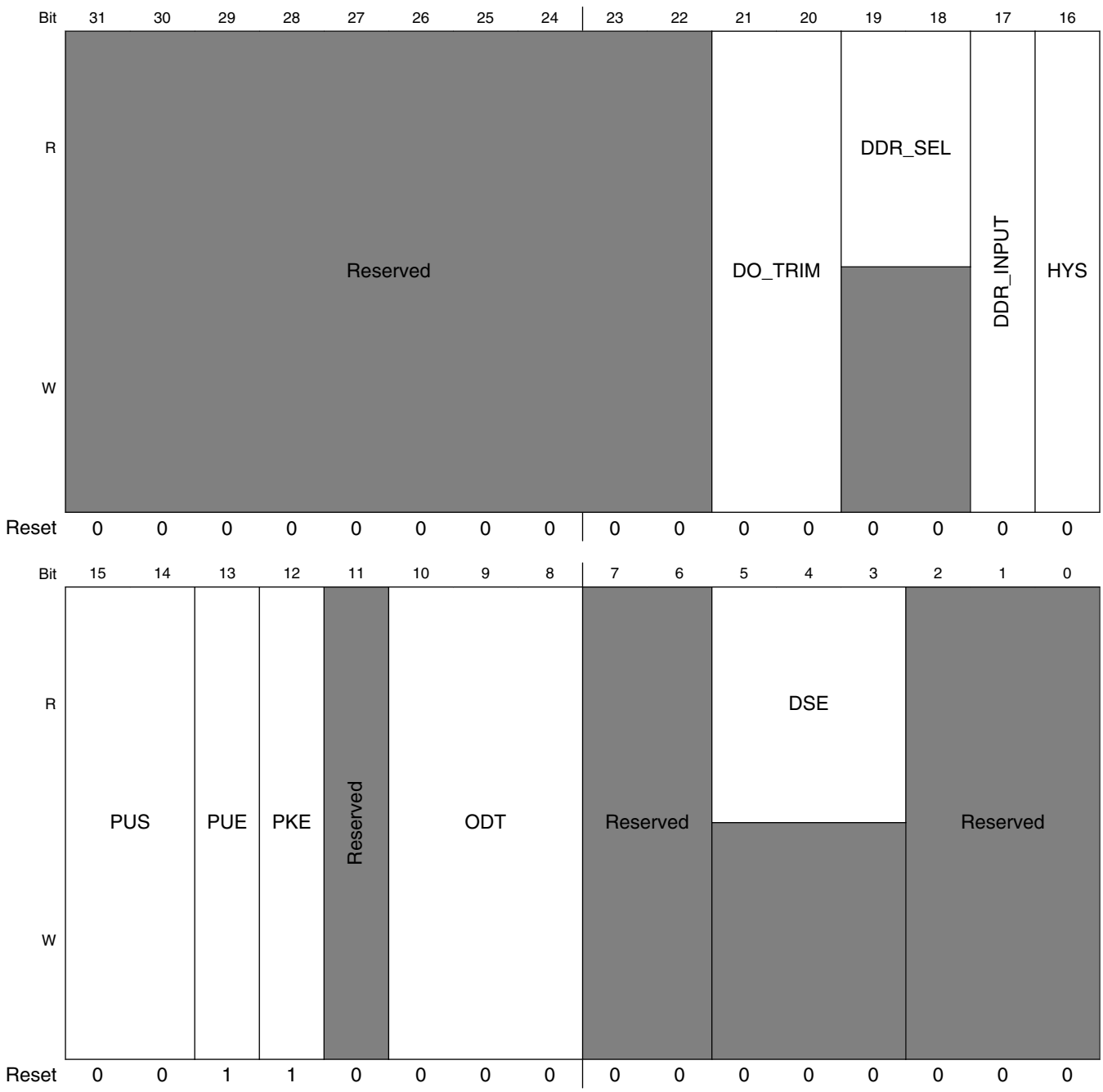
IOMUXC_SW_PAD_CTL_PAD_DRAM_SDBA2 field descriptions (continued)

Field	Description
	001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.153 SW_PAD_CTL_PAD_DRAM_SDCKE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE0)

SW_PAD_CTL Register

Address: 20E_0000h base + 274h offset = 20E_0274h



IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE0 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDCKE0 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_SDCKE0 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_SDCKE0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DRAM_SDCKE0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDCKE0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDCKE0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE0 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DRAM_SDCKE0 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.154 SW_PAD_CTL_PAD_DRAM_SDCKE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE1)

SW_PAD_CTL Register

Address: 20E_0000h base + 278h offset = 20E_0278h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUS		PUE	PKE	Reserved	ODT			Reserved		DSE			Reserved		
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE1 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDCKE1 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_SDCKE1 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_SDCKE1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DRAM_SDCKE1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDCKE1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDCKE1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field

Table continues on the next page...

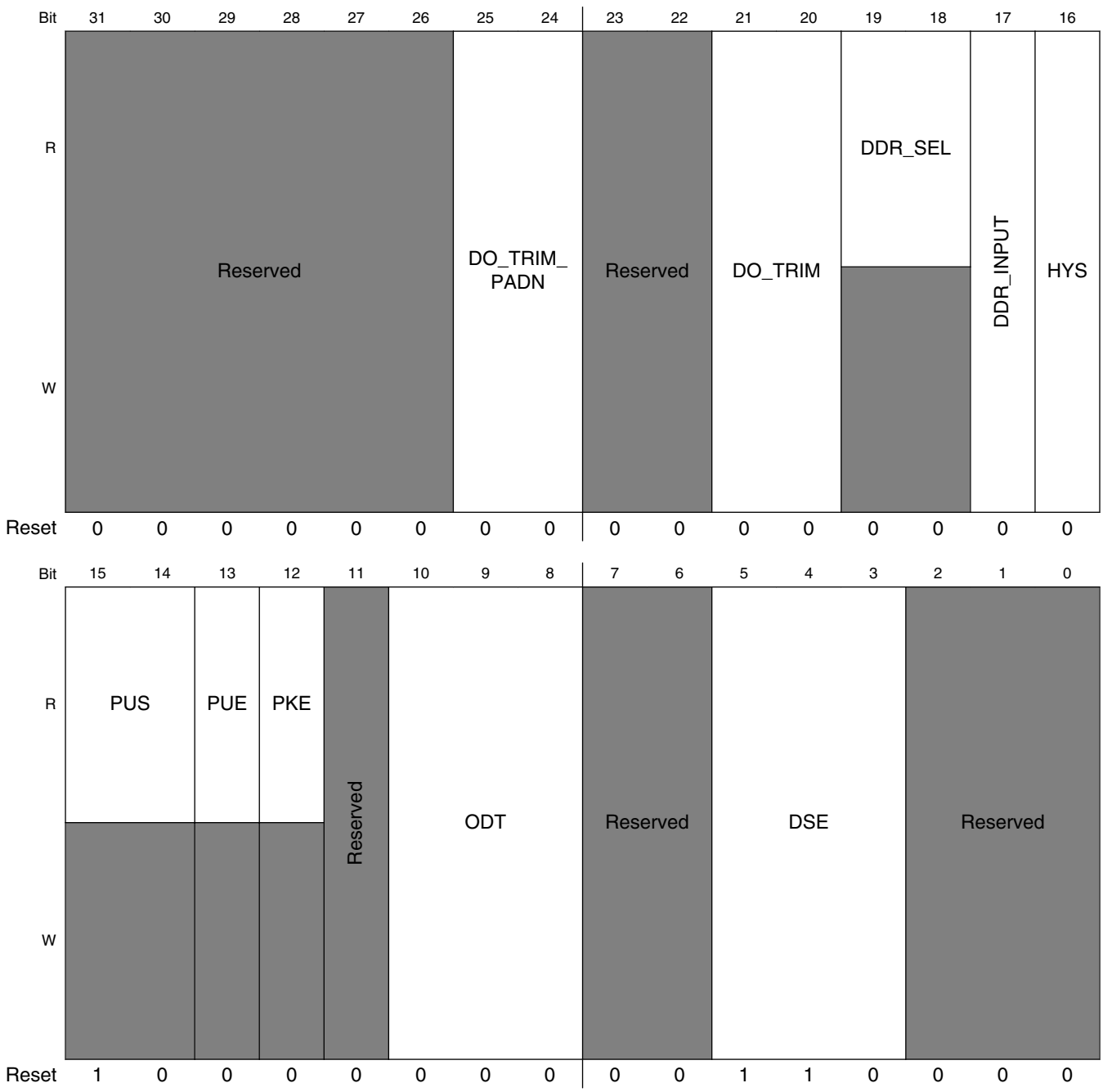
IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE1 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DRAM_SDCKE1 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Read Only Field 0 **DSE** — output driver disabled;
-	This field is reserved. Reserved

30.5.155 SW_PAD_CTL_PAD_DRAM_SDCLK0_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK0_P)

SW_PAD_CTL Register

Address: 20E_0000h base + 27Ch offset = 20E_027Ch



IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK0_P field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–24 DO_TRIM_PADN	do_trim_padn Field Select one out of next values for pad: DRAM_SDCLK0_P 00 **DO_TRIM_PADN_0_min_delay** — min delay 01 **DO_TRIM_PADN_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_PADN_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_PADN_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
23–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDCLK0_P 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_SDCLK0_P 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_SDCLK0_P 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Read Only Field 10 **PUS** — 100K Ohm Pull Up
13 PUE	Pull / Keep Select Field Read Only Field 0 **PUE** — Keeper
12 PKE	Pull / Keep Enable Field Read Only Field

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK0_P field descriptions (continued)

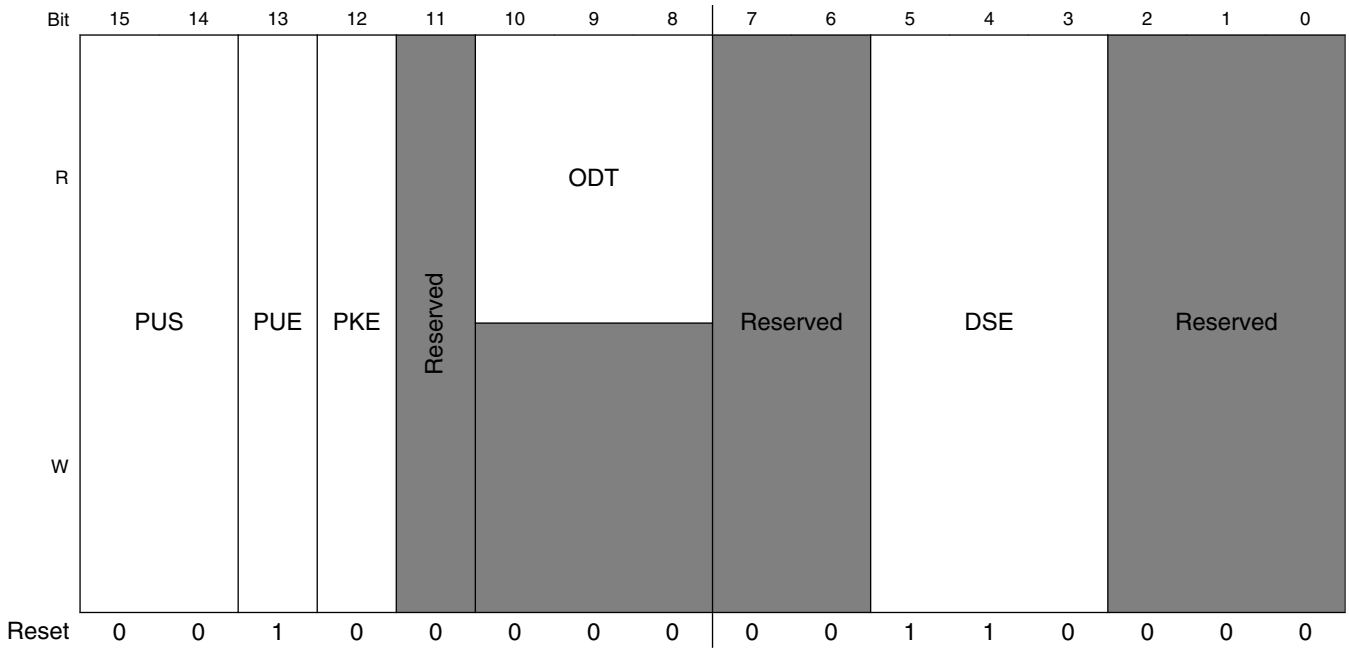
Field	Description
	0 **PKE** — Pull/Keeper Disabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_SDCLK0_P 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDCLK0_P 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.156 SW_PAD_CTL_PAD_DRAM_SDQS0_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS0_P)

SW_PAD_CTL Register

Address: 20E_0000h base + 280h offset = 20E_0280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						DO_TRIM_PADN		Reserved		DO_TRIM		DDR_SEL		DDR_INPUT	HYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS0_P field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–24 DO_TRIM_PADN	do_trim_padn Field Select one out of next values for pad: DRAM_SDQS0_P 00 **DO_TRIM_PADN_0_min_delay** — min delay 01 **DO_TRIM_PADN_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_PADN_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_PADN_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
23–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDQS0_P 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Read Only Field

Table continues on the next page...

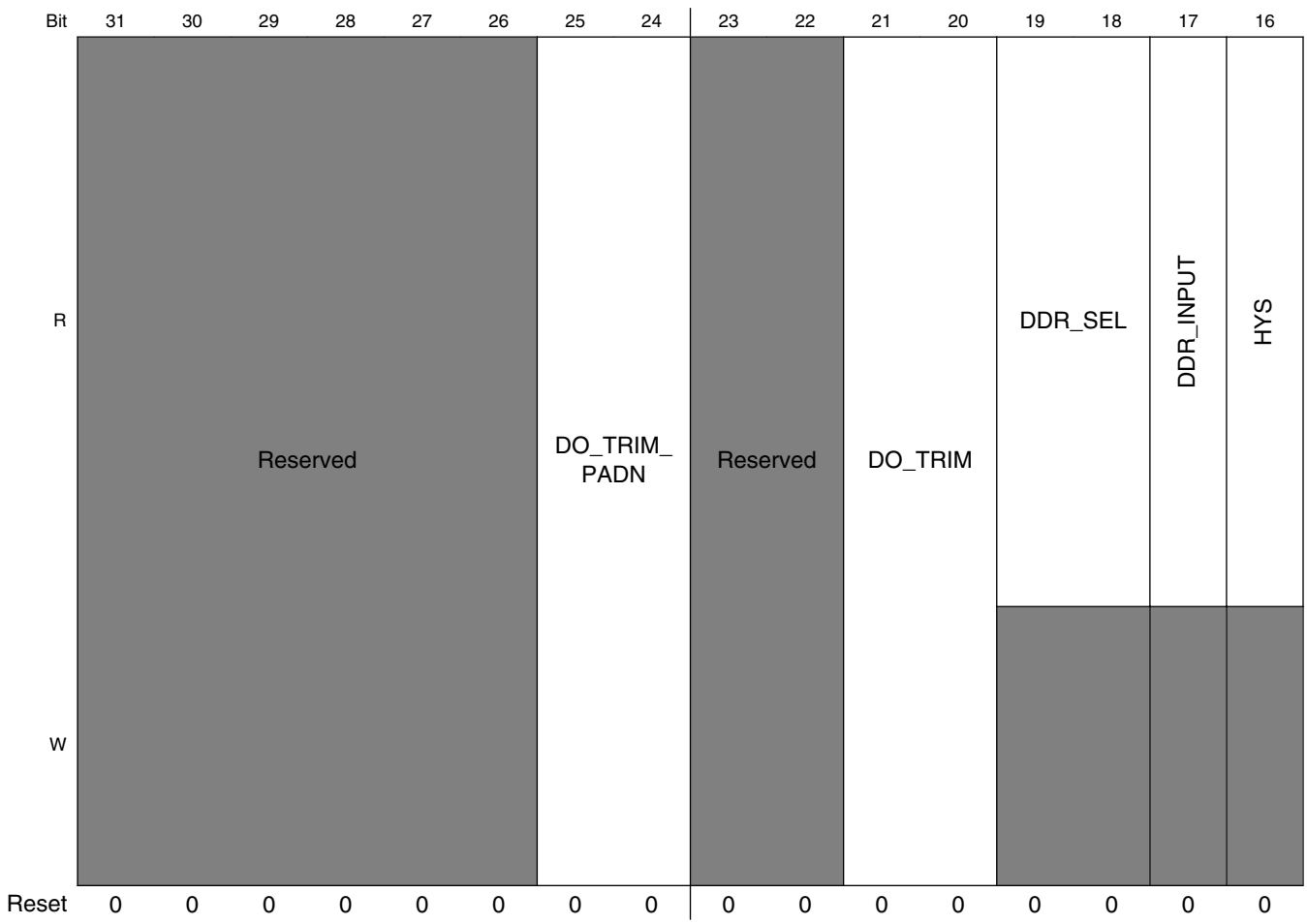
IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS0_P field descriptions (continued)

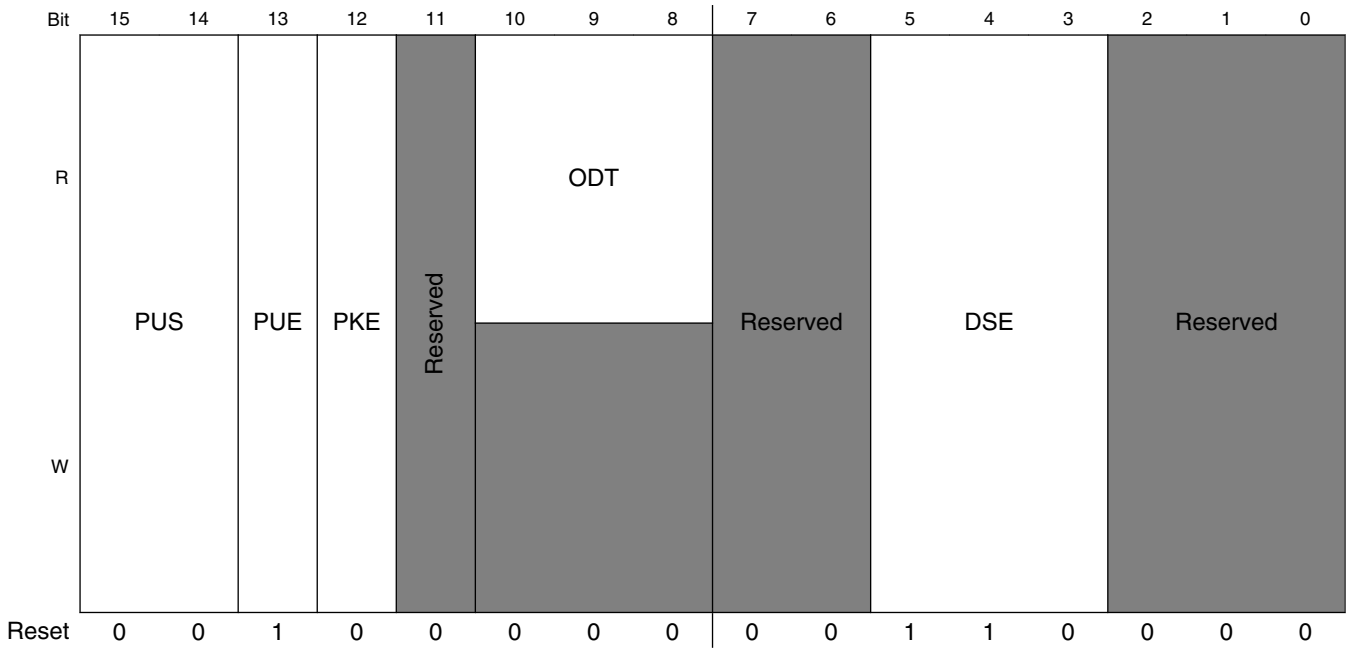
Field	Description
	0 **DDR_INPUT** — CMOS input type
16 HYS	Hyst. Enable Field Read Only Field 0 **HYS** — Hysteresis Disabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DRAM_SDQS0_P 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDQS0_P 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDQS0_P 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Read Only Field 0 **ODT** — off
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDQS0_P 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.157 SW_PAD_CTL_PAD_DRAM_SDQS1_P SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS1_P)

SW_PAD_CTL Register

Address: 20E_0000h base + 284h offset = 20E_0284h





IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS1_P field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–24 DO_TRIM_PADN	do_trim_padn Field Select one out of next values for pad: DRAM_SDQS1_P 00 **DO_TRIM_PADN_0_min_delay** — min delay 01 **DO_TRIM_PADN_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_PADN_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_PADN_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
23–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDQS1_P 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Read Only Field 0 **DDR_SEL** — RESERVED
17 DDR_INPUT	DDR / CMOS Input Mode Field Read Only Field

Table continues on the next page...

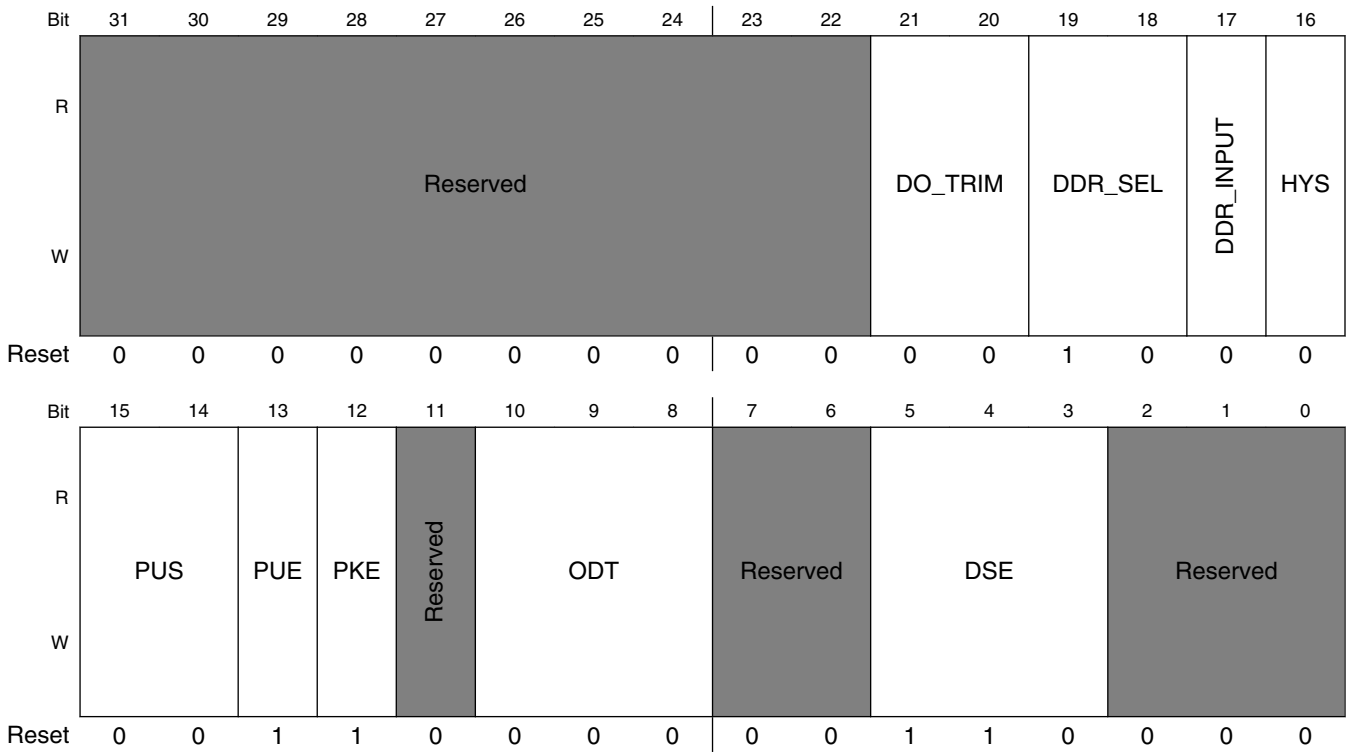
IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS1_P field descriptions (continued)

Field	Description
	0 **DDR_INPUT** — CMOS input type
16 HYS	Hyst. Enable Field Read Only Field 0 **HYS** — Hysteresis Disabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DRAM_SDQS1_P 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDQS1_P 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDQS1_P 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Read Only Field 0 **ODT** — off
7–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDQS1_P 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.158 SW_PAD_CTL_PAD_DRAM_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET)

SW_PAD_CTL Register

Address: 20E_0000h base + 288h offset = 20E_0288h



IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–20 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_RESET 00 **DO_TRIM_0_min_delay** — min delay 01 **DO_TRIM_1_____50ps__ipp_do____pad_delay** — + ~50ps ipp_do -> pad delay 10 **DO_TRIM_2_____100ps_ipp_do____pad_delay** — + ~100ps ipp_do -> pad delay 11 **DO_TRIM_3_____150ps_ipp_do____pad_delay** — + ~150ps ipp_do -> pad delay
19–18 DDR_SEL	ddr_sel Field Select one out of next values for pad: DRAM_RESET 00 **DDR_SEL_0_RESERVED** — RESERVED 01 **DDR_SEL_1_RESERVED** — RESERVED

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET field descriptions (continued)

Field	Description
	10 **DDR_SEL_2_LPDDR2_mode** — LPDDR2 mode 11 **DDR_SEL_3_DDR3_mode** — DDR3 mode
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for pad: DRAM_RESET 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
16 HYS	Hyst. Enable Field Select one out of next values for pad: DRAM_RESET 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DRAM_RESET 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_RESET 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_RESET 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 -	This field is reserved. Reserved
10–8 ODT	On Die Termination Field Select one out of next values for pad: DRAM_RESET 000 **ODT_0_off** — off 001 **ODT_1_120_Ohm_ODT** — 120 Ohm ODT 010 **ODT_2_60_Ohm_ODT** — 60 Ohm ODT 011 **ODT_3_40_Ohm_ODT** — 40 Ohm ODT 100 **ODT_4_30_Ohm_ODT** — 30 Ohm ODT 101 **ODT_5_24_Ohm_ODT** — 24 Ohm ODT 110 **ODT_6_20_Ohm_ODT** — 20 Ohm ODT 111 **ODT_7_17_Ohm_ODT** — 17 Ohm ODT
7–6 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET field descriptions (continued)

Field	Description
5–3 DSE	Drive Strength Field Select one out of next values for pad: DRAM_RESET 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.159 SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_TEST_MODE)

SW_PAD_CTL Register

Address: 20E_0000h base + 28Ch offset = 20E_028Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_TEST_MODE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: TEST_MODE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_TEST_MODE field descriptions (continued)

Field	Description
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: TEST_MODE 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: TEST_MODE 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: TEST_MODE 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: TEST_MODE 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: TEST_MODE 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: TEST_MODE

Table continues on the next page...

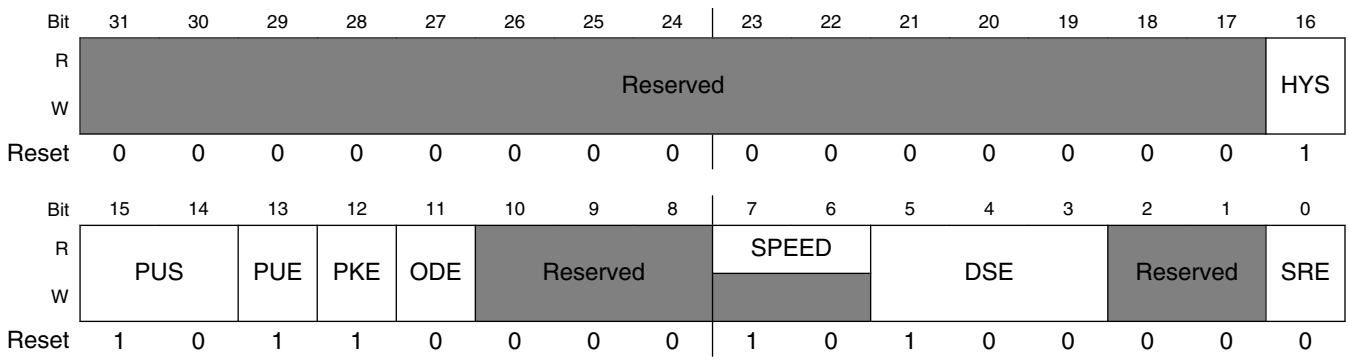
IOMUXC_SW_PAD_CTL_PAD_TEST_MODE field descriptions (continued)

Field	Description
	0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.160 SW_PAD_CTL_PAD_POR_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_POR_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 290h offset = 20E_0290h



IOMUXC_SW_PAD_CTL_PAD_POR_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: POR_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: POR_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: POR_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull

Table continues on the next page...

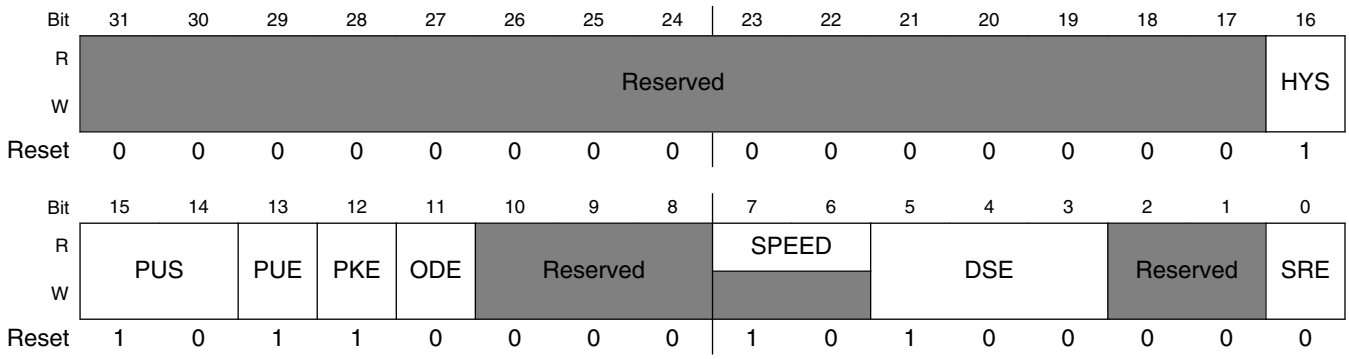
IOMUXC_SW_PAD_CTL_PAD_POR_B field descriptions (continued)

Field	Description
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: POR_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: POR_B 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: POR_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: POR_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.161 SW_PAD_CTL_PAD_ONOFF SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ONOFF)

SW_PAD_CTL Register

Address: 20E_0000h base + 294h offset = 20E_0294h



IOMUXC_SW_PAD_CTL_PAD_ONOFF field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ONOFF 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ONOFF 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ONOFF 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ONOFF 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

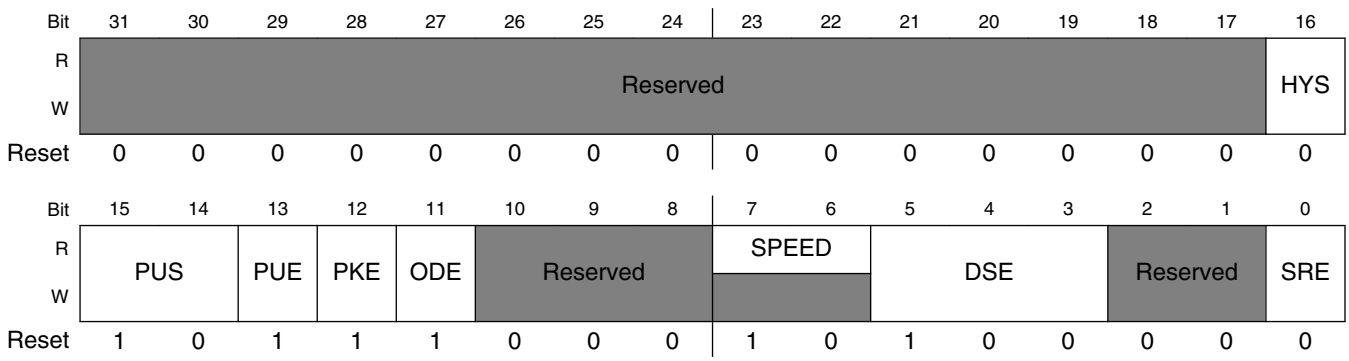
IOMUXC_SW_PAD_CTL_PAD_ONOFF field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: ONOFF 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ONOFF 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ONOFF 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.162 SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ)

SW_PAD_CTL Register

Address: 20E_0000h base + 298h offset = 20E_0298h



IOMUXC_SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_PMIC_ON_REQ 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_PMIC_ON_REQ 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_PMIC_ON_REQ 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_PMIC_ON_REQ 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

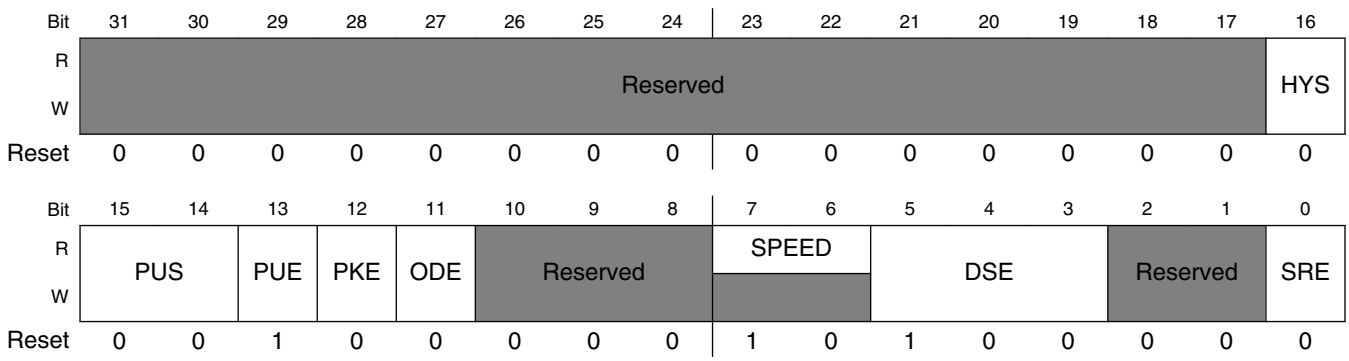
IOMUXC_SW_PAD_CTL_PAD_SNVS_PMIC_ON_REQ field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_PMIC_ON_REQ 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_PMIC_ON_REQ 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_PMIC_ON_REQ 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.163 SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ)

SW_PAD_CTL Register

Address: 20E_0000h base + 29Ch offset = 20E_029Ch



IOMUXC_SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CCM_PMIC_STBY_REQ 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CCM_PMIC_STBY_REQ 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CCM_PMIC_STBY_REQ 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CCM_PMIC_STBY_REQ 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CCM_PMIC_STBY_REQ field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: CCM_PMIC_STBY_REQ 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CCM_PMIC_STBY_REQ 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CCM_PMIC_STBY_REQ 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.164 SW_PAD_CTL_PAD_BOOT_MODE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE0)

SW_PAD_CTL Register

Address: 20E_0000h base + 2A0h offset = 20E_02A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE0 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: BOOT_MODE0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: BOOT_MODE0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: BOOT_MODE0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: BOOT_MODE0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE0 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: BOOT_MODE0 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: BOOT_MODE0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: BOOT_MODE0 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.165 SW_PAD_CTL_PAD_BOOT_MODE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE1)

SW_PAD_CTL Register

Address: 20E_0000h base + 2A4h offset = 20E_02A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE1 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: BOOT_MODE1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: BOOT_MODE1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: BOOT_MODE1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: BOOT_MODE1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

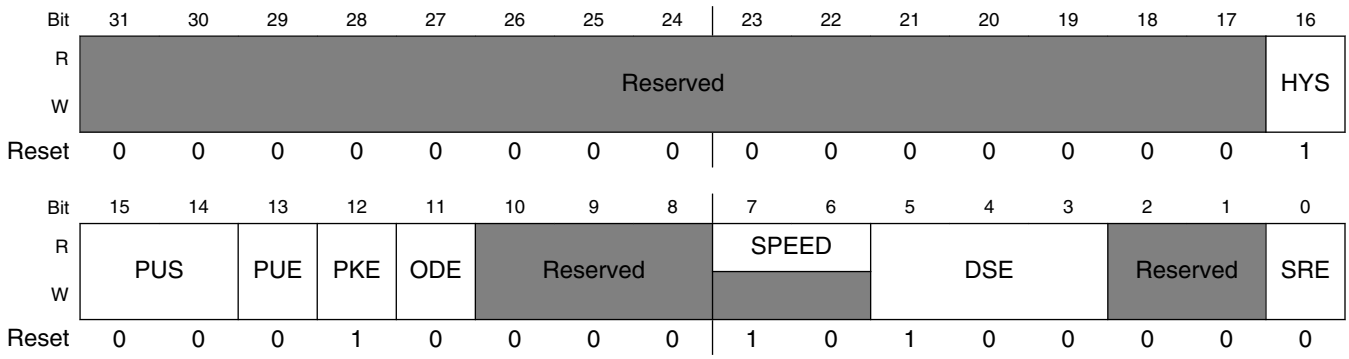
IOMUXC_SW_PAD_CTL_PAD_BOOT_MODE1 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: BOOT_MODE1 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: BOOT_MODE1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: BOOT_MODE1 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.166 SW_PAD_CTL_PAD_SNVS_TAMPER0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER0)

SW_PAD_CTL Register

Address: 20E_0000h base + 2A8h offset = 20E_02A8h



IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER0 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

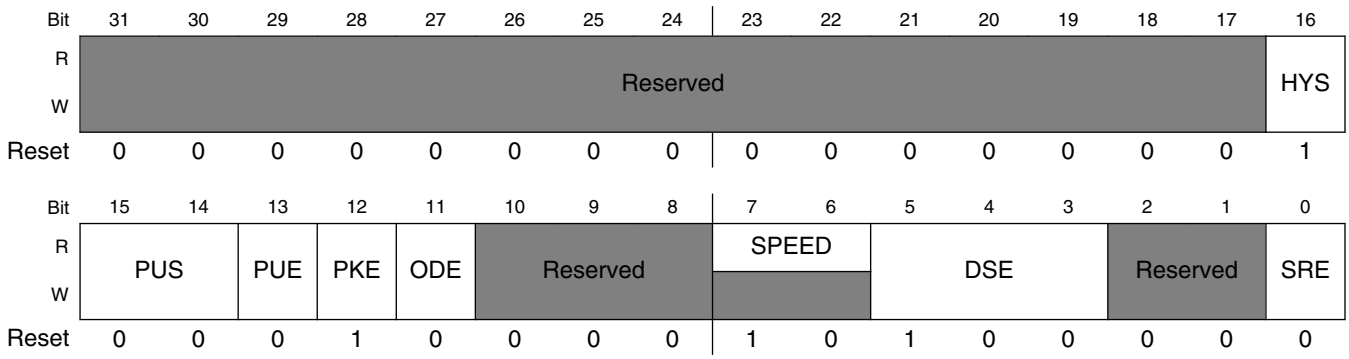
IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER0 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER0 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER0 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.167 SW_PAD_CTL_PAD_SNVS_TAMPER1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER1)

SW_PAD_CTL Register

Address: 20E_0000h base + 2ACh offset = 20E_02ACh



IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER1 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

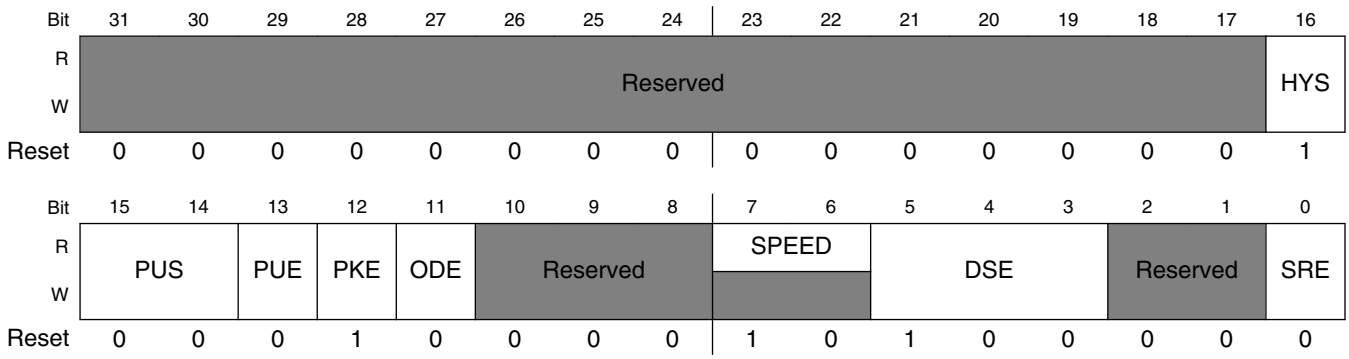
IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER1 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER1 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER1 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.168 SW_PAD_CTL_PAD_SNVS_TAMPER2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER2)

SW_PAD_CTL Register

Address: 20E_0000h base + 2B0h offset = 20E_02B0h



IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER2 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER2 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER2 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER2 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER2 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER2 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER2 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER2 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.169 SW_PAD_CTL_PAD_SNVS_TAMPER3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER3)

SW_PAD_CTL Register

Address: 20E_0000h base + 2B4h offset = 20E_02B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER3 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER3 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER3 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER3 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

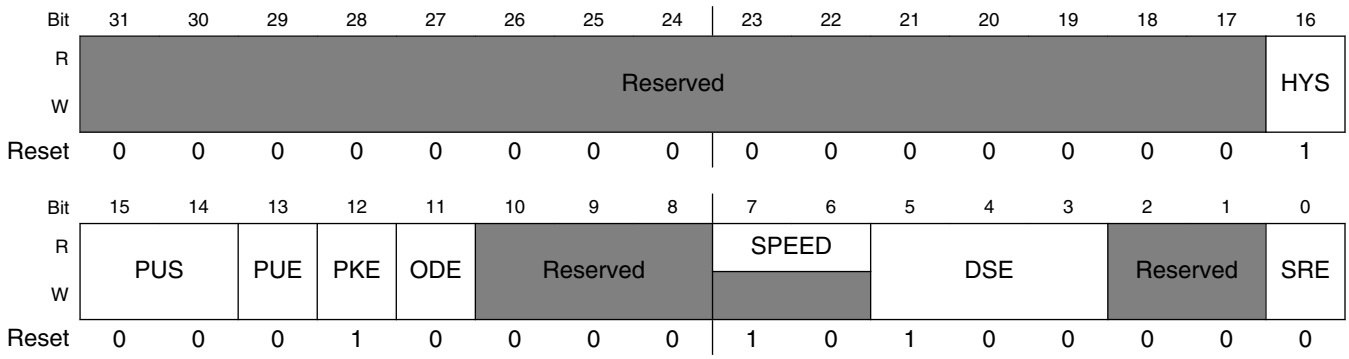
IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER3 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER3 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER3 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER3 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.170 SW_PAD_CTL_PAD_SNVS_TAMPER4 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER4)

SW_PAD_CTL Register

Address: 20E_0000h base + 2B8h offset = 20E_02B8h



IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER4 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER4 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER4 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER4 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER4 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

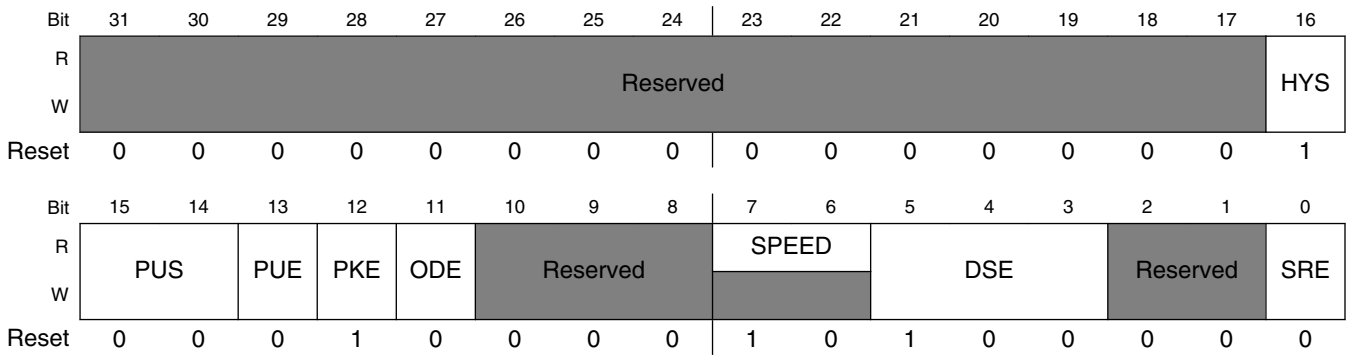
IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER4 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER4 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER4 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER4 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.171 SW_PAD_CTL_PAD_SNVS_TAMPER5 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER5)

SW_PAD_CTL Register

Address: 20E_0000h base + 2BCh offset = 20E_02BCh



IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER5 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER5 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER5 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER5 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER5 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER5 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER5 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER5 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER5 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.172 SW_PAD_CTL_PAD_SNVS_TAMPER6 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER6)

SW_PAD_CTL Register

Address: 20E_0000h base + 2C0h offset = 20E_02C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER6 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER6 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER6 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER6 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER6 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

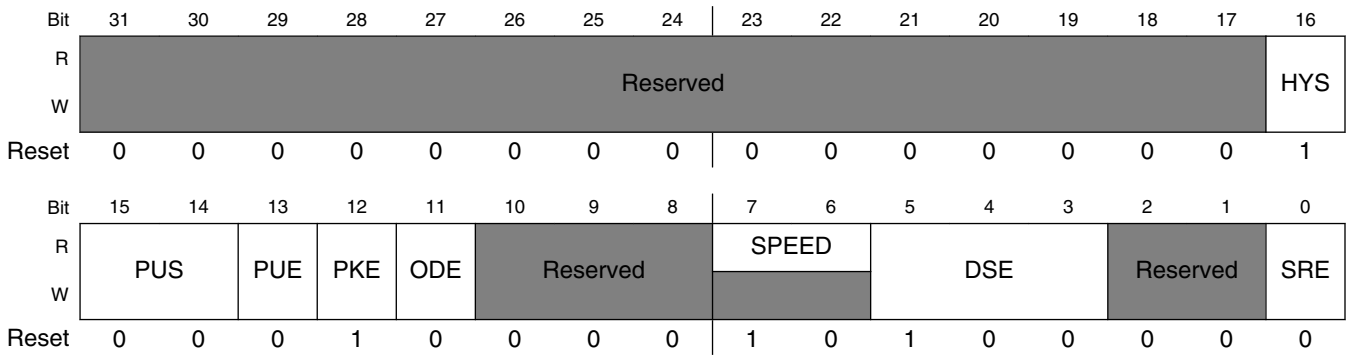
IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER6 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER6 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER6 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER6 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.173 SW_PAD_CTL_PAD_SNVS_TAMPER7 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER7)

SW_PAD_CTL Register

Address: 20E_0000h base + 2C4h offset = 20E_02C4h



IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER7 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER7 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER7 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER7 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER7 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

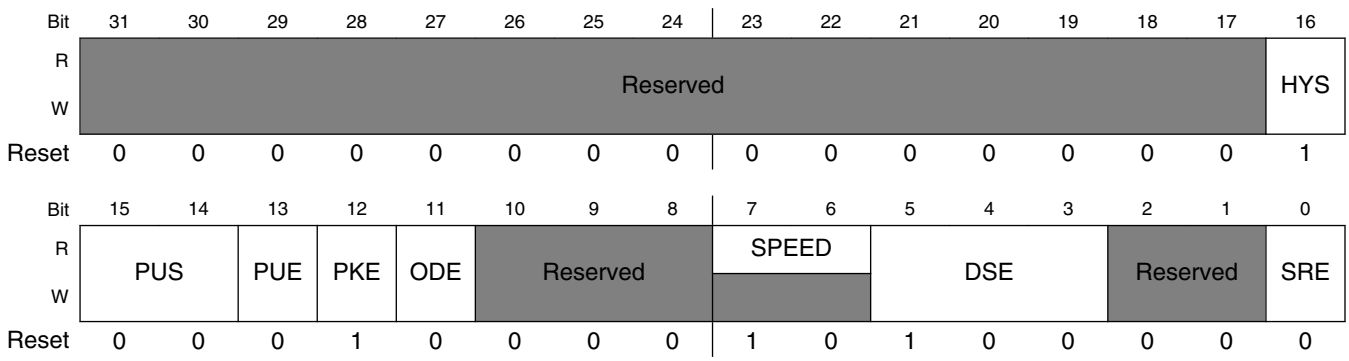
IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER7 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER7 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER7 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER7 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.174 SW_PAD_CTL_PAD_SNVS_TAMPER8 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER8)

SW_PAD_CTL Register

Address: 20E_0000h base + 2C8h offset = 20E_02C8h



IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER8 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER8 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER8 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER8 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER8 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER8 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER8 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER8 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER8 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.175 SW_PAD_CTL_PAD_SNVS_TAMPER9 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER9)

SW_PAD_CTL Register

Address: 20E_0000h base + 2CCh offset = 20E_02CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER9 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SNVS_TAMPER9 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SNVS_TAMPER9 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SNVS_TAMPER9 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SNVS_TAMPER9 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SNVS_TAMPER9 field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: SNVS_TAMPER9 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SNVS_TAMPER9 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SNVS_TAMPER9 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.176 SW_PAD_CTL_PAD_JTAG_MOD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD)

SW_PAD_CTL Register

Address: 20E_0000h base + 2D0h offset = 20E_02D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	1	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_MOD 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: JTAG_MOD 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: JTAG_MOD 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_MOD 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: JTAG_MOD 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: JTAG_MOD 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: JTAG_MOD 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.177 SW_PAD_CTL_PAD_JTAG_TMS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS)

SW_PAD_CTL Register

Address: 20E_0000h base + 2D4h offset = 20E_02D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TMS 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: JTAG_TMS 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: JTAG_TMS 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TMS 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: JTAG_TMS 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: JTAG_TMS 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: JTAG_TMS 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.178 SW_PAD_CTL_PAD_JTAG_TDO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO)

SW_PAD_CTL Register

Address: 20E_0000h base + 2D8h offset = 20E_02D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	1	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1

IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TDO 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: JTAG_TDO 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: JTAG_TDO 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TDO 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: JTAG_TDO 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: JTAG_TDO 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: JTAG_TDO 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.179 SW_PAD_CTL_PAD_JTAG_TDI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI)

SW_PAD_CTL Register

Address: 20E_0000h base + 2DCh offset = 20E_02DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TDI 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: JTAG_TDI 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: JTAG_TDI 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TDI 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: JTAG_TDI 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: JTAG_TDI 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: JTAG_TDI 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.180 SW_PAD_CTL_PAD_JTAG_TCK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK)

SW_PAD_CTL Register

Address: 20E_0000h base + 2E0h offset = 20E_02E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TCK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: JTAG_TCK 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: JTAG_TCK 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TCK 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: JTAG_TCK 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: JTAG_TCK 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: JTAG_TCK 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.181 SW_PAD_CTL_PAD_JTAG_TRST_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 2E4h offset = 20E_02E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TRST_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: JTAG_TRST_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: JTAG_TRST_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TRST_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: JTAG_TRST_B 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Read Only Field 10 **SPEED** — medium(100 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: JTAG_TRST_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: JTAG_TRST_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.182 SW_PAD_CTL_PAD_GPIO1_IO00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO00)

SW_PAD_CTL Register

Address: 20E_0000h base + 2E8h offset = 20E_02E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO00 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO00 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO00 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.183 SW_PAD_CTL_PAD_GPIO1_IO01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO01)

SW_PAD_CTL Register

Address: 20E_0000h base + 2ECh offset = 20E_02ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO01 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO01 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO01 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.184 SW_PAD_CTL_PAD_GPIO1_IO02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO02)

SW_PAD_CTL Register

Address: 20E_0000h base + 2F0h offset = 20E_02F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO02 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO02 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO02 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.185 SW_PAD_CTL_PAD_GPIO1_IO03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO03)

SW_PAD_CTL Register

Address: 20E_0000h base + 2F4h offset = 20E_02F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO03 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO03 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO03 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.186 SW_PAD_CTL_PAD_GPIO1_IO04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO04)

SW_PAD_CTL Register

Address: 20E_0000h base + 2F8h offset = 20E_02F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO04 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO04 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.187 SW_PAD_CTL_PAD_GPIO1_IO05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO05)

SW_PAD_CTL Register

Address: 20E_0000h base + 2FCh offset = 20E_02FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO05 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO05 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO05 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.188 SW_PAD_CTL_PAD_GPIO1_IO06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO06)

SW_PAD_CTL Register

Address: 20E_0000h base + 300h offset = 20E_0300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO06

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO06 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO06 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO06 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.189 SW_PAD_CTL_PAD_GPIO1_IO07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO07)

SW_PAD_CTL Register

Address: 20E_0000h base + 304h offset = 20E_0304h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO07 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO07 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO07 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.190 SW_PAD_CTL_PAD_GPIO1_IO08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08)

SW_PAD_CTL Register

Address: 20E_0000h base + 308h offset = 20E_0308h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO08 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO08 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.191 SW_PAD_CTL_PAD_GPIO1_IO09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09)

SW_PAD_CTL Register

Address: 20E_0000h base + 30Ch offset = 20E_030Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO1_IO09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO1_IO09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO1_IO09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO1_IO09

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO1_IO09 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO09 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.192 SW_PAD_CTL_PAD_UART1_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 310h offset = 20E_0310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART1_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_TX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_TX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_TX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_TX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART1_TX_DATA 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART1_TX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART1_TX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.193 SW_PAD_CTL_PAD_UART1_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 314h offset = 20E_0314h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART1_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_RX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_RX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_RX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_RX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART1_RX_DATA 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART1_RX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART1_RX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.194 SW_PAD_CTL_PAD_UART1_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_CTS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 318h offset = 20E_0318h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART1_CTS_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART1_CTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_CTS_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_CTS_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_CTS_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_CTS_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART1_CTS_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART1_CTS_B 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART1_CTS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART1_CTS_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.195 SW_PAD_CTL_PAD_UART1_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RTS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 31Ch offset = 20E_031Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART1_RTS_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART1_RTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_RTS_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_RTS_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_RTS_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_RTS_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART1_RTS_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART1_RTS_B 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART1_RTS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART1_RTS_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.196 SW_PAD_CTL_PAD_UART2_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 320h offset = 20E_0320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART2_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_TX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_TX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_TX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_TX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART2_TX_DATA 00 **SPEED_0_low_50MHz_** — low(50 MHz) 01 **SPEED_1_medium_100MHz_** — medium(100 MHz) 10 **SPEED_2_medium_100MHz_** — medium(100 MHz) 11 **SPEED_3_max_200MHz_** — max(200 MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART2_TX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART2_TX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.197 SW_PAD_CTL_PAD_UART2_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 324h offset = 20E_0324h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART2_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_RX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_RX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_RX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_RX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART2_RX_DATA 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART2_RX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART2_RX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.198 SW_PAD_CTL_PAD_UART2_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_CTS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 328h offset = 20E_0328h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART2_CTS_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART2_CTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_CTS_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_CTS_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_CTS_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_CTS_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART2_CTS_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART2_CTS_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART2_CTS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART2_CTS_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.199 SW_PAD_CTL_PAD_UART2_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RTS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 32Ch offset = 20E_032Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART2_RTS_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART2_RTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_RTS_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_RTS_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_RTS_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_RTS_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART2_RTS_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART2_RTS_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART2_RTS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART2_RTS_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.200 SW_PAD_CTL_PAD_UART3_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 330h offset = 20E_0330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART3_TX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART3_TX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART3_TX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART3_TX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART3_TX_DATA 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART3_TX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART3_TX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.201 SW_PAD_CTL_PAD_UART3_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 334h offset = 20E_0334h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART3_RX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART3_RX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART3_RX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART3_RX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART3_RX_DATA 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART3_RX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART3_RX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.202 SW_PAD_CTL_PAD_UART3_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 338h offset = 20E_0338h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_CTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART3_CTS_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART3_CTS_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART3_CTS_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART3_CTS_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART3_CTS_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART3_CTS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART3_CTS_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.203 SW_PAD_CTL_PAD_UART3_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 33Ch offset = 20E_033Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_RTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART3_RTS_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART3_RTS_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART3_RTS_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART3_RTS_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART3_RTS_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART3_RTS_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART3_RTS_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.204 SW_PAD_CTL_PAD_UART4_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART4_TX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 340h offset = 20E_0340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART4_TX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART4_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART4_TX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART4_TX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART4_TX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART4_TX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART4_TX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART4_TX_DATA 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART4_TX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART4_TX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.205 SW_PAD_CTL_PAD_UART4_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART4_RX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 344h offset = 20E_0344h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART4_RX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART4_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART4_RX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART4_RX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART4_RX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART4_RX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART4_RX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART4_RX_DATA 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART4_RX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART4_RX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.206 SW_PAD_CTL_PAD_UART5_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART5_TX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 348h offset = 20E_0348h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART5_TX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART5_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART5_TX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART5_TX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART5_TX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART5_TX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART5_TX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART5_TX_DATA 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART5_TX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART5_TX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.207 SW_PAD_CTL_PAD_UART5_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART5_RX_DATA)

SW_PAD_CTL Register

Address: 20E_0000h base + 34Ch offset = 20E_034Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_UART5_RX_DATA field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: UART5_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART5_RX_DATA 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: UART5_RX_DATA 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART5_RX_DATA 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: UART5_RX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART5_RX_DATA field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: UART5_RX_DATA 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: UART5_RX_DATA 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART5_RX_DATA 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.208 SW_PAD_CTL_PAD_ENET1_RX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA0)

SW_PAD_CTL Register

Address: 20E_0000h base + 350h offset = 20E_0350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA0 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RX_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_RX_DATA0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_RX_DATA0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_RX_DATA0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_RX_DATA0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA0 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_RX_DATA0 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_RX_DATA0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RX_DATA0 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.209 SW_PAD_CTL_PAD_ENET1_RX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA1)

SW_PAD_CTL Register

Address: 20E_0000h base + 354h offset = 20E_0354h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA1 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RX_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_RX_DATA1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_RX_DATA1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_RX_DATA1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_RX_DATA1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_DATA1 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_RX_DATA1 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_RX_DATA1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RX_DATA1 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.210 SW_PAD_CTL_PAD_ENET1_RX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_EN)

SW_PAD_CTL Register

Address: 20E_0000h base + 358h offset = 20E_0358h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_EN field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RX_EN 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_RX_EN 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_RX_EN 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_RX_EN 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_RX_EN

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_EN field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_RX_EN 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_RX_EN 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RX_EN 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.211 SW_PAD_CTL_PAD_ENET1_TX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA0)

SW_PAD_CTL Register

Address: 20E_0000h base + 35Ch offset = 20E_035Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA0 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_TX_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_TX_DATA0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_TX_DATA0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_TX_DATA0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_TX_DATA0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA0 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_TX_DATA0 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_TX_DATA0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_TX_DATA0 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.212 SW_PAD_CTL_PAD_ENET1_TX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA1)

SW_PAD_CTL Register

Address: 20E_0000h base + 360h offset = 20E_0360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA1 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_TX_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_TX_DATA1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_TX_DATA1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_TX_DATA1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_TX_DATA1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_DATA1 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_TX_DATA1 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_TX_DATA1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_TX_DATA1 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.213 SW_PAD_CTL_PAD_ENET1_TX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_EN)

SW_PAD_CTL Register

Address: 20E_0000h base + 364h offset = 20E_0364h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_EN field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_TX_EN 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_TX_EN 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_TX_EN 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_TX_EN 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_TX_EN

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_EN field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_TX_EN 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_TX_EN 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_TX_EN 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.214 SW_PAD_CTL_PAD_ENET1_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK)

SW_PAD_CTL Register

Address: 20E_0000h base + 368h offset = 20E_0368h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_TX_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_TX_CLK 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_TX_CLK 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_TX_CLK 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_TX_CLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_TX_CLK 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_TX_CLK 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_TX_CLK 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.215 SW_PAD_CTL_PAD_ENET1_RX_ER SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_ER)

SW_PAD_CTL Register

Address: 20E_0000h base + 36Ch offset = 20E_036Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_ER field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RX_ER 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET1_RX_ER 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET1_RX_ER 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET1_RX_ER 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET1_RX_ER

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_ER field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET1_RX_ER 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET1_RX_ER 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RX_ER 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.216 SW_PAD_CTL_PAD_ENET2_RX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA0)

SW_PAD_CTL Register

Address: 20E_0000h base + 370h offset = 20E_0370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA0 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_RX_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_RX_DATA0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_RX_DATA0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_RX_DATA0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_RX_DATA0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA0 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_RX_DATA0 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_RX_DATA0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_RX_DATA0 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.217 SW_PAD_CTL_PAD_ENET2_RX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA1)

SW_PAD_CTL Register

Address: 20E_0000h base + 374h offset = 20E_0374h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA1 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_RX_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_RX_DATA1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_RX_DATA1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_RX_DATA1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_RX_DATA1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_DATA1 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_RX_DATA1 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_RX_DATA1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_RX_DATA1 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.218 SW_PAD_CTL_PAD_ENET2_RX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_EN)

SW_PAD_CTL Register

Address: 20E_0000h base + 378h offset = 20E_0378h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_EN field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_RX_EN 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_RX_EN 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_RX_EN 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_RX_EN 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_RX_EN

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_EN field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_RX_EN 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_RX_EN 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_RX_EN 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.219 SW_PAD_CTL_PAD_ENET2_TX_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA0)

SW_PAD_CTL Register

Address: 20E_0000h base + 37Ch offset = 20E_037Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA0 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_TX_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_TX_DATA0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_TX_DATA0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_TX_DATA0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_TX_DATA0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA0 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_TX_DATA0 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_TX_DATA0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_TX_DATA0 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.220 SW_PAD_CTL_PAD_ENET2_TX_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA1)

SW_PAD_CTL Register

Address: 20E_0000h base + 380h offset = 20E_0380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA1 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_TX_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_TX_DATA1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_TX_DATA1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_TX_DATA1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_TX_DATA1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_DATA1 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_TX_DATA1 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_TX_DATA1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_TX_DATA1 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.221 SW_PAD_CTL_PAD_ENET2_TX_EN SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_EN)

SW_PAD_CTL Register

Address: 20E_0000h base + 384h offset = 20E_0384h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_EN field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_TX_EN 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_TX_EN 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_TX_EN 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_TX_EN 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_TX_EN

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_EN field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_TX_EN 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_TX_EN 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_TX_EN 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.222 SW_PAD_CTL_PAD_ENET2_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_CLK)

SW_PAD_CTL Register

Address: 20E_0000h base + 388h offset = 20E_0388h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_CLK field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_TX_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_TX_CLK 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_TX_CLK 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_TX_CLK 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_TX_CLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_TX_CLK field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_TX_CLK 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_TX_CLK 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_TX_CLK 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.223 SW_PAD_CTL_PAD_ENET2_RX_ER SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_ER)

SW_PAD_CTL Register

Address: 20E_0000h base + 38Ch offset = 20E_038Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_ER field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ENET2_RX_ER 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ENET2_RX_ER 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: ENET2_RX_ER 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ENET2_RX_ER 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ENET2_RX_ER

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET2_RX_ER field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: ENET2_RX_ER 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ENET2_RX_ER 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ENET2_RX_ER 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.224 SW_PAD_CTL_PAD_LCD_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_CLK)

SW_PAD_CTL Register

Address: 20E_0000h base + 390h offset = 20E_0390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_CLK field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_CLK 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_CLK 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_CLK 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_CLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_CLK field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_CLK 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_CLK 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_CLK 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.225 SW_PAD_CTL_PAD_LCD_ENABLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE)

SW_PAD_CTL Register

Address: 20E_0000h base + 394h offset = 20E_0394h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_ENABLE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_ENABLE 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_ENABLE 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_ENABLE 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_ENABLE

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_ENABLE 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_ENABLE 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_ENABLE 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.226 SW_PAD_CTL_PAD_LCD_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC)

SW_PAD_CTL Register

Address: 20E_0000h base + 398h offset = 20E_0398h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_HSYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_HSYNC 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_HSYNC 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_HSYNC 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_HSYNC

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_HSYNC 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_HSYNC 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_HSYNC 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.227 SW_PAD_CTL_PAD_LCD_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC)

SW_PAD_CTL Register

Address: 20E_0000h base + 39Ch offset = 20E_039Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_VSYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_VSYNC 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_VSYNC 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_VSYNC 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_VSYNC

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_VSYNC 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_VSYNC 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_VSYNC 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.228 SW_PAD_CTL_PAD_LCD_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_RESET)

SW_PAD_CTL Register

Address: 20E_0000h base + 3A0h offset = 20E_03A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_RESET field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_RESET 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_RESET 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_RESET 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_RESET 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_RESET

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_RESET field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_RESET 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_RESET 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_RESET 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.229 SW_PAD_CTL_PAD_LCD_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00)

SW_PAD_CTL Register

Address: 20E_0000h base + 3A4h offset = 20E_03A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA00 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA00 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.230 SW_PAD_CTL_PAD_LCD_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01)

SW_PAD_CTL Register

Address: 20E_0000h base + 3A8h offset = 20E_03A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA01 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA01 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.231 SW_PAD_CTL_PAD_LCD_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02)

SW_PAD_CTL Register

Address: 20E_0000h base + 3ACh offset = 20E_03ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA02 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA02 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.232 SW_PAD_CTL_PAD_LCD_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03)

SW_PAD_CTL Register

Address: 20E_0000h base + 3B0h offset = 20E_03B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA03 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA03 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.233 SW_PAD_CTL_PAD_LCD_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04)

SW_PAD_CTL Register

Address: 20E_0000h base + 3B4h offset = 20E_03B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA04 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA04 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.234 SW_PAD_CTL_PAD_LCD_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05)

SW_PAD_CTL Register

Address: 20E_0000h base + 3B8h offset = 20E_03B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA05 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA05 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.235 SW_PAD_CTL_PAD_LCD_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06)

SW_PAD_CTL Register

Address: 20E_0000h base + 3BCh offset = 20E_03BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA06

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA06 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA06 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.236 SW_PAD_CTL_PAD_LCD_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07)

SW_PAD_CTL Register

Address: 20E_0000h base + 3C0h offset = 20E_03C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA07 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA07 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.237 SW_PAD_CTL_PAD_LCD_DATA08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08)

SW_PAD_CTL Register

Address: 20E_0000h base + 3C4h offset = 20E_03C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA08 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA08 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.238 SW_PAD_CTL_PAD_LCD_DATA09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09)

SW_PAD_CTL Register

Address: 20E_0000h base + 3C8h offset = 20E_03C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA09

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA09 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA09 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.239 SW_PAD_CTL_PAD_LCD_DATA10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10)

SW_PAD_CTL Register

Address: 20E_0000h base + 3CCh offset = 20E_03CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA10 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA10 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA10 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA10

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA10 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA10 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.240 SW_PAD_CTL_PAD_LCD_DATA11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11)

SW_PAD_CTL Register

Address: 20E_0000h base + 3D0h offset = 20E_03D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA11 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA11 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA11 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA11

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA11 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA11 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.241 SW_PAD_CTL_PAD_LCD_DATA12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12)

SW_PAD_CTL Register

Address: 20E_0000h base + 3D4h offset = 20E_03D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA12 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA12 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA12 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA12

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA12 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA12 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.242 SW_PAD_CTL_PAD_LCD_DATA13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13)

SW_PAD_CTL Register

Address: 20E_0000h base + 3D8h offset = 20E_03D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA13 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA13 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA13 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA13

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA13 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA13 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.243 SW_PAD_CTL_PAD_LCD_DATA14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14)

SW_PAD_CTL Register

Address: 20E_0000h base + 3DCh offset = 20E_03DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA14 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA14 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA14 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA14

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA14 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA14 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.244 SW_PAD_CTL_PAD_LCD_DATA15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15)

SW_PAD_CTL Register

Address: 20E_0000h base + 3E0h offset = 20E_03E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA15 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA15 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA15 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA15

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA15 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA15 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA15 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.245 SW_PAD_CTL_PAD_LCD_DATA16 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16)

SW_PAD_CTL Register

Address: 20E_0000h base + 3E4h offset = 20E_03E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA16 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA16 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA16 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA16 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA16

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA16 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA16 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA16 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.246 SW_PAD_CTL_PAD_LCD_DATA17 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17)

SW_PAD_CTL Register

Address: 20E_0000h base + 3E8h offset = 20E_03E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA17 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA17 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA17 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA17 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA17

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA17 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA17 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA17 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.247 SW_PAD_CTL_PAD_LCD_DATA18 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18)

SW_PAD_CTL Register

Address: 20E_0000h base + 3ECh offset = 20E_03ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA18 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA18 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA18 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA18 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA18

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA18 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA18 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA18 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.248 SW_PAD_CTL_PAD_LCD_DATA19 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19)

SW_PAD_CTL Register

Address: 20E_0000h base + 3F0h offset = 20E_03F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA19 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA19 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA19 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA19 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA19

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA19 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA19 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA19 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.249 SW_PAD_CTL_PAD_LCD_DATA20 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20)

SW_PAD_CTL Register

Address: 20E_0000h base + 3F4h offset = 20E_03F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA20 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA20 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA20 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA20 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA20

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA20 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA20 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA20 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.250 SW_PAD_CTL_PAD_LCD_DATA21 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21)

SW_PAD_CTL Register

Address: 20E_0000h base + 3F8h offset = 20E_03F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA21 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA21 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA21 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA21 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA21

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA21 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA21 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA21 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.251 SW_PAD_CTL_PAD_LCD_DATA22 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22)

SW_PAD_CTL Register

Address: 20E_0000h base + 3FCh offset = 20E_03FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA22 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA22 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA22 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA22 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA22

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA22 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA22 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA22 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.252 SW_PAD_CTL_PAD_LCD_DATA23 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23)

SW_PAD_CTL Register

Address: 20E_0000h base + 400h offset = 20E_0400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA23 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LCD_DATA23 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: LCD_DATA23 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: LCD_DATA23 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: LCD_DATA23

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: LCD_DATA23 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA23 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA23 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.253 SW_PAD_CTL_PAD_NAND_RE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_RE_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 404h offset = 20E_0404h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_RE_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_RE_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_RE_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_RE_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_RE_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_RE_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_RE_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_RE_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_RE_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_RE_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.254 SW_PAD_CTL_PAD_NAND_WE_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_WE_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 408h offset = 20E_0408h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_WE_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_WE_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_WE_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_WE_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_WE_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_WE_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_WE_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_WE_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_WE_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_WE_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.255 SW_PAD_CTL_PAD_NAND_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA00)

SW_PAD_CTL Register

Address: 20E_0000h base + 40Ch offset = 20E_040Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA00 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA00 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA00 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.256 SW_PAD_CTL_PAD_NAND_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA01)

SW_PAD_CTL Register

Address: 20E_0000h base + 410h offset = 20E_0410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA01 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA01 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA01 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.257 SW_PAD_CTL_PAD_NAND_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA02)

SW_PAD_CTL Register

Address: 20E_0000h base + 414h offset = 20E_0414h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA02 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA02 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA02 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.258 SW_PAD_CTL_PAD_NAND_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA03)

SW_PAD_CTL Register

Address: 20E_0000h base + 418h offset = 20E_0418h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA03 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA03 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA03 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.259 SW_PAD_CTL_PAD_NAND_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA04)

SW_PAD_CTL Register

Address: 20E_0000h base + 41Ch offset = 20E_041Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA04 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA04 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.260 SW_PAD_CTL_PAD_NAND_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA05)

SW_PAD_CTL Register

Address: 20E_0000h base + 420h offset = 20E_0420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA05 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA05 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA05 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.261 SW_PAD_CTL_PAD_NAND_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA06)

SW_PAD_CTL Register

Address: 20E_0000h base + 424h offset = 20E_0424h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA06

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA06 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA06 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA06 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.262 SW_PAD_CTL_PAD_NAND_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DATA07)

SW_PAD_CTL Register

Address: 20E_0000h base + 428h offset = 20E_0428h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DATA07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DATA07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DATA07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DATA07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DATA07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DATA07 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DATA07 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DATA07 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DATA07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.263 SW_PAD_CTL_PAD_NAND_ALE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_ALE)

SW_PAD_CTL Register

Address: 20E_0000h base + 42Ch offset = 20E_042Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_ALE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_ALE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_ALE 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_ALE 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_ALE 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_ALE

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_ALE field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_ALE 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_ALE 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_ALE 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.264 SW_PAD_CTL_PAD_NAND_WP_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_WP_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 430h offset = 20E_0430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_WP_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_WP_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_WP_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_WP_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_WP_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_WP_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_WP_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_WP_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_WP_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_WP_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.265 SW_PAD_CTL_PAD_NAND_READY_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_READY_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 434h offset = 20E_0434h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_READY_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_READY_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_READY_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_READY_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_READY_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_READY_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_READY_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_READY_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_READY_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_READY_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.266 SW_PAD_CTL_PAD_NAND_CE0_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CE0_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 438h offset = 20E_0438h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_CE0_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_CE0_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_CE0_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_CE0_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_CE0_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_CE0_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_CE0_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_CE0_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_CE0_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_CE0_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.267 SW_PAD_CTL_PAD_NAND_CE1_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CE1_B)

SW_PAD_CTL Register

Address: 20E_0000h base + 43Ch offset = 20E_043Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_CE1_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_CE1_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_CE1_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_CE1_B 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_CE1_B 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_CE1_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_CE1_B field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_CE1_B 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_CE1_B 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_CE1_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.268 SW_PAD_CTL_PAD_NAND_CLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_CLE)

SW_PAD_CTL Register

Address: 20E_0000h base + 440h offset = 20E_0440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_CLE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_CLE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_CLE 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_CLE 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_CLE 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_CLE

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_CLE field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_CLE 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_CLE 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_CLE 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.269 SW_PAD_CTL_PAD_NAND_DQS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_NAND_DQS)

SW_PAD_CTL Register

Address: 20E_0000h base + 444h offset = 20E_0444h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_NAND_DQS field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: NAND_DQS 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: NAND_DQS 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: NAND_DQS 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: NAND_DQS 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: NAND_DQS

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_NAND_DQS field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: NAND_DQS 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: NAND_DQS 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NAND_DQS 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.270 SW_PAD_CTL_PAD_SD1_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CMD)

SW_PAD_CTL Register

Address: 20E_0000h base + 448h offset = 20E_0448h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_CMD field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_CMD 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_CMD 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_CMD 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_CMD 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_CMD

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_CMD field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: SD1_CMD 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SD1_CMD 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_CMD 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.271 SW_PAD_CTL_PAD_SD1_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CLK)

SW_PAD_CTL Register

Address: 20E_0000h base + 44Ch offset = 20E_044Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_CLK field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_CLK 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_CLK 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_CLK 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_CLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_CLK field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: SD1_CLK 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SD1_CLK 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_CLK 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.272 SW_PAD_CTL_PAD_SD1_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0)

SW_PAD_CTL Register

Address: 20E_0000h base + 450h offset = 20E_0450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA0 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA0 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA0 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_DATA0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: SD1_DATA0 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA0 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA0 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.273 SW_PAD_CTL_PAD_SD1_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1)

SW_PAD_CTL Register

Address: 20E_0000h base + 454h offset = 20E_0454h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA1 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA1 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA1 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_DATA1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: SD1_DATA1 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA1 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA1 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.274 SW_PAD_CTL_PAD_SD1_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2)

SW_PAD_CTL Register

Address: 20E_0000h base + 458h offset = 20E_0458h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA2 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA2 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA2 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_DATA2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: SD1_DATA2 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA2 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA2 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.275 SW_PAD_CTL_PAD_SD1_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3)

SW_PAD_CTL Register

Address: 20E_0000h base + 45Ch offset = 20E_045Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA3 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA3 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA3 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_DATA3

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: SD1_DATA3 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA3 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA3 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.276 SW_PAD_CTL_PAD_CSI_MCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK)

SW_PAD_CTL Register

Address: 20E_0000h base + 460h offset = 20E_0460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_MCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_MCLK 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_MCLK 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_MCLK 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_MCLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_MCLK 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_MCLK 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_MCLK 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.277 SW_PAD_CTL_PAD_CSI_PIXCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK)

SW_PAD_CTL Register

Address: 20E_0000h base + 464h offset = 20E_0464h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_PIXCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_PIXCLK 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_PIXCLK 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_PIXCLK 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_PIXCLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_PIXCLK 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_PIXCLK 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_PIXCLK 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.278 SW_PAD_CTL_PAD_CSI_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC)

SW_PAD_CTL Register

Address: 20E_0000h base + 468h offset = 20E_0468h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_VSYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_VSYNC 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_VSYNC 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_VSYNC 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_VSYNC

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_VSYNC 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_VSYNC 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_VSYNC 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.279 SW_PAD_CTL_PAD_CSI_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC)

SW_PAD_CTL Register

Address: 20E_0000h base + 46Ch offset = 20E_046Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_HSYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_HSYNC 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_HSYNC 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_HSYNC 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_HSYNC

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_HSYNC 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_HSYNC 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_HSYNC 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.280 SW_PAD_CTL_PAD_CSI_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA00)

SW_PAD_CTL Register

Address: 20E_0000h base + 470h offset = 20E_0470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA00 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA00 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA00 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.281 SW_PAD_CTL_PAD_CSI_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA01)

SW_PAD_CTL Register

Address: 20E_0000h base + 474h offset = 20E_0474h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA01 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA01 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA01 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.282 SW_PAD_CTL_PAD_CSI_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA02)

SW_PAD_CTL Register

Address: 20E_0000h base + 478h offset = 20E_0478h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA02 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA02 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA02 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.283 SW_PAD_CTL_PAD_CSI_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA03)

SW_PAD_CTL Register

Address: 20E_0000h base + 47Ch offset = 20E_047Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA03 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA03 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA03 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.284 SW_PAD_CTL_PAD_CSI_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA04)

SW_PAD_CTL Register

Address: 20E_0000h base + 480h offset = 20E_0480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA04 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA04 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.285 SW_PAD_CTL_PAD_CSI_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA05)

SW_PAD_CTL Register

Address: 20E_0000h base + 484h offset = 20E_0484h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA05 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA05 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA05 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.286 SW_PAD_CTL_PAD_CSI_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA06)

SW_PAD_CTL Register

Address: 20E_0000h base + 488h offset = 20E_0488h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA06

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA06 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA06 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA06 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.287 SW_PAD_CTL_PAD_CSI_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_CSI_DATA07)

SW_PAD_CTL Register

Address: 20E_0000h base + 48Ch offset = 20E_048Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_DATA07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_DATA07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_DATA07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_DATA07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_DATA07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_CSI_DATA07 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: CSI_DATA07 00 **SPEED_0_low_50MHz_** — low(50MHz) 01 **SPEED_1_medium_100MHz_** — medium(100MHz) 10 **SPEED_2_medium_100MHz_** — medium(100MHz) 11 **SPEED_3_max_200MHz_** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: CSI_DATA07 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_DATA07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

30.5.288 SW_PAD_CTL_GRP_ADDDS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_ADDDS)

SW_GRP Register

Address: 20E_0000h base + 490h offset = 20E_0490h

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

IOMUXC_SW_PAD_CTL_GRP_ADDDS field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for group: ADDDS (Pads: DRAM_ADDR00 DRAM_ADDR01 DRAM_ADDR02 DRAM_ADDR03 DRAM_ADDR04 DRAM_ADDR05 DRAM_ADDR06 DRAM_ADDR07 DRAM_ADDR08 DRAM_ADDR09 DRAM_ADDR10 DRAM_ADDR11 DRAM_ADDR12 DRAM_ADDR13 DRAM_ADDR14 DRAM_ADDR15 DRAM_SDBA0 DRAM_SDBA1) 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.289 SW_PAD_CTL_GRP_DDRMODE_CTL SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRMODE_CTL)

SW_GRP Register

Address: 20E_0000h base + 494h offset = 20E_0494h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														DDR_INPUT	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_PAD_CTL_GRP_DDRMODE_CTL field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for group: DDRMODE_CTL (Pads: DRAM_SDQS0_P DRAM_SDQS1_P) 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
-	This field is reserved. Reserved

30.5.290 SW_PAD_CTL_GRP_B0DS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_B0DS)

SW_GRP Register

Address: 20E_0000h base + 498h offset = 20E_0498h

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

IOMUXC_SW_PAD_CTL_GRP_B0DS field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for group: B0DS (Pads: DRAM_DATA00 DRAM_DATA01 DRAM_DATA02 DRAM_DATA03 DRAM_DATA04 DRAM_DATA05 DRAM_DATA06 DRAM_DATA07) 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.291 SW_PAD_CTL_GRP_DDRPK SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRPK)

SW_GRP Register

Address: 20E_0000h base + 49Ch offset = 20E_049Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_PAD_CTL_GRP_DDRPK field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13 PUE	Pull / Keep Select Field Select one out of next values for group: DDRPK (Pads: DRAM_ADDR00 DRAM_ADDR01 DRAM_ADDR02 DRAM_ADDR03 DRAM_ADDR04 DRAM_ADDR05 DRAM_ADDR06 DRAM_ADDR07 DRAM_ADDR08 DRAM_ADDR09 DRAM_ADDR10 DRAM_ADDR11 DRAM_ADDR12 DRAM_ADDR13 DRAM_ADDR14 DRAM_ADDR15 DRAM_CAS_B DRAM_CS0_B DRAM_CS1_B DRAM_DATA00 DRAM_DATA01 DRAM_DATA02 DRAM_DATA03 DRAM_DATA04 DRAM_DATA05 DRAM_DATA06 DRAM_DATA07 DRAM_DATA08 DRAM_DATA09 DRAM_DATA10 DRAM_DATA11 DRAM_DATA12 DRAM_DATA13 DRAM_DATA14 DRAM_DATA15 DRAM_DQM0 DRAM_DQM1 DRAM_RAS_B DRAM_SDBA0 DRAM_SDBA1 DRAM_SDCLK0_P DRAM_SDWE_B) 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
- 	This field is reserved. Reserved

30.5.292 SW_PAD_CTL_GRP_CTLDS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_CTLDS)

SW_GRP Register

Address: 20E_0000h base + 4A0h offset = 20E_04A0h

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

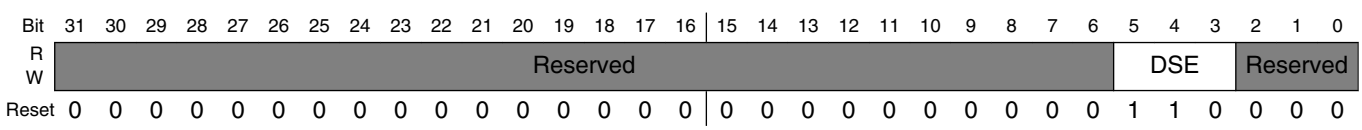
IOMUXC_SW_PAD_CTL_GRP_CTLDS field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for group: CTLDS (Pads: DRAM_CS0_B DRAM_CS1_B DRAM_SDBA2 DRAM_SDCKE0 DRAM_SDCKE1 DRAM_SDWE_B) 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.293 SW_PAD_CTL_GRP_B1DS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_B1DS)

SW_GRP Register

Address: 20E_0000h base + 4A4h offset = 20E_04A4h



IOMUXC_SW_PAD_CTL_GRP_B1DS field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–3 DSE	Drive Strength Field Select one out of next values for group: B1DS (Pads: DRAM_DATA08 DRAM_DATA09 DRAM_DATA10 DRAM_DATA11 DRAM_DATA12 DRAM_DATA13 DRAM_DATA14 DRAM_DATA15) 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_260_Ohm___3_3V__150_Ohm_1_8V__240_Ohm_for_DDR_** — R0(260 Ohm @ 3.3V, 150 Ohm@1.8V, 240 Ohm for DDR) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4

Table continues on the next page...

IOMUXC_SW_PAD_CTL_GRP_B1DS field descriptions (continued)

Field	Description
	101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
-	This field is reserved. Reserved

30.5.294 SW_PAD_CTL_GRP_DDRHYS SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRHYS)

SW_GRP Register

Address: 20E_0000h base + 4A8h offset = 20E_04A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SW_PAD_CTL_GRP_DDRHYS field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for group: DDRHYS (Pads: DRAM_DATA00 DRAM_DATA01 DRAM_DATA02 DRAM_DATA03 DRAM_DATA04 DRAM_DATA05 DRAM_DATA06 DRAM_DATA07 DRAM_DATA08 DRAM_DATA09 DRAM_DATA10 DRAM_DATA11 DRAM_DATA12 DRAM_DATA13 DRAM_DATA14 DRAM_DATA15 DRAM_SDQS0_P DRAM_SDQS1_P) 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
-	This field is reserved. Reserved

30.5.295 SW_PAD_CTL_GRP_DDRPKE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRPKE)

SW_GRP Register

Address: 20E_0000h base + 4ACh offset = 20E_04ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

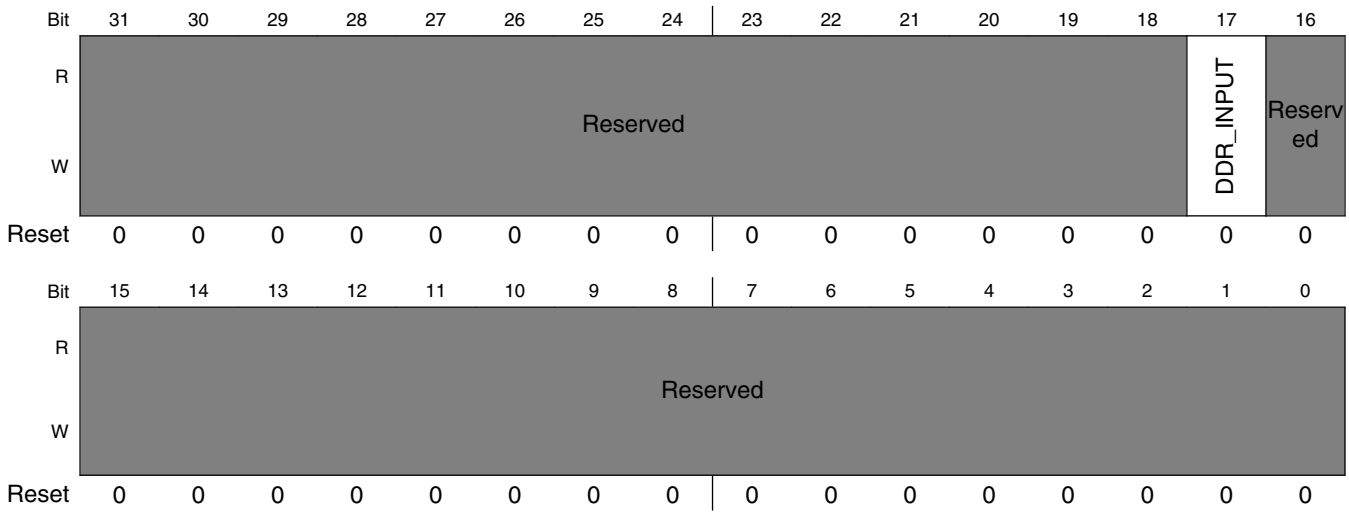
IOMUXC_SW_PAD_CTL_GRP_DDRPKE field descriptions

Field	Description
31–13 -	This field is reserved. Reserved
12 PKE	Pull / Keep Enable Field Select one out of next values for group: DDRPKE (Pads: DRAM_ADDR00 DRAM_ADDR01 DRAM_ADDR02 DRAM_ADDR03 DRAM_ADDR04 DRAM_ADDR05 DRAM_ADDR06 DRAM_ADDR07 DRAM_ADDR08 DRAM_ADDR09 DRAM_ADDR10 DRAM_ADDR11 DRAM_ADDR12 DRAM_ADDR13 DRAM_ADDR14 DRAM_ADDR15 DRAM_CAS_B DRAM_CS0_B DRAM_CS1_B DRAM_DATA00 DRAM_DATA01 DRAM_DATA02 DRAM_DATA03 DRAM_DATA04 DRAM_DATA05 DRAM_DATA06 DRAM_DATA07 DRAM_DATA08 DRAM_DATA09 DRAM_DATA10 DRAM_DATA11 DRAM_DATA12 DRAM_DATA13 DRAM_DATA14 DRAM_DATA15 DRAM_DQM0 DRAM_DQM1 DRAM_RAS_B DRAM_SDBA0 DRAM_SDBA1 DRAM_SDCLK0_P DRAM_SDWE_B) 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
-	This field is reserved. Reserved

30.5.296 SW_PAD_CTL_GRP_DDRMODE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDRMODE)

SW_GRP Register

Address: 20E_0000h base + 4B0h offset = 20E_04B0h



IOMUXC_SW_PAD_CTL_GRP_DDRMODE field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for group: DDRMODE (Pads: DRAM_DATA00 DRAM_DATA01 DRAM_DATA02 DRAM_DATA03 DRAM_DATA04 DRAM_DATA05 DRAM_DATA06 DRAM_DATA07 DRAM_DATA08 DRAM_DATA09 DRAM_DATA10 DRAM_DATA11 DRAM_DATA12 DRAM_DATA13 DRAM_DATA14 DRAM_DATA15) 0 **DDR_INPUT_0_CMOS_input_type** — CMOS input type 1 **DDR_INPUT_1_Differential_input_mode** — Differential input mode
-	This field is reserved. Reserved

30.5.297 SW_PAD_CTL_GRP_DDR_TYPE SW GRP Register (IOMUXC_SW_PAD_CTL_GRP_DDR_TYPE)

SW_GRP Register

Address: 20E_0000h base + 4B4h offset = 20E_04B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												DDR_SEL		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

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

IOMUXC_SW_PAD_CTL_GRP_DDR_TYPE field descriptions

Field	Description
31–20 -	This field is reserved. Reserved
19–18 DDR_SEL	ddr_sel Field Select one out of next values for group: DDR_TYPE (Pads: DRAM_ADDR00 DRAM_ADDR01 DRAM_ADDR02 DRAM_ADDR03 DRAM_ADDR04 DRAM_ADDR05 DRAM_ADDR06 DRAM_ADDR07 DRAM_ADDR08 DRAM_ADDR09 DRAM_ADDR10 DRAM_ADDR11 DRAM_ADDR12 DRAM_ADDR13 DRAM_ADDR14 DRAM_ADDR15 DRAM_CAS_B DRAM_CS0_B DRAM_CS1_B DRAM_DATA00 DRAM_DATA01 DRAM_DATA02 DRAM_DATA03 DRAM_DATA04 DRAM_DATA05 DRAM_DATA06 DRAM_DATA07 DRAM_DATA08 DRAM_DATA09 DRAM_DATA10 DRAM_DATA11 DRAM_DATA12 DRAM_DATA13 DRAM_DATA14 DRAM_DATA15 DRAM_DQM0 DRAM_DQM1 DRAM_ODT0 DRAM_ODT1 DRAM_RAS_B DRAM_SDBA0 DRAM_SDBA1 DRAM_SDBA2 DRAM_SDCKE0 DRAM_SDCKE1 DRAM_SDCLK0_P DRAM_SDQS0_P DRAM_SDQS1_P DRAM_SDWE_B) 00 **DDR_SEL_0_RESERVED** — RESERVED 01 **DDR_SEL_1_RESERVED** — RESERVED 10 **DDR_SEL_2_LPDDR2_mode** — LPDDR2 mode 11 **DDR_SEL_3_DDR3_mode** — DDR3 mode
-	This field is reserved. Reserved

30.5.298 USB_OTG1_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_ID_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4B8h offset = 20E_04B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USB_OTG1_ID_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: anatop, In Pin: usb_otg_id 00 **GPIO1_IO00_ALT2** — Selecting Pad: GPIO1_IO00 for Mode: ALT2 01 **UART3_TX_DATA_ALT8** — Selecting Pad: UART3_TX_DATA for Mode: ALT8 10 **SD1_DATA0_ALT8** — Selecting Pad: SD1_DATA0 for Mode: ALT8

30.5.299 USB_OTG2_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_ID_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4BCh offset = 20E_04BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USB_OTG2_ID_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: anatop, In Pin: usb_uh1_id 00 **GPIO1_IO05_ALT2** — Selecting Pad: GPIO1_IO05 for Mode: ALT2 01 **ENET2_TX_CLK_ALT8** — Selecting Pad: ENET2_TX_CLK for Mode: ALT8 10 **SD1_DATA3_ALT8** — Selecting Pad: SD1_DATA3 for Mode: ALT8

30.5.300 CCM_PMIC_READY_SELECT_INPUT DAISY Register (IOMUXC_CCM_PMIC_READY_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4C0h offset = 20E_04C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

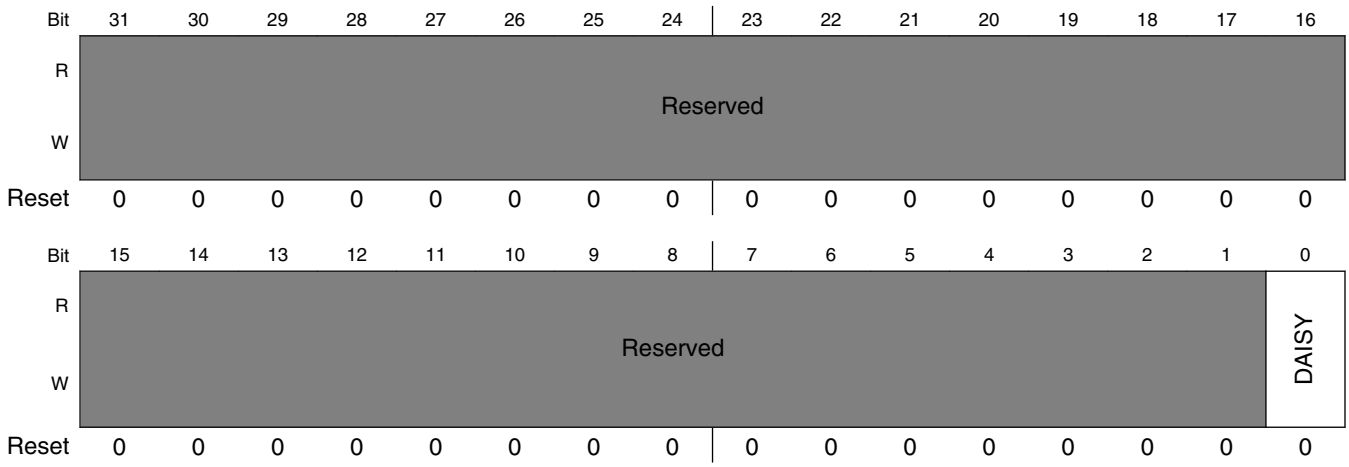
IOMUXC_CCM_PMIC_READY_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ccm, In Pin: pmic_ready 0 **JTAG_MOD_ALT4** — Selecting Pad: JTAG_MOD for Mode: ALT4 1 **GPIO1_IO08_ALT6** — Selecting Pad: GPIO1_IO08 for Mode: ALT6

30.5.301 CSI_DATA02_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA02_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4C4h offset = 20E_04C4h



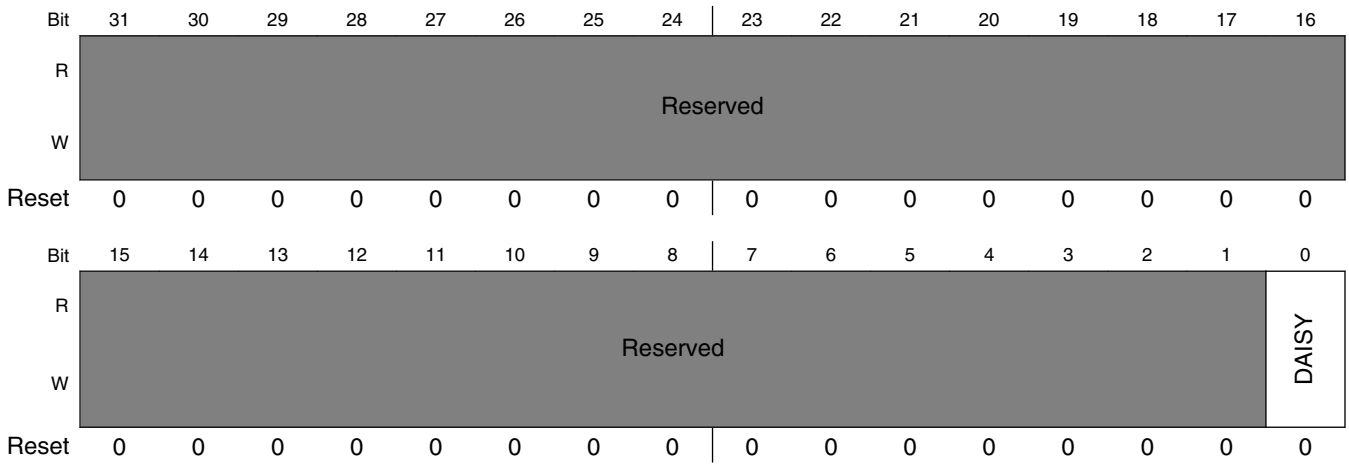
IOMUXC_CSI_DATA02_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d2 0 **CSI_DATA00_ALT0** — Selecting Pad: CSI_DATA00 for Mode: ALT0 1 **UART1_TX_DATA_ALT3** — Selecting Pad: UART1_TX_DATA for Mode: ALT3

30.5.302 CSI_DATA03_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA03_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4C8h offset = 20E_04C8h



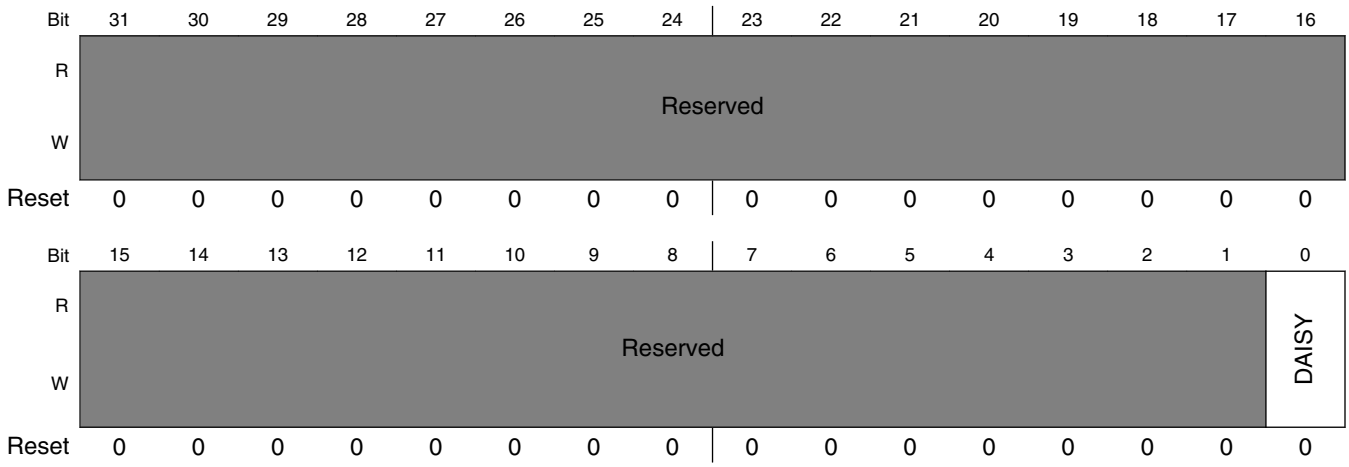
IOMUXC_CSI_DATA03_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d3 0 **CSI_DATA01_ALT0** — Selecting Pad: CSI_DATA01 for Mode: ALT0 1 **UART1_RX_DATA_ALT3** — Selecting Pad: UART1_RX_DATA for Mode: ALT3

30.5.303 CSI_DATA05_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA05_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4CCh offset = 20E_04CCh



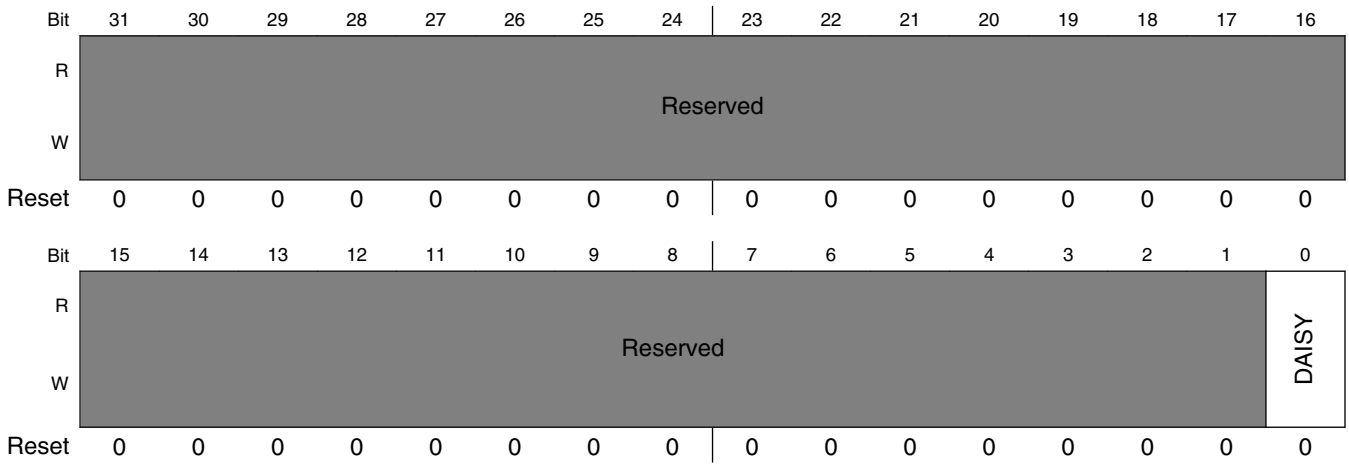
IOMUXC_CSI_DATA05_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d5 0 **CSI_DATA03_ALT0** — Selecting Pad: CSI_DATA03 for Mode: ALT0 1 **UART1_RTS_B_ALT3** — Selecting Pad: UART1_RTS_B for Mode: ALT3

30.5.304 CSI_DATA00_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA00_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4D0h offset = 20E_04D0h



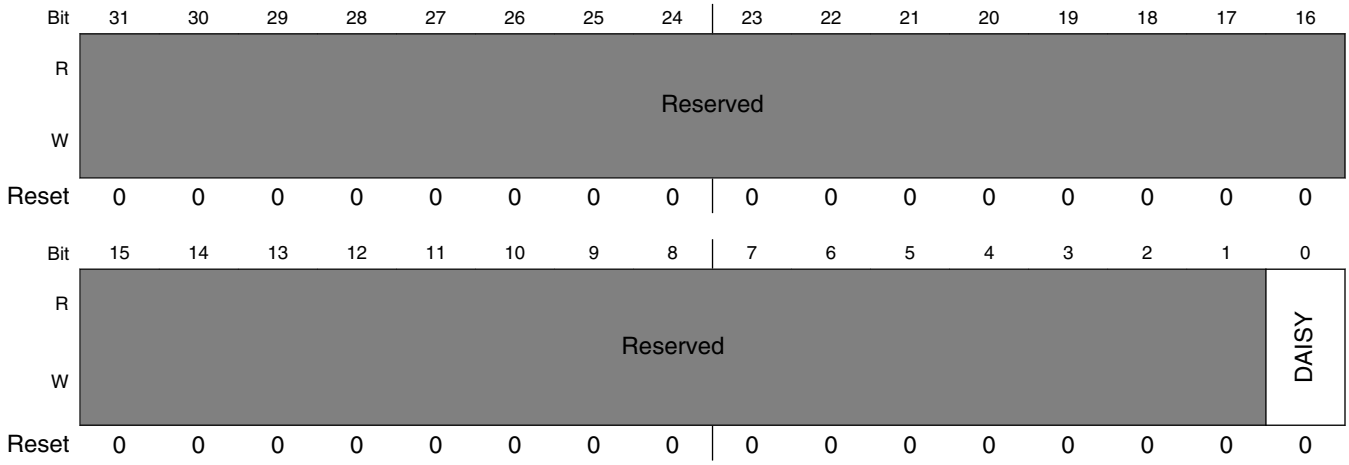
IOMUXC_CSI_DATA00_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d0 0 **UART3_RX_DATA_ALT3** — Selecting Pad: UART3_RX_DATA for Mode: ALT3 1 **LCD_DATA17_ALT3** — Selecting Pad: LCD_DATA17 for Mode: ALT3

30.5.305 CSI_DATA01_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA01_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4D4h offset = 20E_04D4h



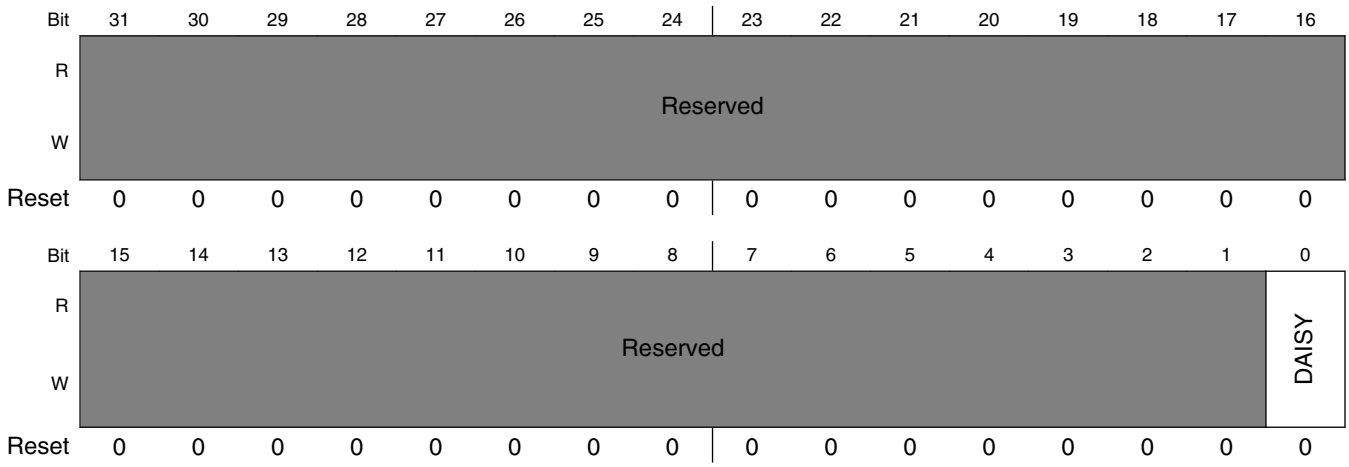
IOMUXC_CSI_DATA01_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d1 0 **UART3_TX_DATA_ALT3** — Selecting Pad: UART3_TX_DATA for Mode: ALT3 1 **LCD_DATA16_ALT3** — Selecting Pad: LCD_DATA16 for Mode: ALT3

30.5.306 CSI_DATA04_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA04_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4D8h offset = 20E_04D8h



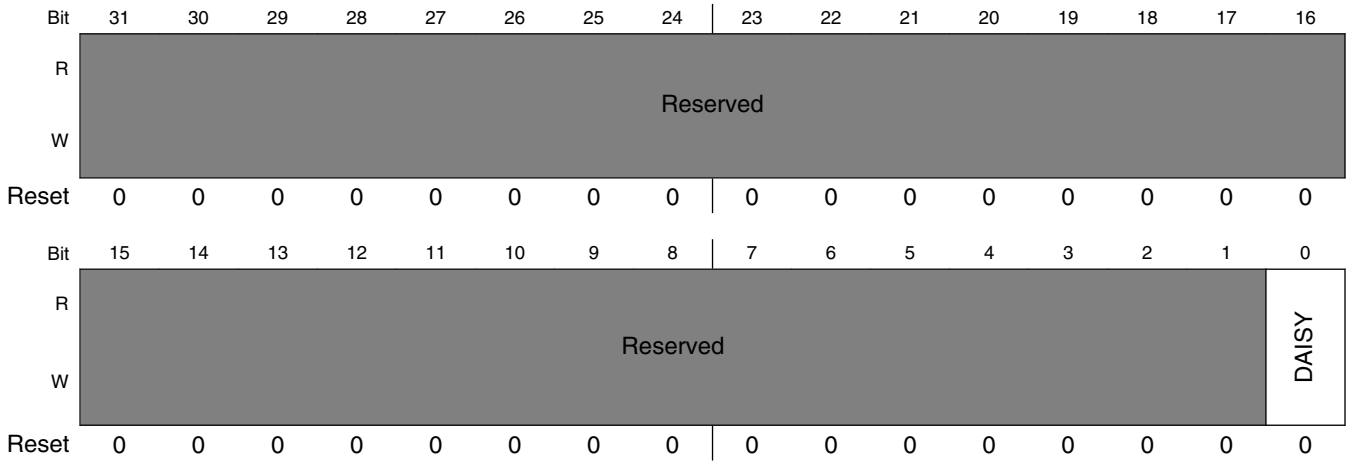
IOMUXC_CSI_DATA04_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d4 0 **UART1_CTS_B_ALT3** — Selecting Pad: UART1_CTS_B for Mode: ALT3 1 **CSI_DATA02_ALT0** — Selecting Pad: CSI_DATA02 for Mode: ALT0

30.5.307 CSI_DATA06_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA06_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4DCh offset = 20E_04DCh



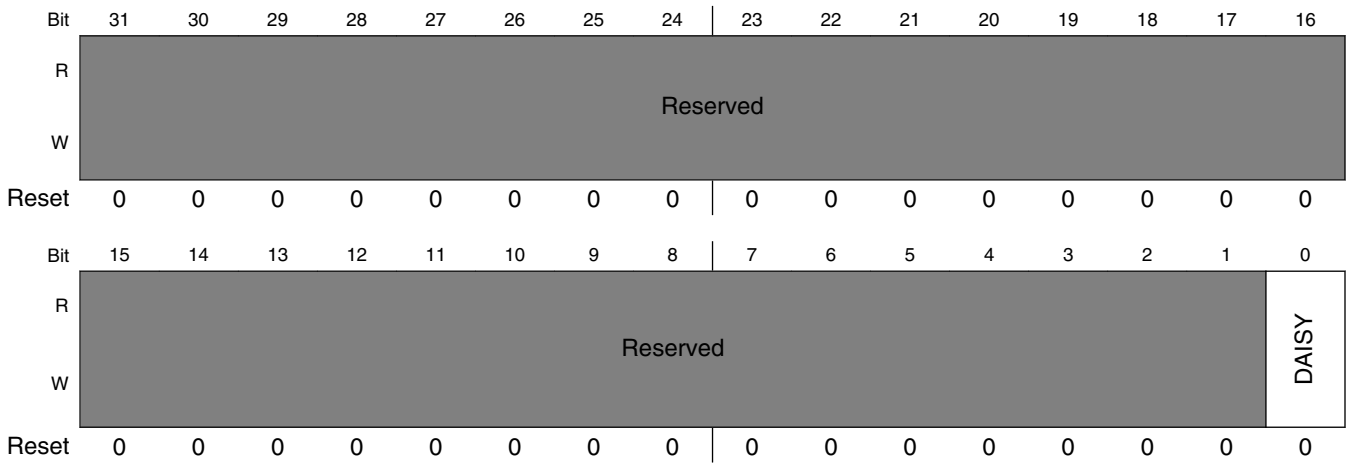
IOMUXC_CSI_DATA06_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d6 0 **UART2_TX_DATA_ALT3** — Selecting Pad: UART2_TX_DATA for Mode: ALT3 1 **CSI_DATA04_ALT0** — Selecting Pad: CSI_DATA04 for Mode: ALT0

30.5.308 CSI_DATA07_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA07_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4E0h offset = 20E_04E0h



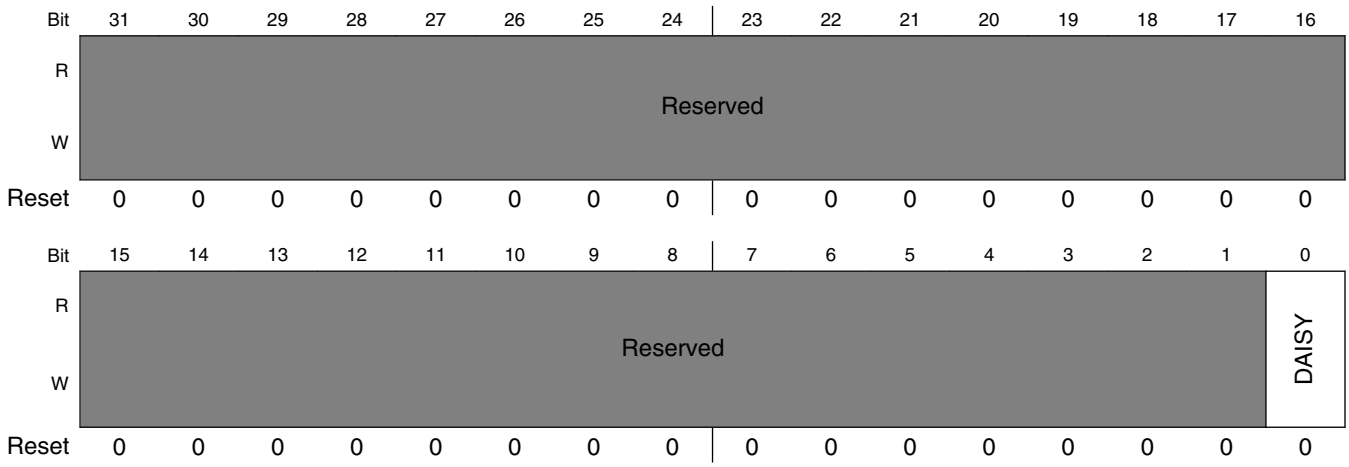
IOMUXC_CSI_DATA07_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d7 0 **UART2_RX_DATA_ALT3** — Selecting Pad: UART2_RX_DATA for Mode: ALT3 1 **CSI_DATA05_ALT0** — Selecting Pad: CSI_DATA05 for Mode: ALT0

30.5.309 CSI_DATA08_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA08_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4E4h offset = 20E_04E4h



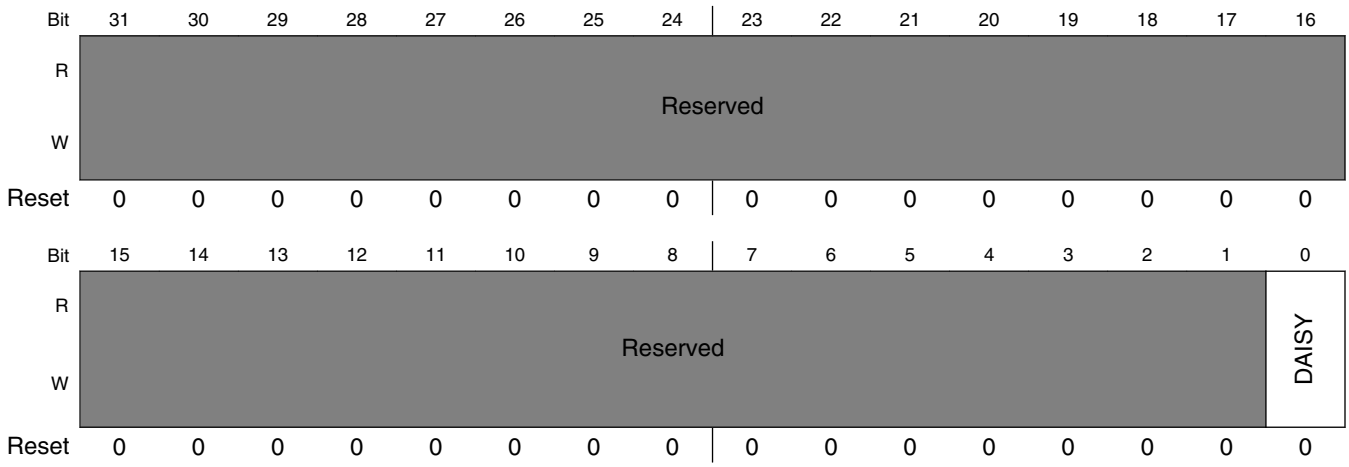
IOMUXC_CSI_DATA08_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d8 0 **UART2_CTS_B_ALT3** — Selecting Pad: UART2_CTS_B for Mode: ALT3 1 **CSI_DATA06_ALT0** — Selecting Pad: CSI_DATA06 for Mode: ALT0

30.5.310 CSI_DATA09_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA09_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4E8h offset = 20E_04E8h



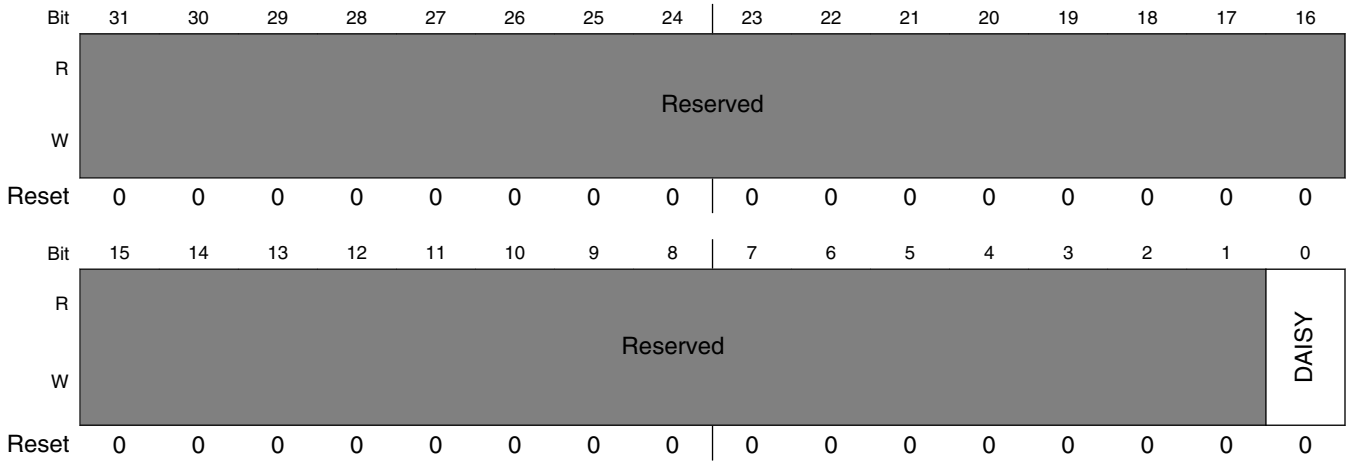
IOMUXC_CSI_DATA09_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d9 0 **UART2_RTS_B_ALT3** — Selecting Pad: UART2_RTS_B for Mode: ALT3 1 **CSI_DATA07_ALT0** — Selecting Pad: CSI_DATA07 for Mode: ALT0

30.5.311 CSI_DATA10_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA10_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4ECh offset = 20E_04ECh



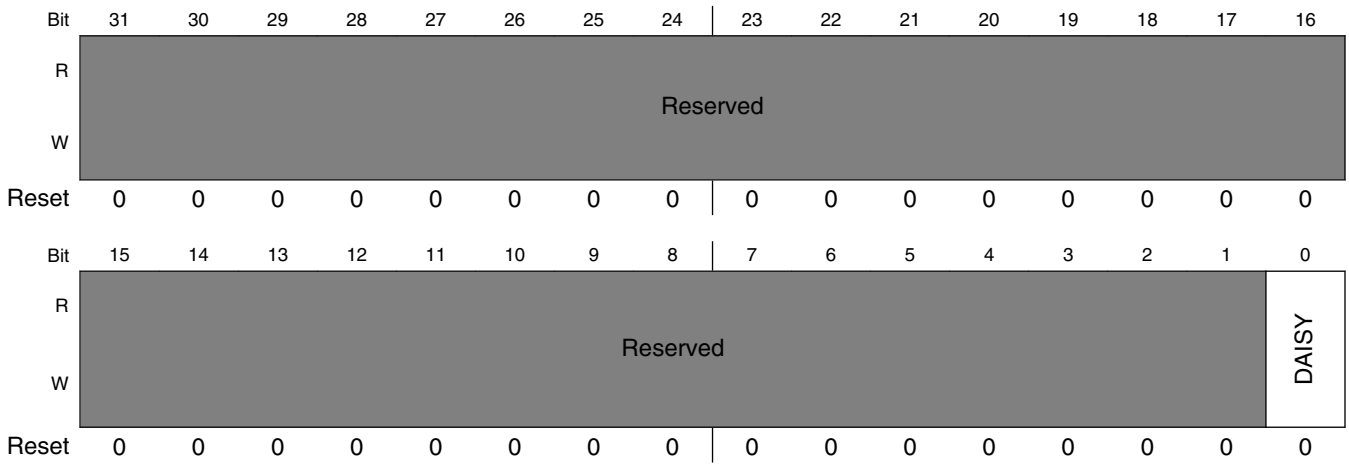
IOMUXC_CSI_DATA10_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d10 0 **UART3_CTS_B_ALT3** — Selecting Pad: UART3_CTS_B for Mode: ALT3 1 **LCD_DATA18_ALT3** — Selecting Pad: LCD_DATA18 for Mode: ALT3

30.5.312 CSI_DATA11_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA11_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4F0h offset = 20E_04F0h



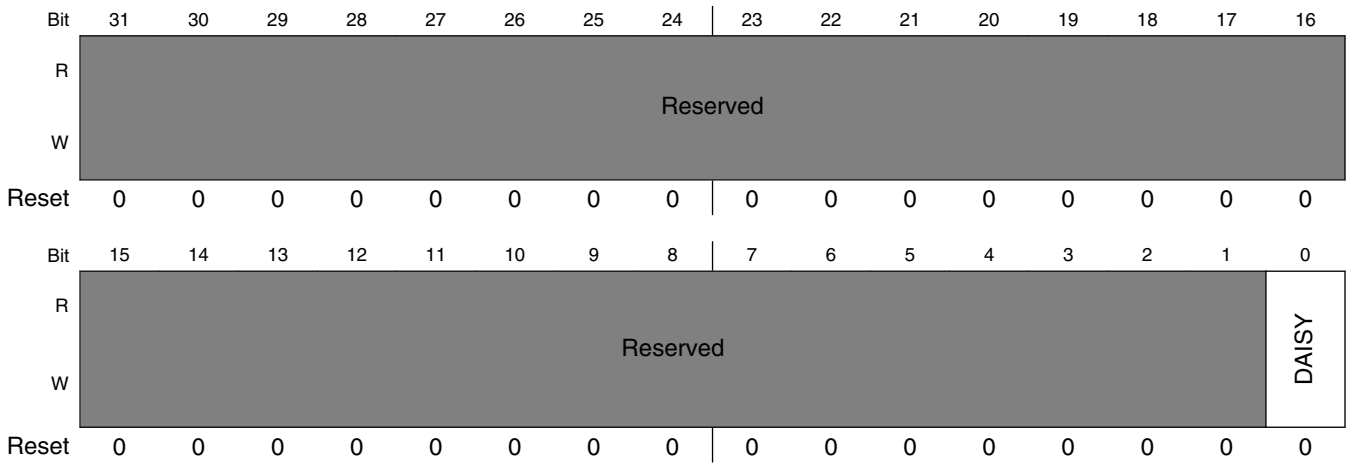
IOMUXC_CSI_DATA11_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d11 0 **UART3_RTS_B_ALT3** — Selecting Pad: UART3_RTS_B for Mode: ALT3 1 **LCD_DATA19_ALT3** — Selecting Pad: LCD_DATA19 for Mode: ALT3

30.5.313 CSI_DATA12_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA12_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4F4h offset = 20E_04F4h



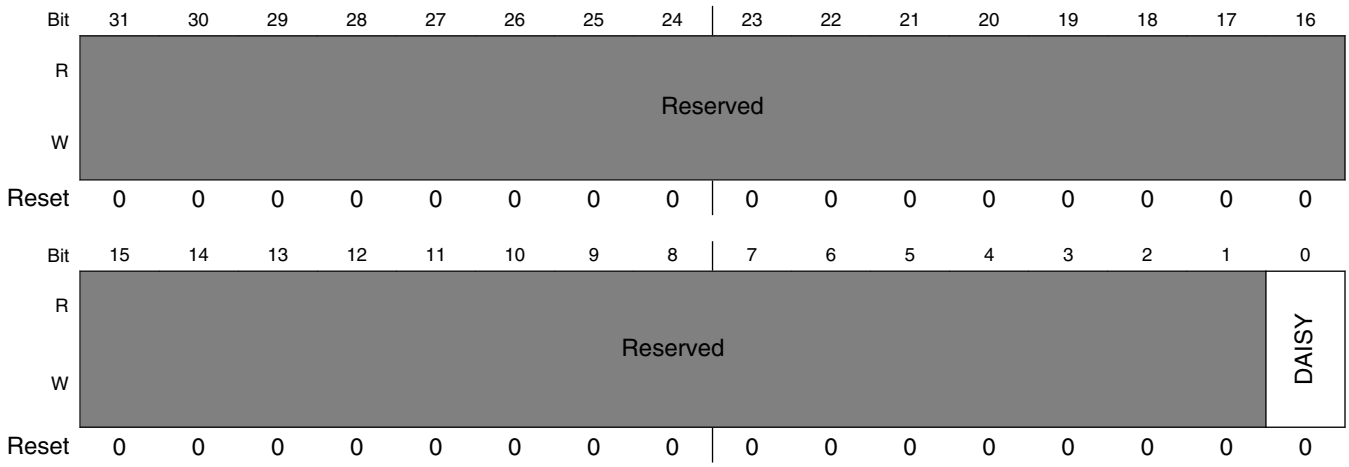
IOMUXC_CSI_DATA12_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d12 0 **UART4_TX_DATA_ALT3** — Selecting Pad: UART4_TX_DATA for Mode: ALT3 1 **LCD_DATA20_ALT3** — Selecting Pad: LCD_DATA20 for Mode: ALT3

30.5.314 CSI_DATA13_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA13_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4F8h offset = 20E_04F8h



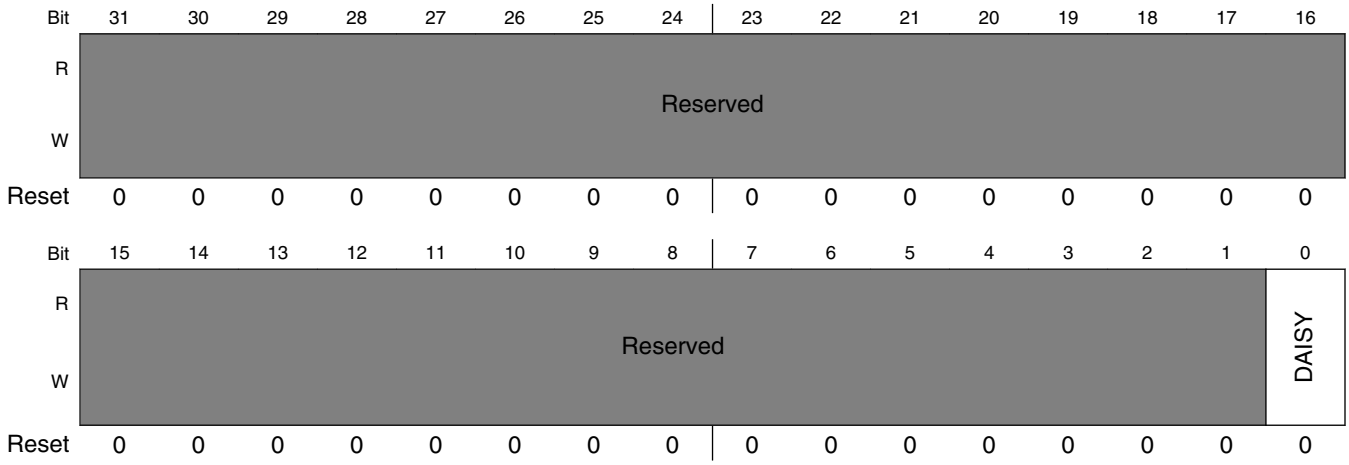
IOMUXC_CSI_DATA13_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d13 0 **UART4_RX_DATA_ALT3** — Selecting Pad: UART4_RX_DATA for Mode: ALT3 1 **LCD_DATA21_ALT3** — Selecting Pad: LCD_DATA21 for Mode: ALT3

30.5.315 CSI_DATA14_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA14_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 4FCh offset = 20E_04FCh



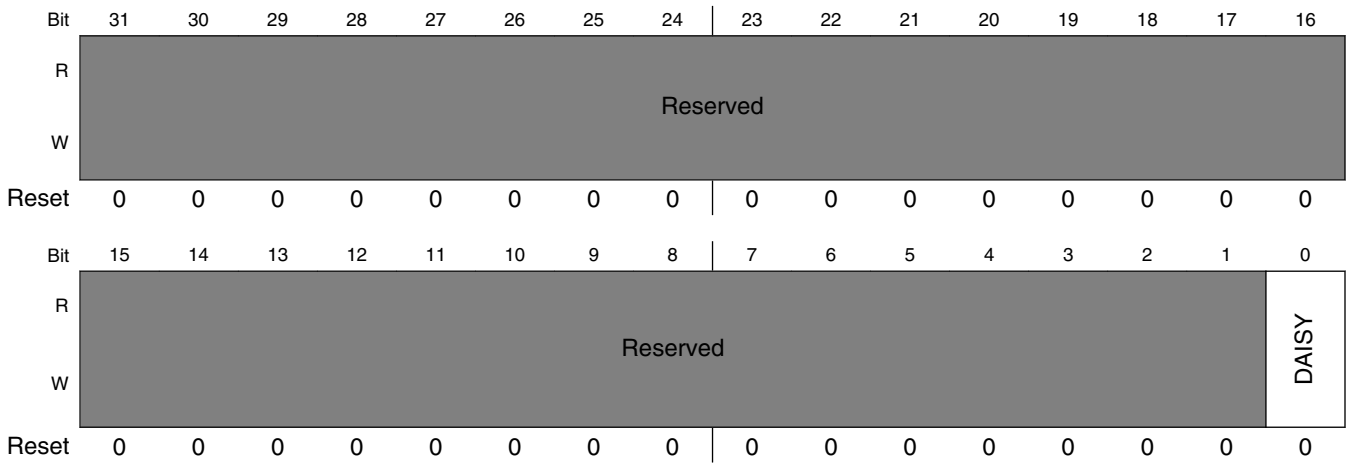
IOMUXC_CSI_DATA14_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d14 0 **UART5_TX_DATA_ALT3** — Selecting Pad: UART5_TX_DATA for Mode: ALT3 1 **LCD_DATA22_ALT3** — Selecting Pad: LCD_DATA22 for Mode: ALT3

30.5.316 CSI_DATA15_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA15_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 500h offset = 20E_0500h



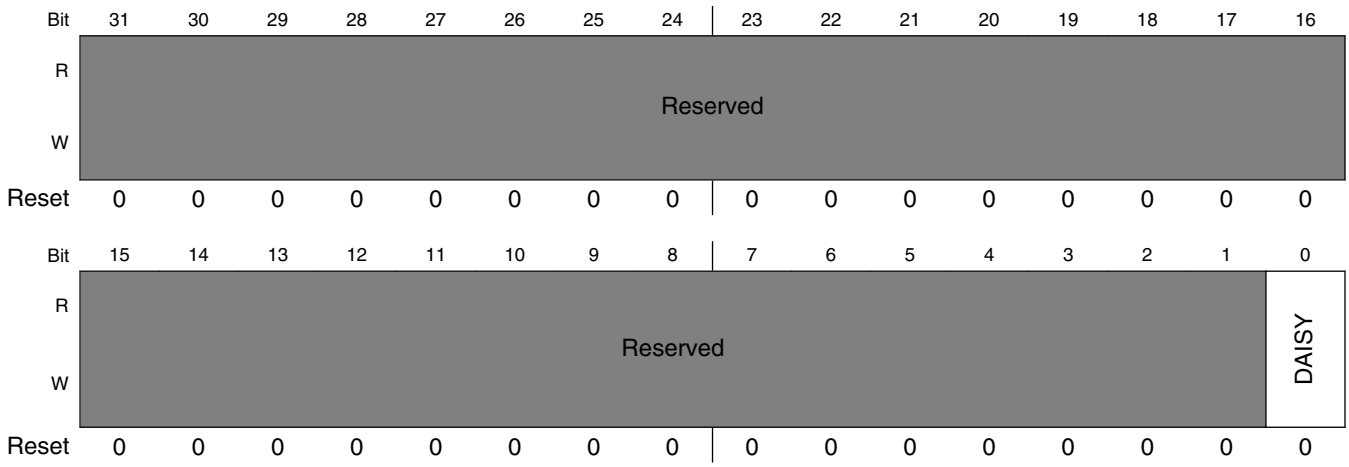
IOMUXC_CSI_DATA15_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d15 0 **UART5_RX_DATA_ALT3** — Selecting Pad: UART5_RX_DATA for Mode: ALT3 1 **LCD_DATA23_ALT3** — Selecting Pad: LCD_DATA23 for Mode: ALT3

30.5.317 CSI_DATA16_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA16_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 504h offset = 20E_0504h



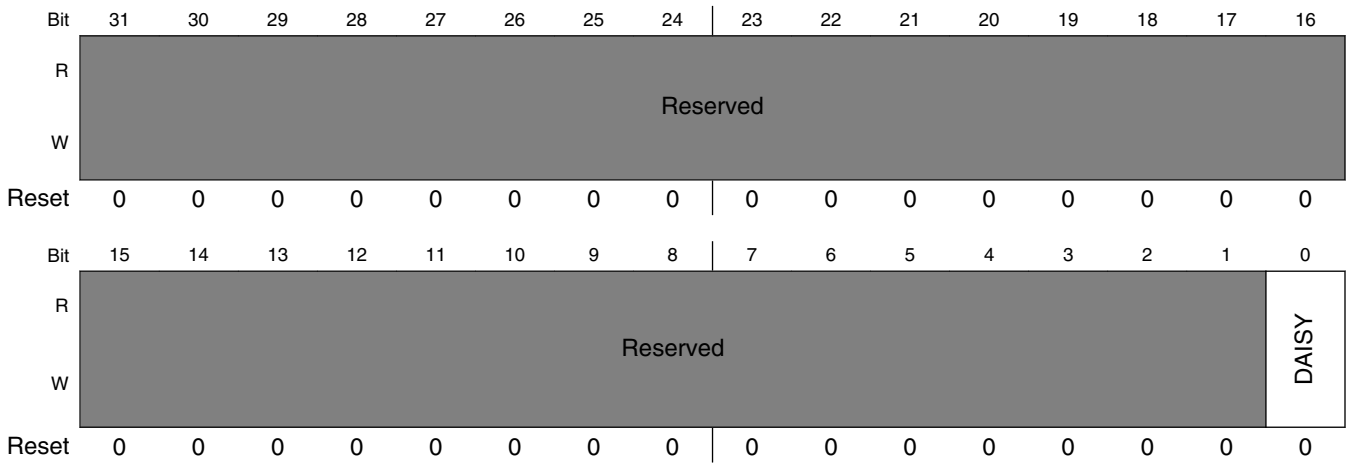
IOMUXC_CSI_DATA16_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d16 0 **ENET1_RX_DATA0_ALT3** — Selecting Pad: ENET1_RX_DATA0 for Mode: ALT3 1 **LCD_DATA08_ALT3** — Selecting Pad: LCD_DATA08 for Mode: ALT3

30.5.318 CSI_DATA17_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA17_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 508h offset = 20E_0508h



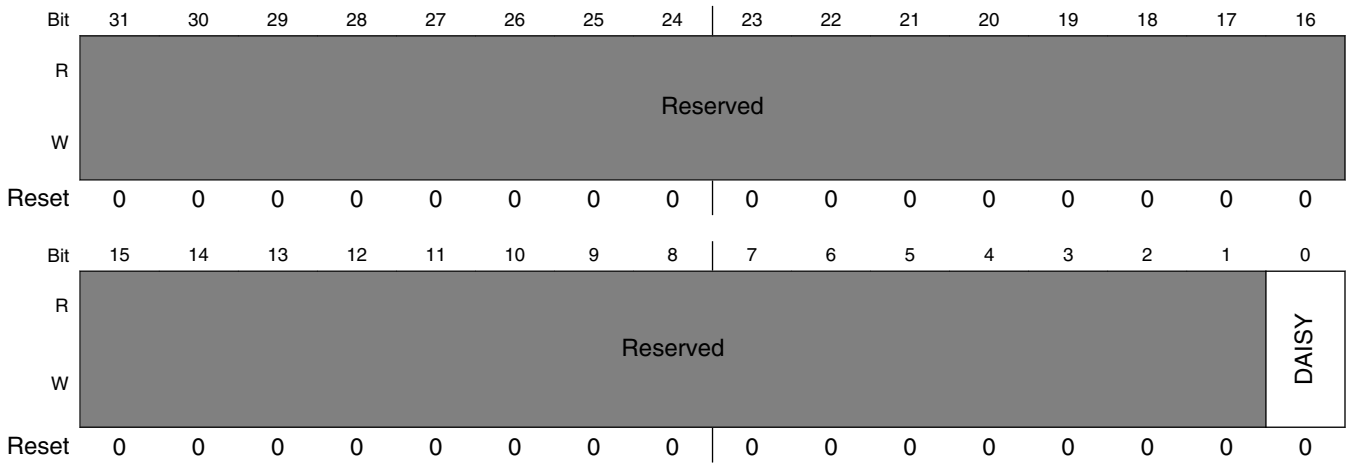
IOMUXC_CSI_DATA17_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d17 0 **ENET1_RX_DATA1_ALT3** — Selecting Pad: ENET1_RX_DATA1 for Mode: ALT3 1 **LCD_DATA09_ALT3** — Selecting Pad: LCD_DATA09 for Mode: ALT3

30.5.319 CSI_DATA18_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA18_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 50Ch offset = 20E_050Ch



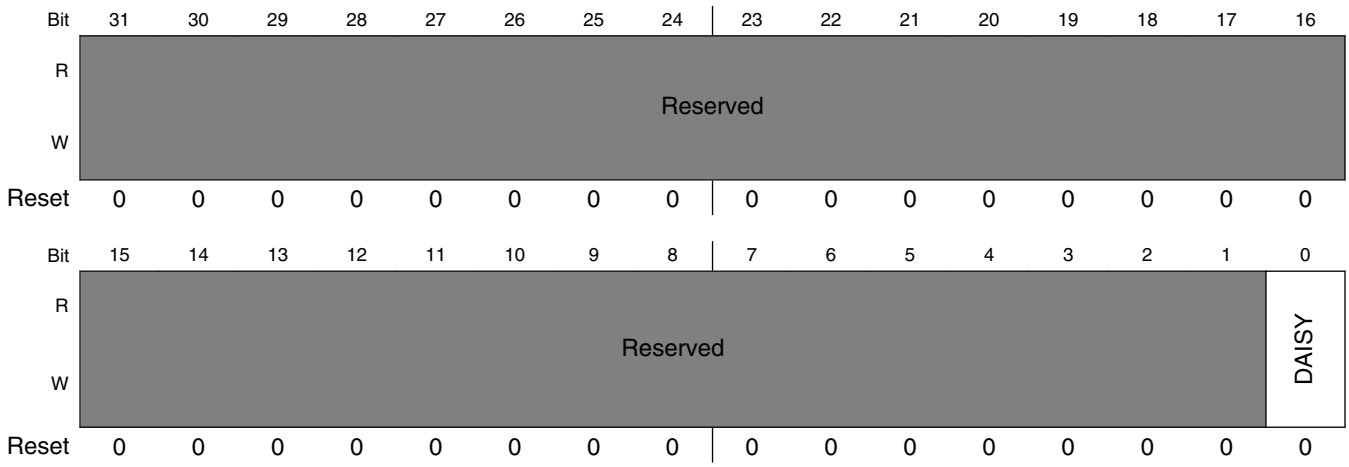
IOMUXC_CSI_DATA18_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d18 0 **ENET1_RX_EN_ALT3** — Selecting Pad: ENET1_RX_EN for Mode: ALT3 1 **LCD_DATA10_ALT3** — Selecting Pad: LCD_DATA10 for Mode: ALT3

30.5.320 CSI_DATA19_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA19_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 510h offset = 20E_0510h



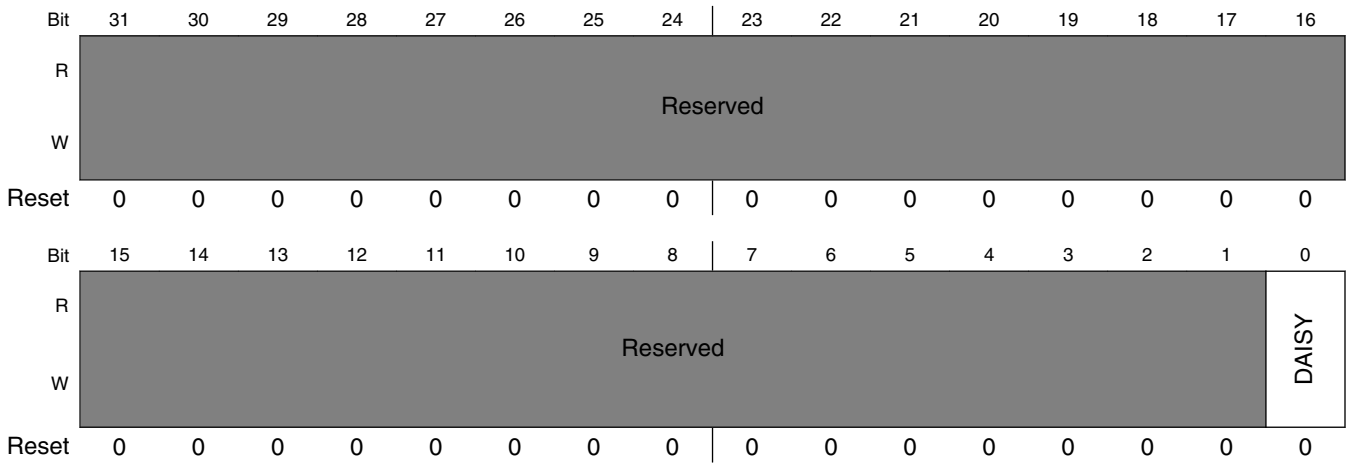
IOMUXC_CSI_DATA19_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d19 0 **ENET1_TX_DATA0_ALT3** — Selecting Pad: ENET1_TX_DATA0 for Mode: ALT3 1 **LCD_DATA11_ALT3** — Selecting Pad: LCD_DATA11 for Mode: ALT3

30.5.321 CSI_DATA20_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA20_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 514h offset = 20E_0514h



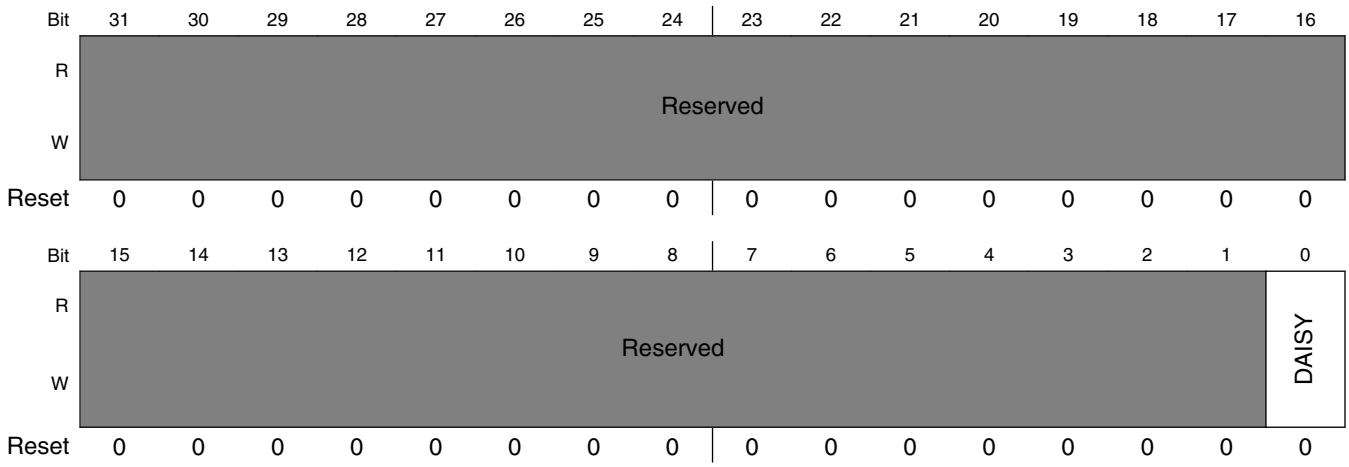
IOMUXC_CSI_DATA20_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d20 0 **ENET1_TX_DATA1_ALT3** — Selecting Pad: ENET1_TX_DATA1 for Mode: ALT3 1 **LCD_DATA12_ALT3** — Selecting Pad: LCD_DATA12 for Mode: ALT3

30.5.322 CSI_DATA21_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA21_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 518h offset = 20E_0518h



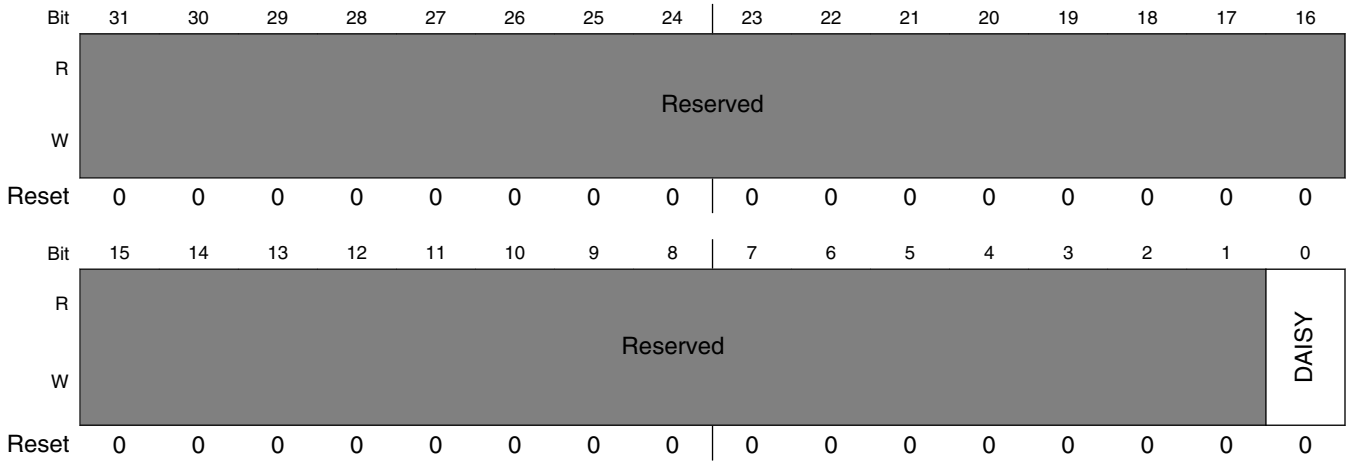
IOMUXC_CSI_DATA21_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d21 0 **ENET1_TX_EN_ALT3** — Selecting Pad: ENET1_TX_EN for Mode: ALT3 1 **LCD_DATA13_ALT3** — Selecting Pad: LCD_DATA13 for Mode: ALT3

30.5.323 CSI_DATA22_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA22_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 51Ch offset = 20E_051Ch



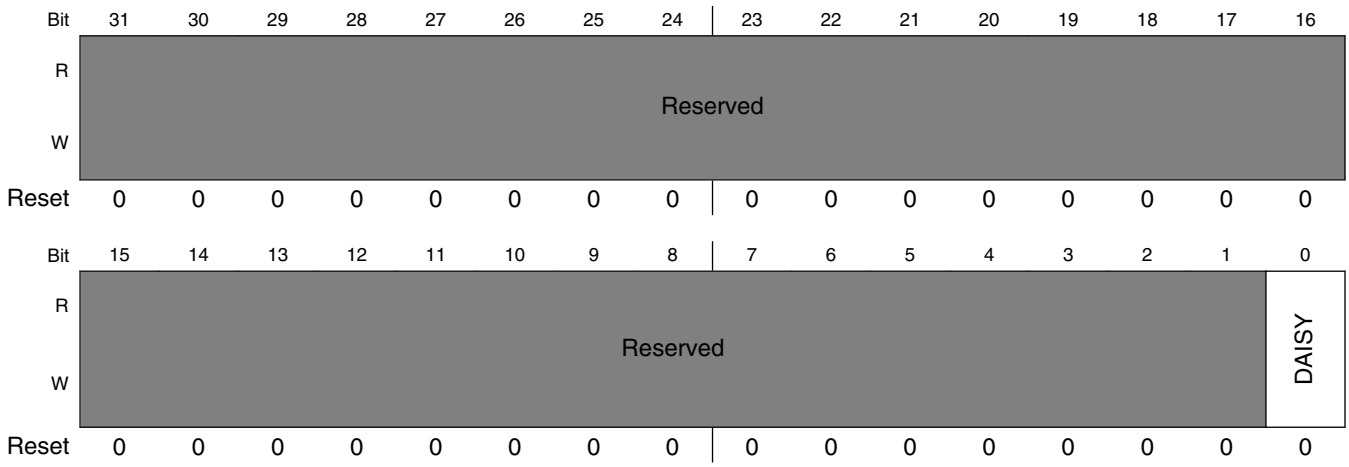
IOMUXC_CSI_DATA22_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d22 0 **ENET1_TX_CLK_ALT3** — Selecting Pad: ENET1_TX_CLK for Mode: ALT3 1 **LCD_DATA14_ALT3** — Selecting Pad: LCD_DATA14 for Mode: ALT3

30.5.324 CSI_DATA23_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA23_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 520h offset = 20E_0520h



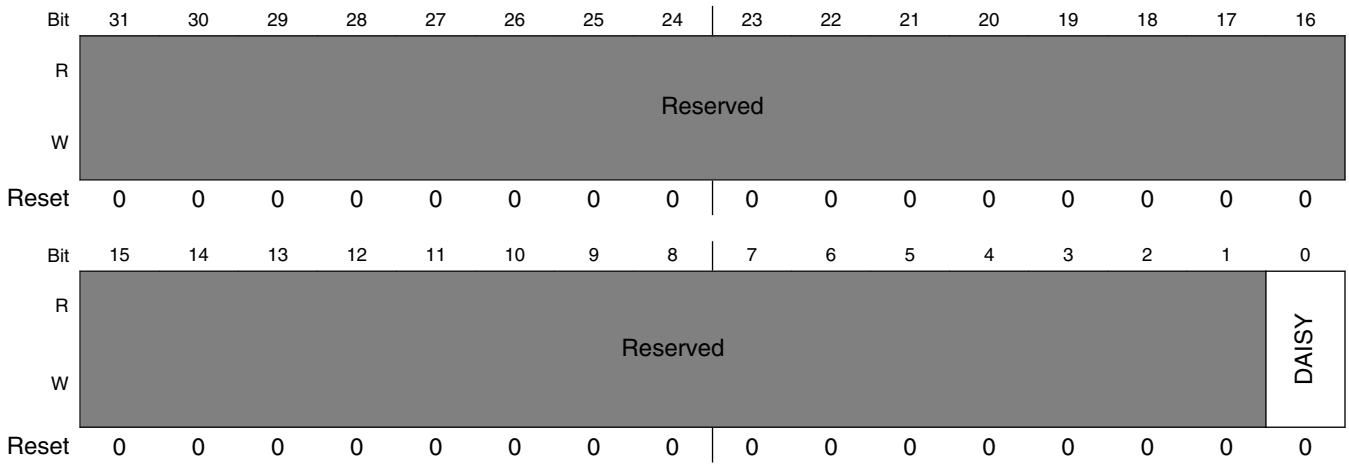
IOMUXC_CSI_DATA23_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d23 0 **ENET1_RX_ER_ALT3** — Selecting Pad: ENET1_RX_ER for Mode: ALT3 1 **LCD_DATA15_ALT3** — Selecting Pad: LCD_DATA15 for Mode: ALT3

30.5.325 CSI_HSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_HSYNC_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 524h offset = 20E_0524h



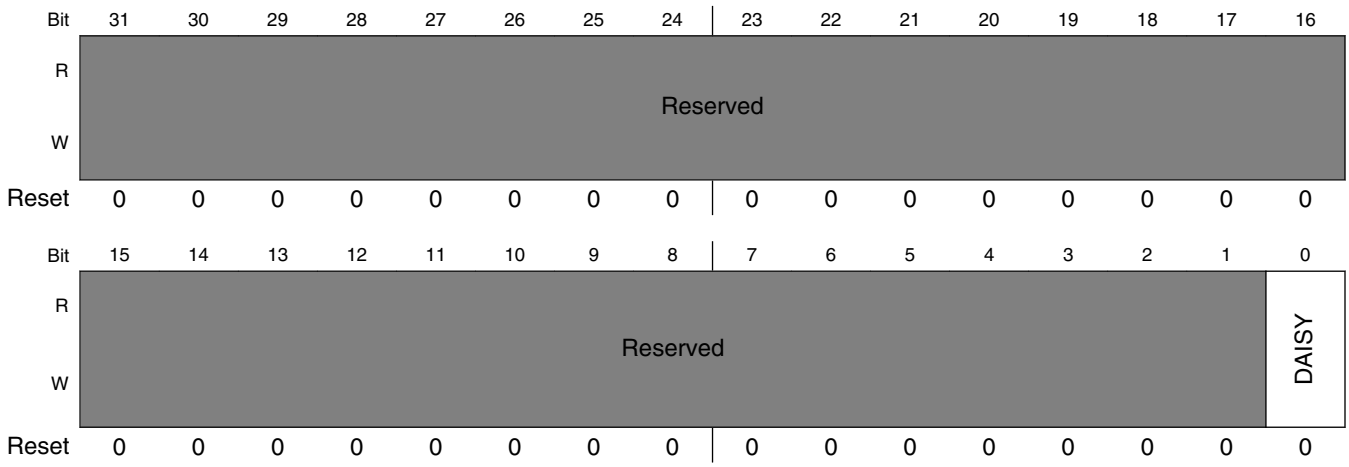
IOMUXC_CSI_HSYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_hsync 0 **CSI_HSYNC_ALT0** — Selecting Pad: CSI_HSYNC for Mode: ALT0 1 **GPIO1_IO09_ALT3** — Selecting Pad: GPIO1_IO09 for Mode: ALT3

30.5.326 CSI_PIXCLK_SELECT_INPUT DAISY Register (IOMUXC_CSI_PIXCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 528h offset = 20E_0528h



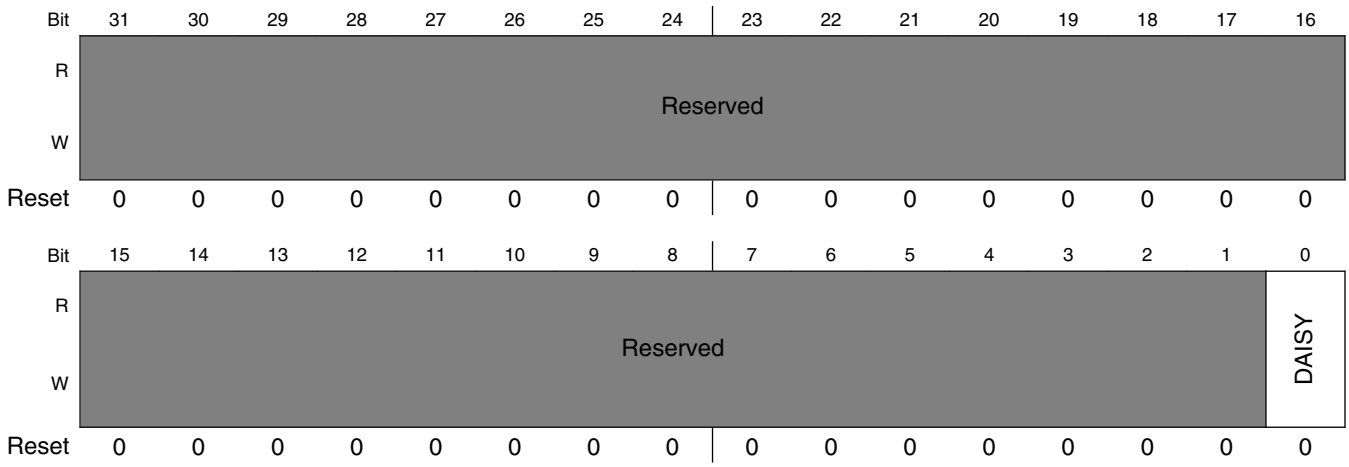
IOMUXC_CSI_PIXCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_pixclk 0 **GPIO1_IO07_ALT3** — Selecting Pad: GPIO1_IO07 for Mode: ALT3 1 **CSI_PIXCLK_ALT0** — Selecting Pad: CSI_PIXCLK for Mode: ALT0

30.5.327 CSI_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_VSYNC_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 52Ch offset = 20E_052Ch



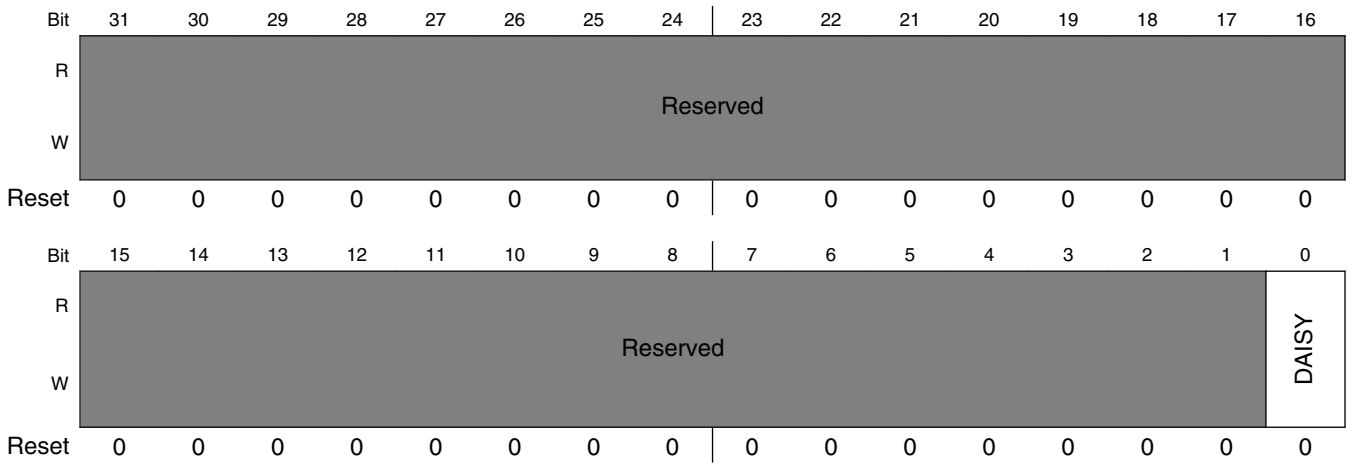
IOMUXC_CSI_VSYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_vsync 0 **CSI_VSYNC_ALT0** — Selecting Pad: CSI_VSYNC for Mode: ALT0 1 **GPIO1_IO08_ALT3** — Selecting Pad: GPIO1_IO08 for Mode: ALT3

30.5.328 CSI_FIELD_SELECT_INPUT DAISY Register (IOMUXC_CSI_FIELD_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 530h offset = 20E_0530h



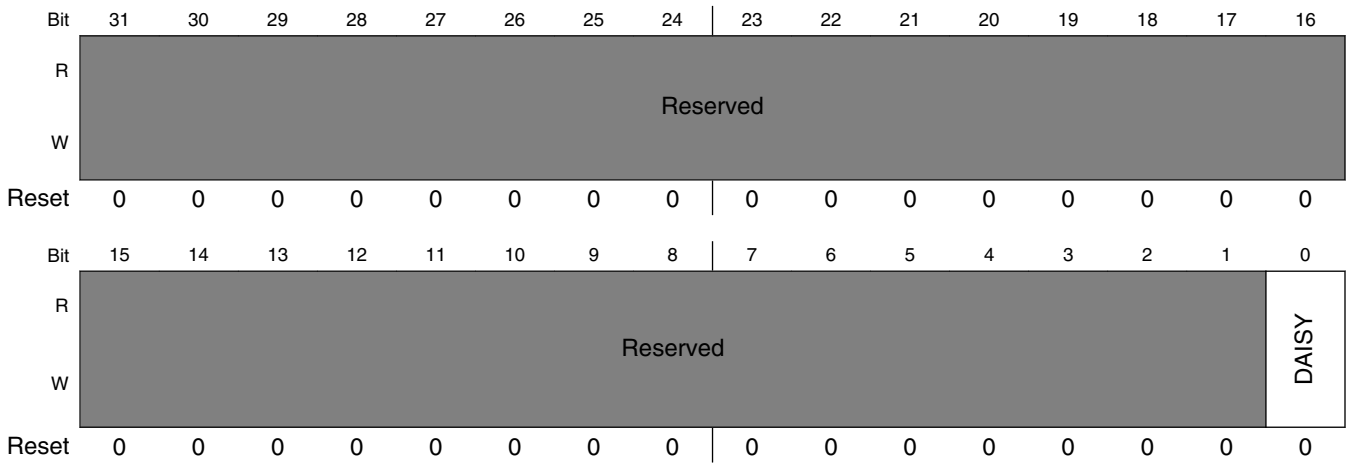
IOMUXC_CSI_FIELD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: tvdecoder_in_field 0 **GPIO1_IO05_ALT3** — Selecting Pad: GPIO1_IO05 for Mode: ALT3 1 **NAND_DQS_ALT1** — Selecting Pad: NAND_DQS for Mode: ALT1

30.5.329 ECSPI1_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 534h offset = 20E_0534h



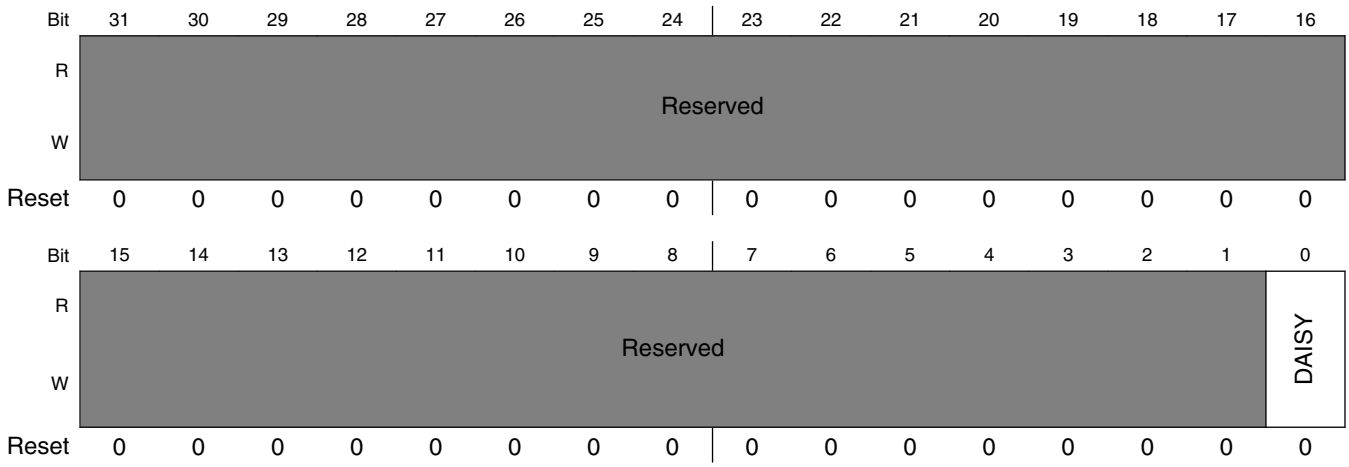
IOMUXC_ECSPI1_SCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi1, In Pin: cspi_clk_in 0 **LCD_DATA20_ALT2** — Selecting Pad: LCD_DATA20 for Mode: ALT2 1 **CSI_DATA04_ALT3** — Selecting Pad: CSI_DATA04 for Mode: ALT3

30.5.330 ECSPI1_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MISO_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 538h offset = 20E_0538h



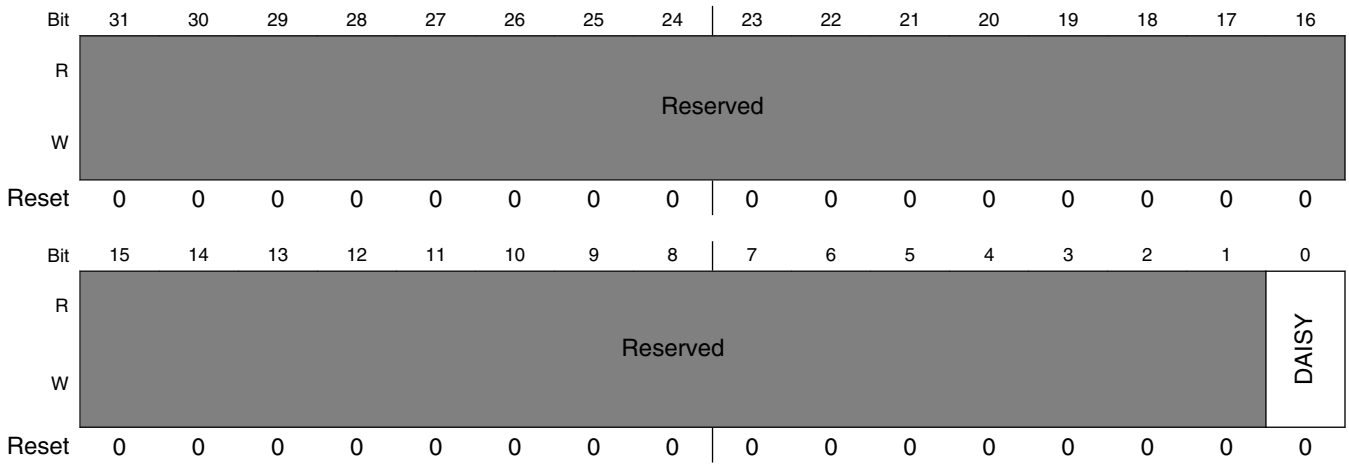
IOMUXC_ECSPI1_MISO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi1, In Pin: miso 0 **LCD_DATA23_ALT2** — Selecting Pad: LCD_DATA23 for Mode: ALT2 1 **CSI_DATA07_ALT3** — Selecting Pad: CSI_DATA07 for Mode: ALT3

30.5.331 ECSPI1_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MOSI_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 53Ch offset = 20E_053Ch



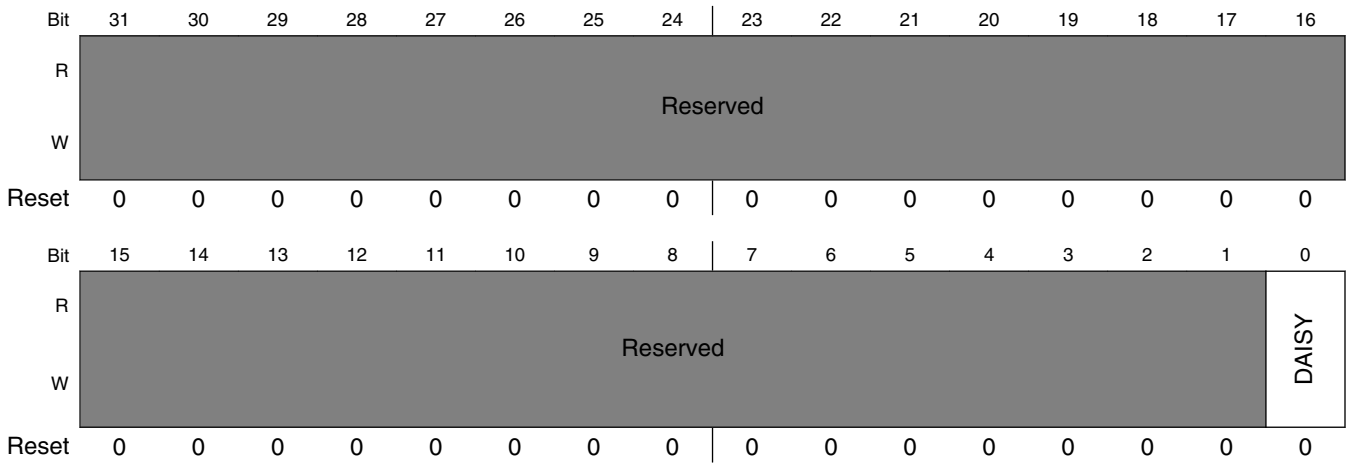
IOMUXC_ECSPI1_MOSI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi1, In Pin: mosi 0 **LCD_DATA22_ALT2** — Selecting Pad: LCD_DATA22 for Mode: ALT2 1 **CSI_DATA06_ALT3** — Selecting Pad: CSI_DATA06 for Mode: ALT3

30.5.332 ECSPI1_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SS0_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 540h offset = 20E_0540h



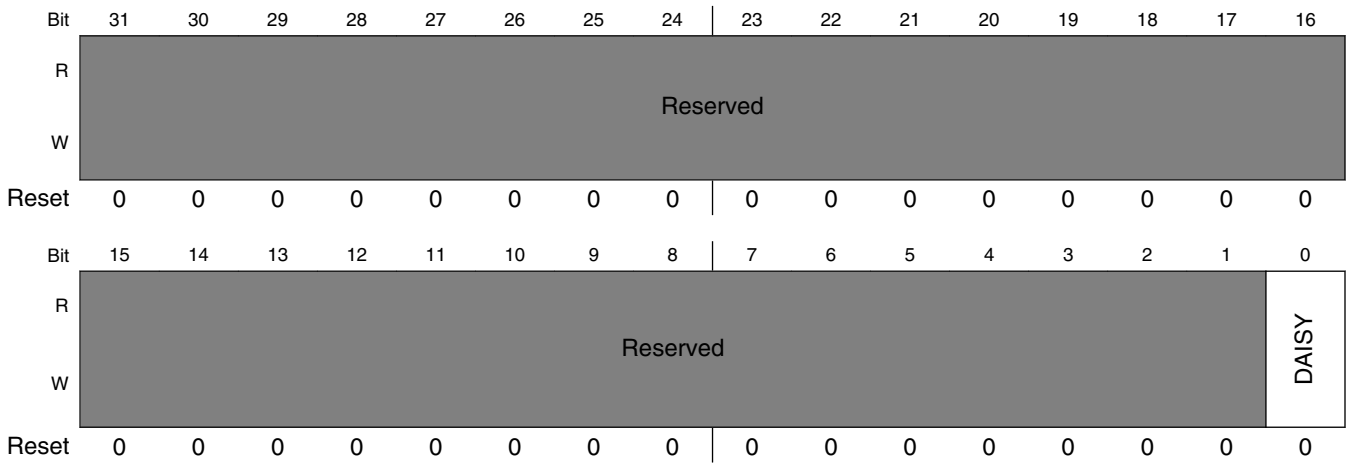
IOMUXC_ECSPI1_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi1, In Pin: ss_b0 0 **LCD_DATA21_ALT2** — Selecting Pad: LCD_DATA21 for Mode: ALT2 1 **CSI_DATA05_ALT3** — Selecting Pad: CSI_DATA05 for Mode: ALT3

30.5.333 ECSPI2_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 544h offset = 20E_0544h



IOMUXC_ECSPI2_SCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi2, In Pin: cspi_clk_in 0 **CSI_DATA00_ALT3** — Selecting Pad: CSI_DATA00 for Mode: ALT3 1 **UART4_TX_DATA_ALT8** — Selecting Pad: UART4_TX_DATA for Mode: ALT8

30.5.334 ECSPI2_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MISO_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 548h offset = 20E_0548h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

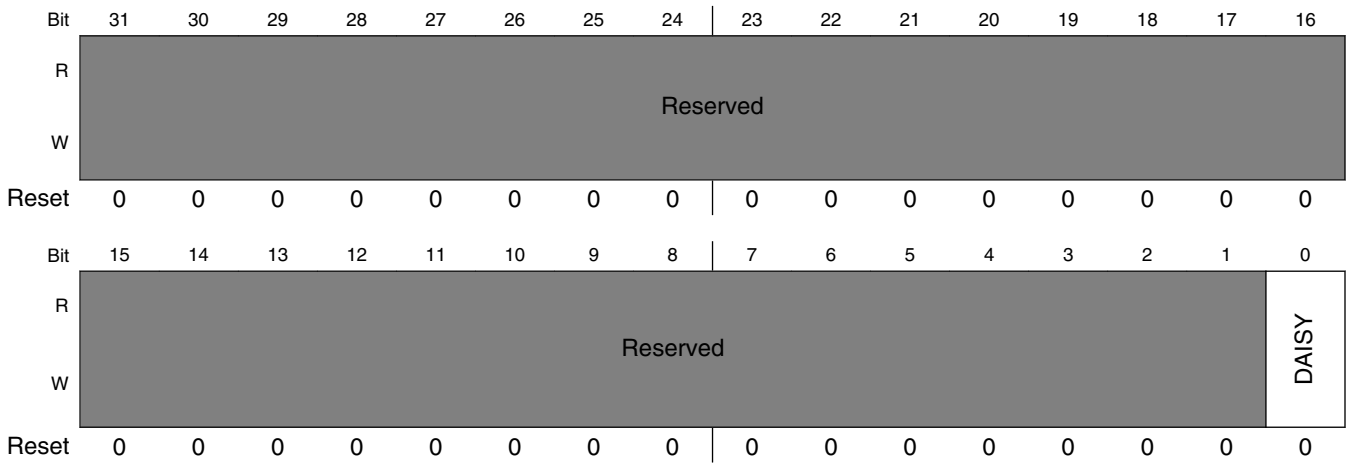
IOMUXC_ECSPI2_MISO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi2, In Pin: miso 0 **CSI_DATA03_ALT3** — Selecting Pad: CSI_DATA03 for Mode: ALT3 1 **UART5_RX_DATA_ALT8** — Selecting Pad: UART5_RX_DATA for Mode: ALT8

30.5.335 ECSPI2_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MOSI_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 54Ch offset = 20E_054Ch



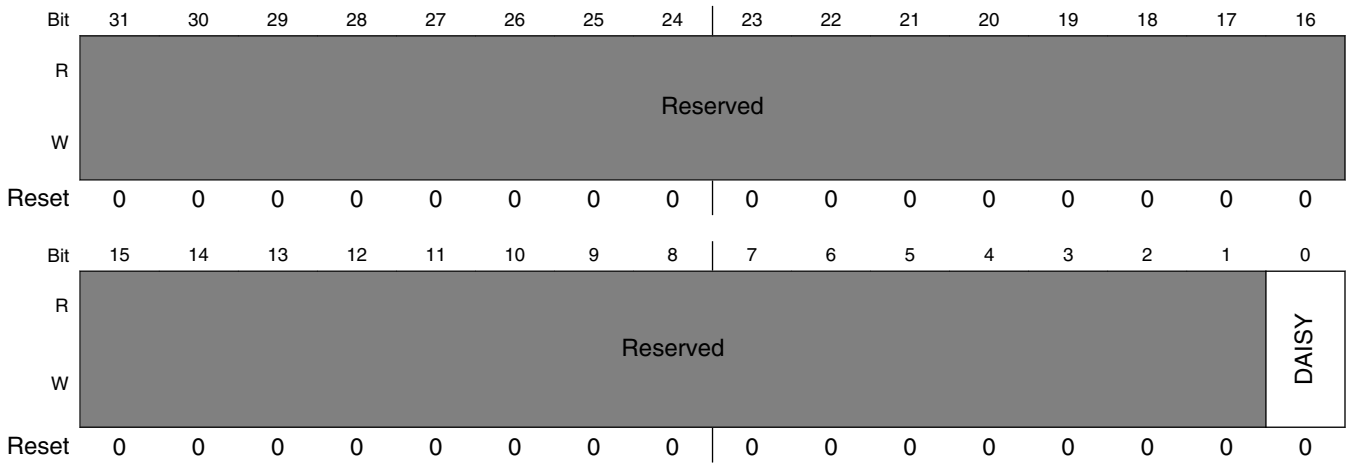
IOMUXC_ECSPI2_MOSI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi2, In Pin: mosi 0 **UART5_TX_DATA_ALT8** — Selecting Pad: UART5_TX_DATA for Mode: ALT8 1 **CSI_DATA02_ALT3** — Selecting Pad: CSI_DATA02 for Mode: ALT3

30.5.336 ECSPI2_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SS0_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 550h offset = 20E_0550h



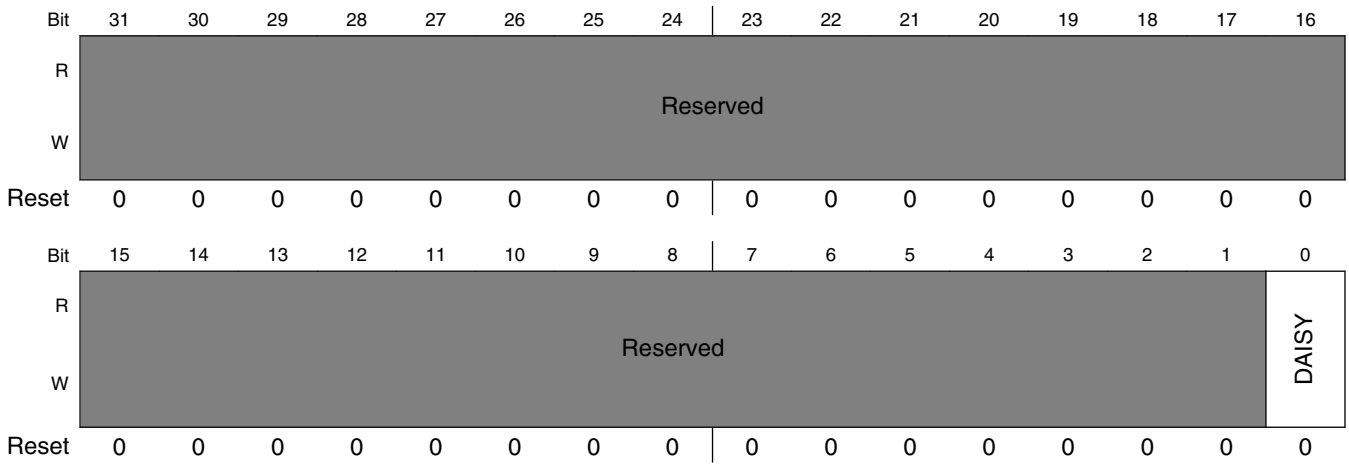
IOMUXC_ECSPI2_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi2, In Pin: ss_b0 0 **CSI_DATA01_ALT3** — Selecting Pad: CSI_DATA01 for Mode: ALT3 1 **UART4_RX_DATA_ALT8** — Selecting Pad: UART4_RX_DATA for Mode: ALT8

30.5.337 ECSPI3_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 554h offset = 20E_0554h



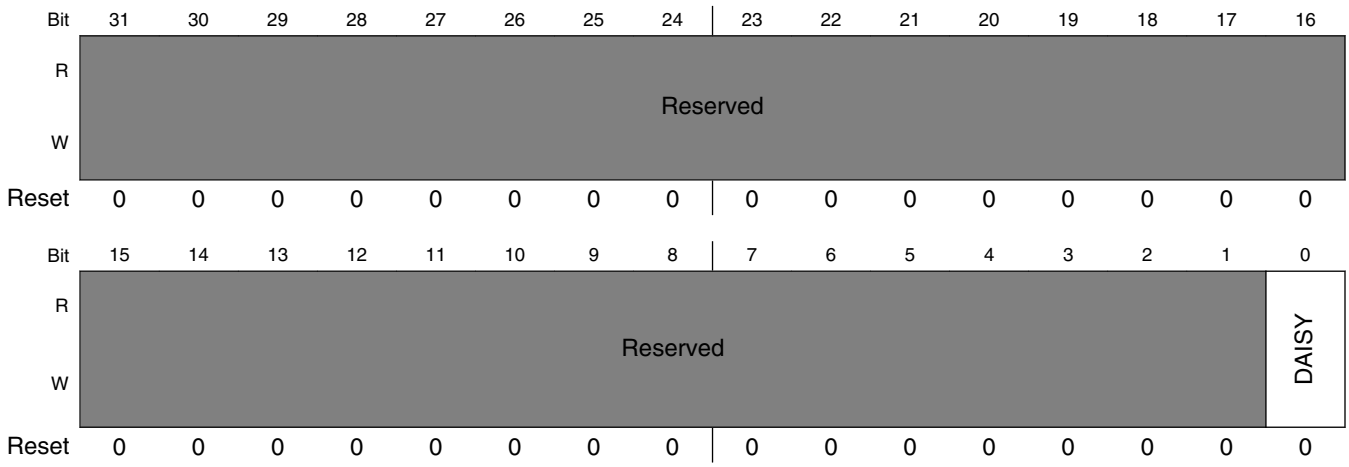
IOMUXC_ECSPI3_SCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi3, In Pin: cspi_clk_in 0 **UART2_RX_DATA_ALT8** — Selecting Pad: UART2_RX_DATA for Mode: ALT8 1 **NAND_CE0_B_ALT3** — Selecting Pad: NAND_CE0_B for Mode: ALT3

30.5.338 ECSPI3_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MISO_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 558h offset = 20E_0558h



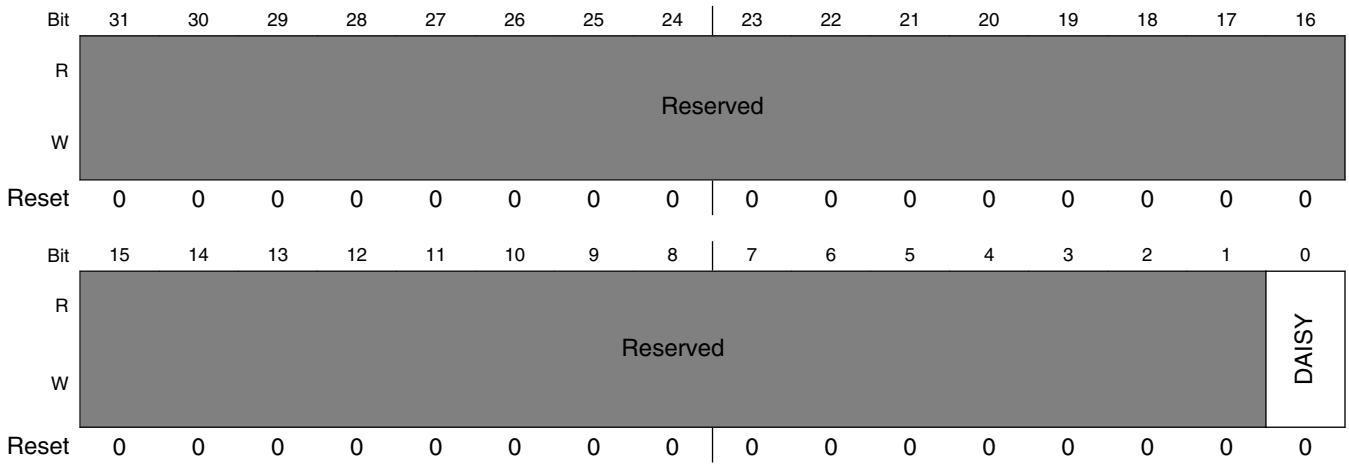
IOMUXC_ECSPI3_MISO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi3, In Pin: miso 0 **UART2_RTS_B_ALT8** — Selecting Pad: UART2_RTS_B for Mode: ALT8 1 **NAND_CLE_ALT3** — Selecting Pad: NAND_CLE for Mode: ALT3

30.5.339 ECSPI3_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MOSI_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 55Ch offset = 20E_055Ch



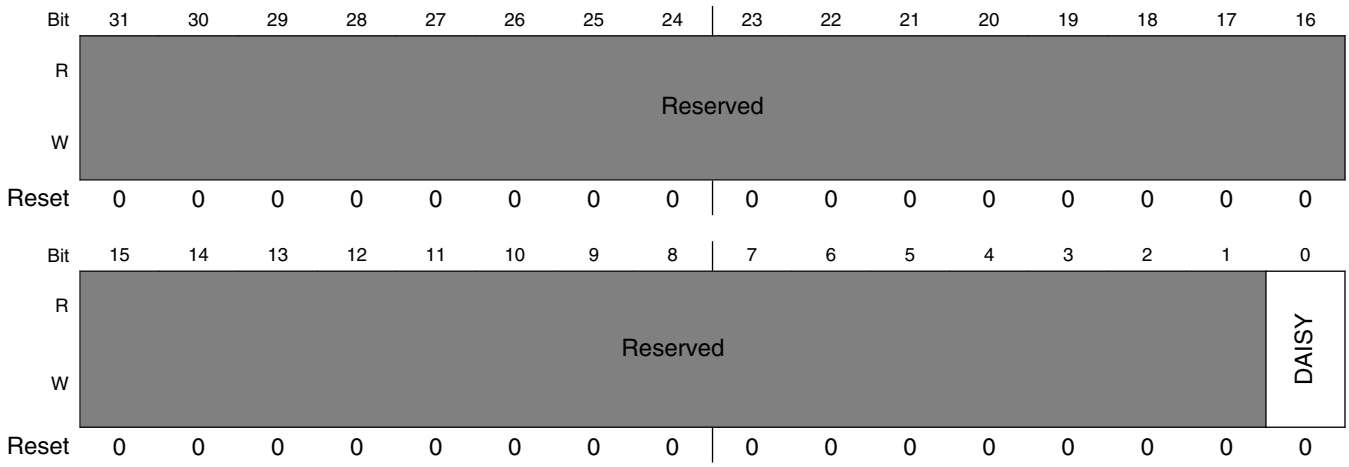
IOMUXC_ECSPI3_MOSI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi3, In Pin: mosi 0 **UART2_CTS_B_ALT8** — Selecting Pad: UART2_CTS_B for Mode: ALT8 1 **NAND_CE1_B_ALT3** — Selecting Pad: NAND_CE1_B for Mode: ALT3

30.5.340 ECSPI3_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SS0_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 560h offset = 20E_0560h



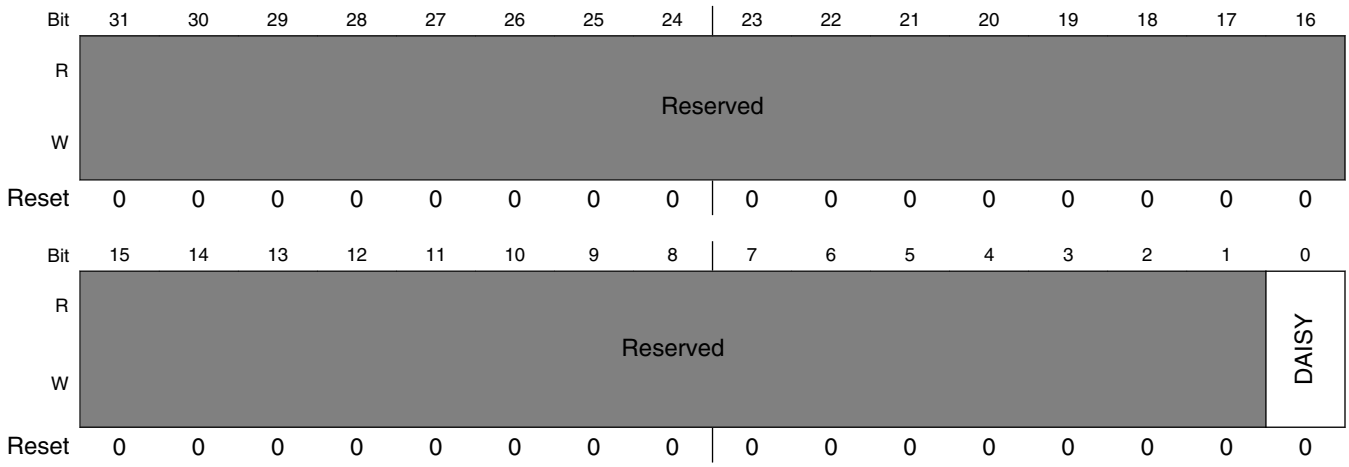
IOMUXC_ECSPI3_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi3, In Pin: ss_b0 0 **UART2_TX_DATA_ALT8** — Selecting Pad: UART2_TX_DATA for Mode: ALT8 1 **NAND_READY_B_ALT3** — Selecting Pad: NAND_READY_B for Mode: ALT3

30.5.341 ECSPI4_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 564h offset = 20E_0564h



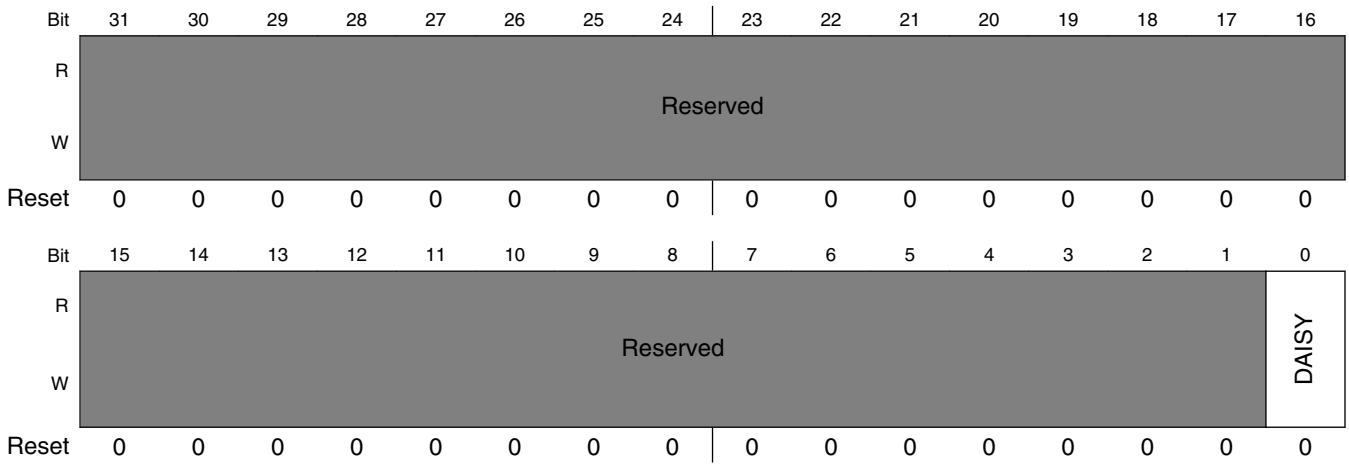
IOMUXC_ECSPI4_SCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi4, In Pin: cspi_clk_in 0 **ENET2_TX_DATA1_ALT3** — Selecting Pad: ENET2_TX_DATA1 for Mode: ALT3 1 **NAND_DATA04_ALT3** — Selecting Pad: NAND_DATA04 for Mode: ALT3

30.5.342 ECSPI4_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MISO_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 568h offset = 20E_0568h



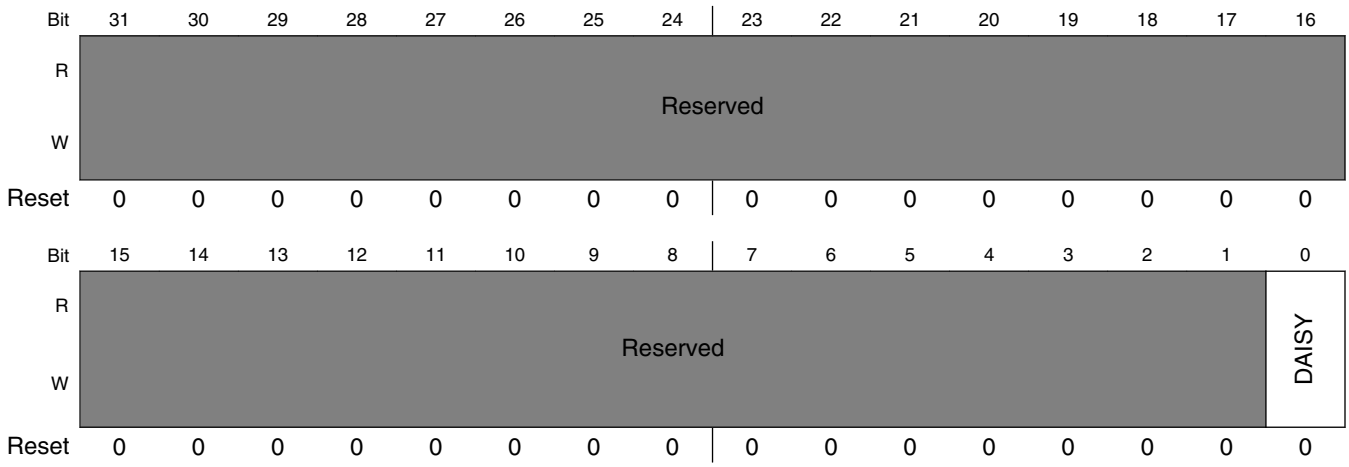
IOMUXC_ECSPI4_MISO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi4, In Pin: miso 0 **ENET2_TX_CLK_ALT3** — Selecting Pad: ENET2_TX_CLK for Mode: ALT3 1 **NAND_DATA06_ALT3** — Selecting Pad: NAND_DATA06 for Mode: ALT3

30.5.343 ECSPI4_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MOSI_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 56Ch offset = 20E_056Ch



IOMUXC_ECSPI4_MOSI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi4, In Pin: mosi 0 **ENET2_TX_EN_ALT3** — Selecting Pad: ENET2_TX_EN for Mode: ALT3 1 **NAND_DATA05_ALT3** — Selecting Pad: NAND_DATA05 for Mode: ALT3

30.5.344 ECSPI4_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SS0_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 570h offset = 20E_0570h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_ECSPI4_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ecspi4, In Pin: ss_b0 0 **ENET2_RX_ER_ALT3** — Selecting Pad: ENET2_RX_ER for Mode: ALT3 1 **NAND_DATA07_ALT3** — Selecting Pad: NAND_DATA07 for Mode: ALT3

30.5.345 ENET1_REF_CLK1_SELECT_INPUT DAISY Register (IOMUXC_ENET1_REF_CLK1_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 574h offset = 20E_0574h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_ENET1_REF_CLK1_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet1, In Pin: rmii 00 **GPIO1_IO00_ALT3** — Selecting Pad: GPIO1_IO00 for Mode: ALT3 01 **GPIO1_IO04_ALT0** — Selecting Pad: GPIO1_IO04 for Mode: ALT0 10 **ENET1_TX_CLK_ALT4** — Selecting Pad: ENET1_TX_CLK for Mode: ALT4

30.5.346 ENET1_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET1_MAC0_MDIO_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 578h offset = 20E_0578h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_ENET1_MAC0_MDIO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet1, In Pin: mac0_mdio 0 **GPIO1_IO06_ALT0** — Selecting Pad: GPIO1_IO06 for Mode: ALT0 1 **ENET2_RX_DATA0_ALT4** — Selecting Pad: ENET2_RX_DATA0 for Mode: ALT4

30.5.347 ENET2_REF_CLK2_SELECT_INPUT DAISY Register (IOMUXC_ENET2_REF_CLK2_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 57Ch offset = 20E_057Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

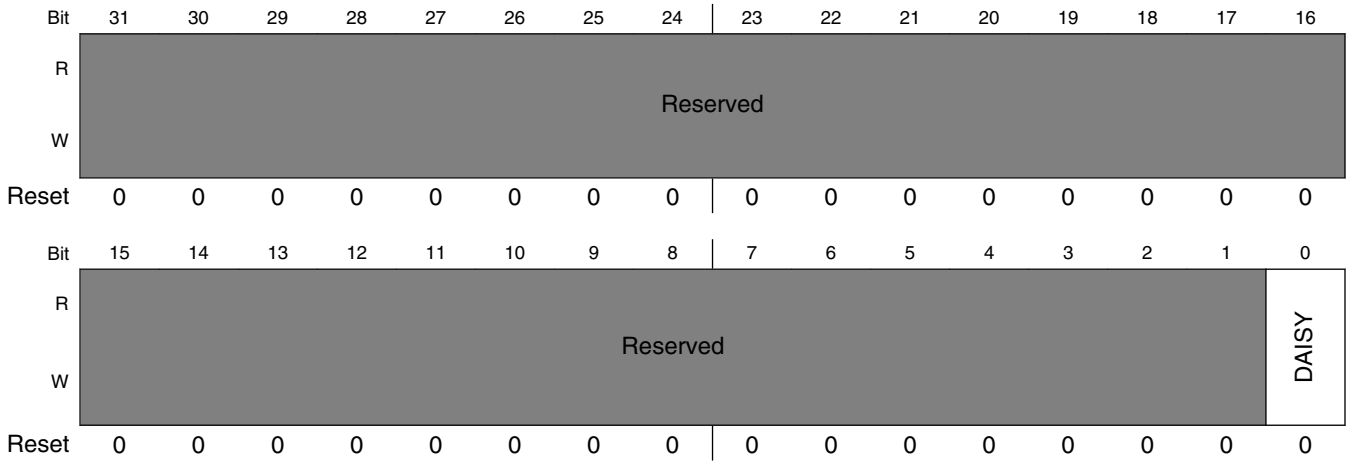
IOMUXC_ENET2_REF_CLK2_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: rmii 00 **GPIO1_IO01_ALT3** — Selecting Pad: GPIO1_IO01 for Mode: ALT3 01 **GPIO1_IO05_ALT0** — Selecting Pad: GPIO1_IO05 for Mode: ALT0 10 **ENET2_TX_CLK_ALT4** — Selecting Pad: ENET2_TX_CLK for Mode: ALT4

30.5.348 ENET2_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET2_MAC0_MDIO_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 580h offset = 20E_0580h



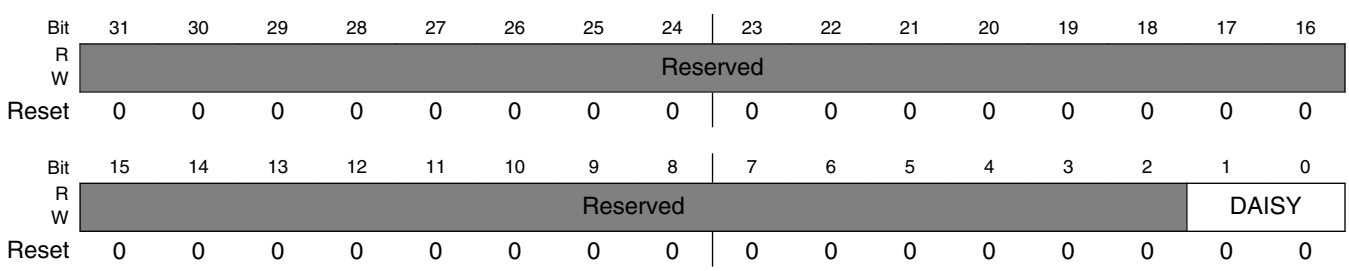
IOMUXC_ENET2_MAC0_MDIO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: mac0_mdio 0 **GPIO1_IO06_ALT1** — Selecting Pad: GPIO1_IO06 for Mode: ALT1 1 **ENET1_TX_DATA1_ALT4** — Selecting Pad: ENET1_TX_DATA1 for Mode: ALT4

30.5.349 FLEXCAN1_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN1_RX_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 584h offset = 20E_0584h



IOMUXC_FLEXCAN1_RX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexcan1, In Pin: RX 00 **UART3_RTS_B_ALT2** — Selecting Pad: UART3_RTS_B for Mode: ALT2 01 **ENET1_RX_DATA1_ALT4** — Selecting Pad: ENET1_RX_DATA1 for Mode: ALT4 10 **LCD_DATA09_ALT8** — Selecting Pad: LCD_DATA09 for Mode: ALT8 11 **SD1_DATA1_ALT3** — Selecting Pad: SD1_DATA1 for Mode: ALT3

30.5.350 FLEXCAN2_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN2_RX_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 588h offset = 20E_0588h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

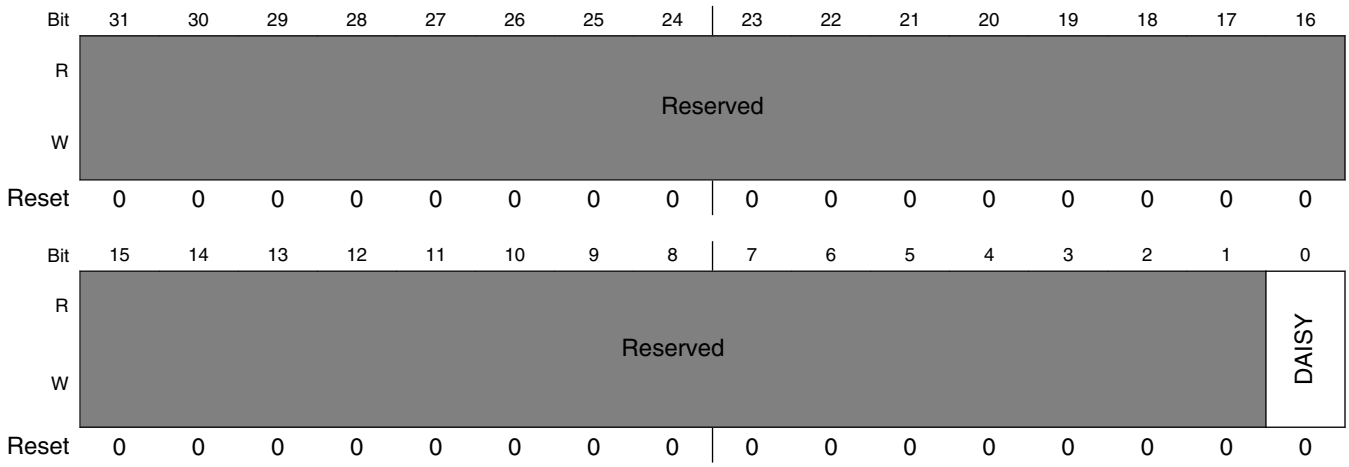
IOMUXC_FLEXCAN2_RX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexcan2, In Pin: RX 00 **UART2_RTS_B_ALT2** — Selecting Pad: UART2_RTS_B for Mode: ALT2 01 **ENET1_TX_DATA0_ALT4** — Selecting Pad: ENET1_TX_DATA0 for Mode: ALT4 10 **LCD_DATA11_ALT8** — Selecting Pad: LCD_DATA11 for Mode: ALT8 11 **SD1_DATA3_ALT3** — Selecting Pad: SD1_DATA3 for Mode: ALT3

30.5.351 GPT1_CAPTURE1_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CAPTURE1_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 58Ch offset = 20E_058Ch



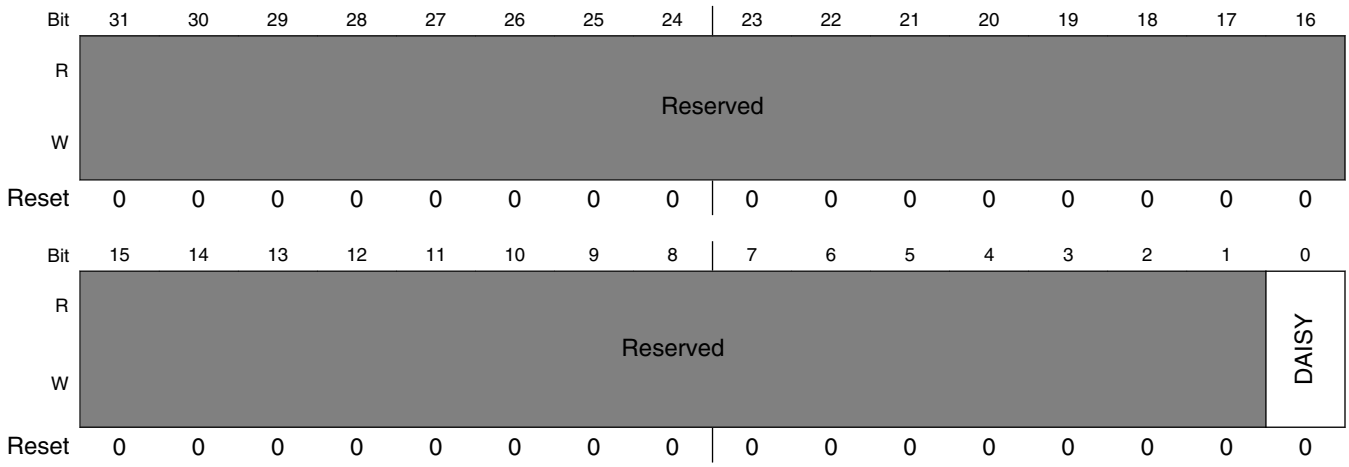
IOMUXC_GPT1_CAPTURE1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt1, In Pin: CAPTURE1 0 **GPIO1_IO00_ALT1** — Selecting Pad: GPIO1_IO00 for Mode: ALT1 1 **UART2_TX_DATA_ALT4** — Selecting Pad: UART2_TX_DATA for Mode: ALT4

30.5.352 GPT1_CAPTURE2_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CAPTURE2_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 590h offset = 20E_0590h



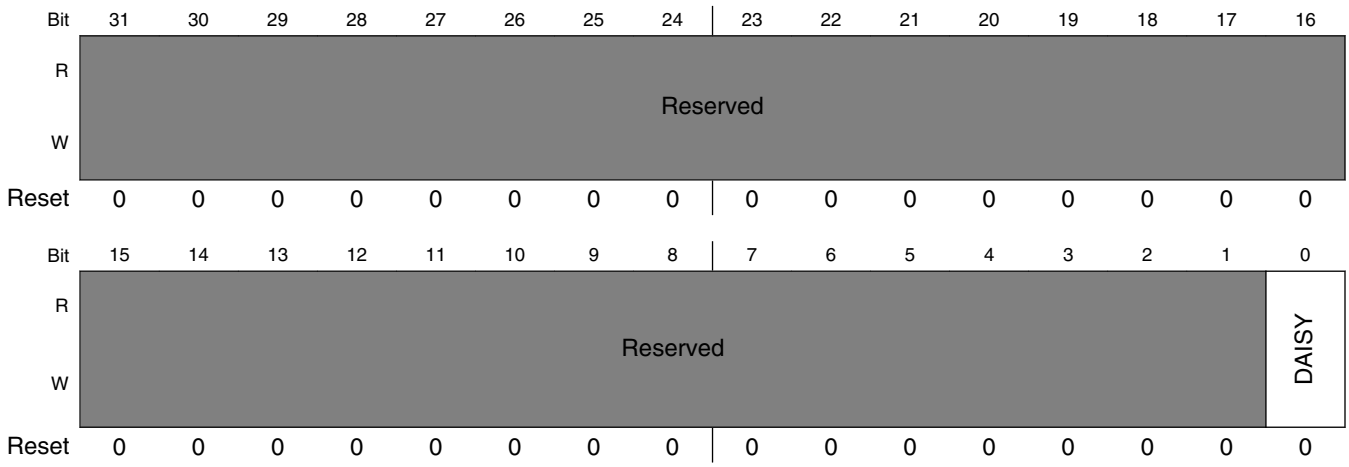
IOMUXC_GPT1_CAPTURE2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt1, In Pin: CAPTURE2 0 **UART2_RX_DATA_ALT4** — Selecting Pad: UART2_RX_DATA for Mode: ALT4 1 **ENET1_RX_ER_ALT8** — Selecting Pad: ENET1_RX_ER for Mode: ALT8

30.5.353 GPT1_CLK_SELECT_INPUT DAISY Register (IOMUXC_GPT1_CLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 594h offset = 20E_0594h



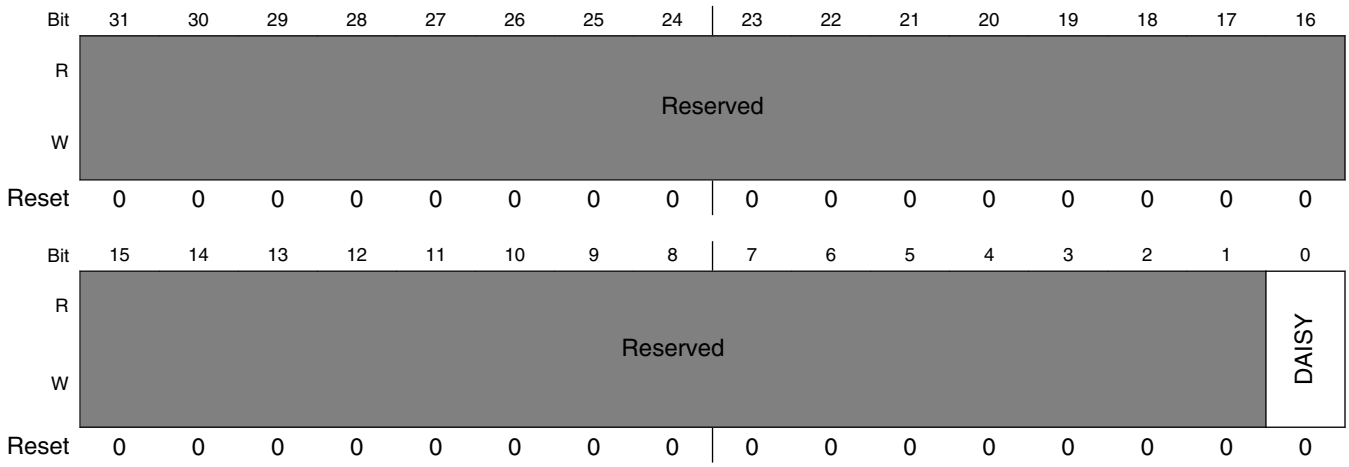
IOMUXC_GPT1_CLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt1, In Pin: CLK 0 **UART1_RX_DATA_ALT4** — Selecting Pad: UART1_RX_DATA for Mode: ALT4 1 **ENET1_TX_CLK_ALT8** — Selecting Pad: ENET1_TX_CLK for Mode: ALT8

30.5.354 GPT2_CAPTURE1_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CAPTURE1_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 598h offset = 20E_0598h



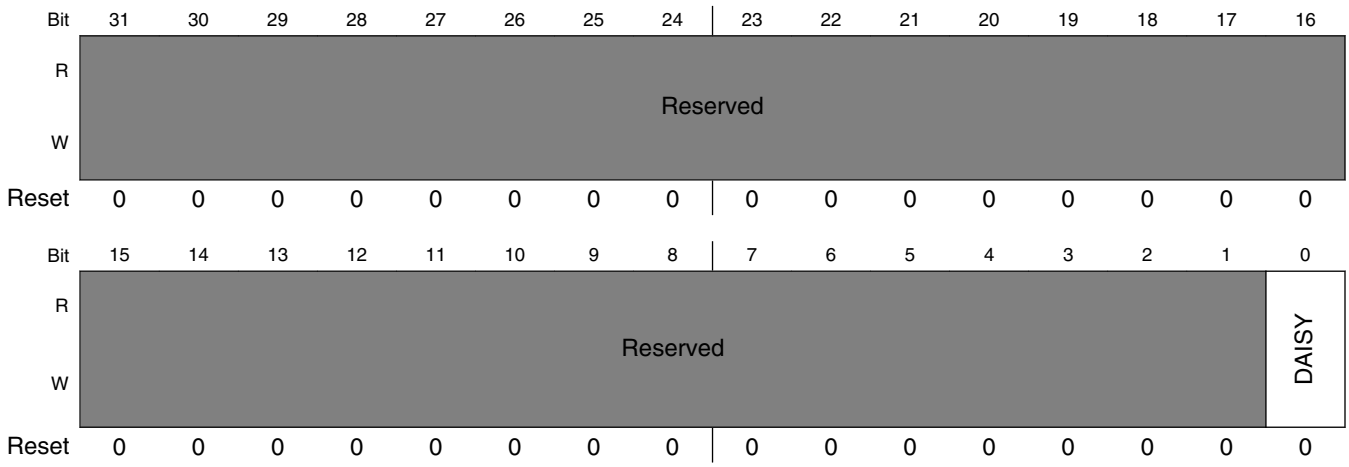
IOMUXC_GPT2_CAPTURE1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt2, In Pin: CAPTURE1 0 **JTAG_TMS_ALT1** — Selecting Pad: JTAG_TMS for Mode: ALT1 1 **SD1_DATA2_ALT1** — Selecting Pad: SD1_DATA2 for Mode: ALT1

30.5.355 GPT2_CAPTURE2_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CAPTURE2_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 59Ch offset = 20E_059Ch



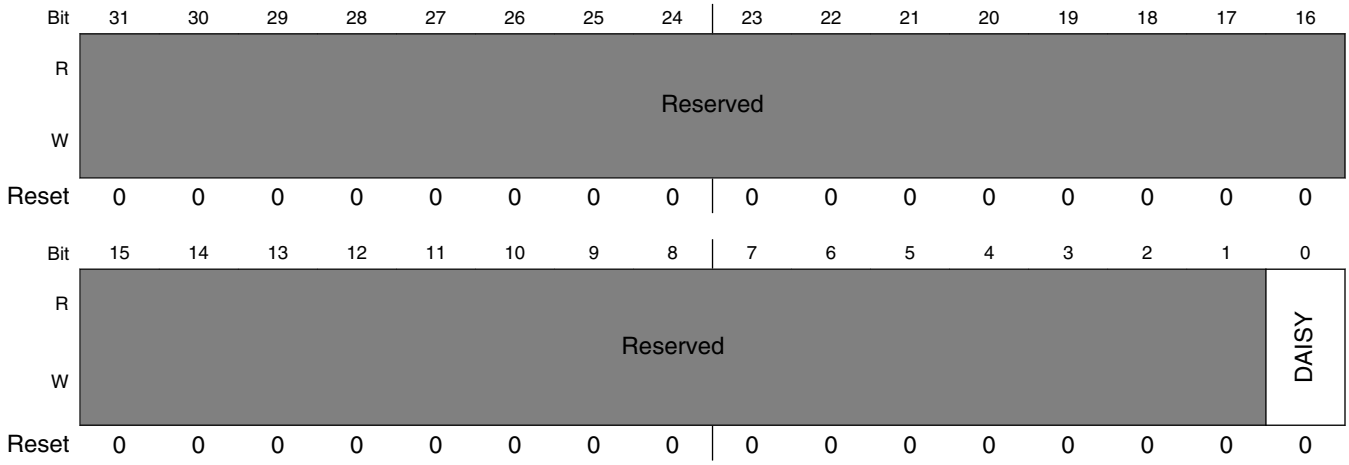
IOMUXC_GPT2_CAPTURE2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt2, In Pin: CAPTURE2 0 **JTAG_TDO_ALT1** — Selecting Pad: JTAG_TDO for Mode: ALT1 1 **SD1_DATA3_ALT1** — Selecting Pad: SD1_DATA3 for Mode: ALT1

30.5.356 GPT2_CLK_SELECT_INPUT DAISY Register (IOMUXC_GPT2_CLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5A0h offset = 20E_05A0h



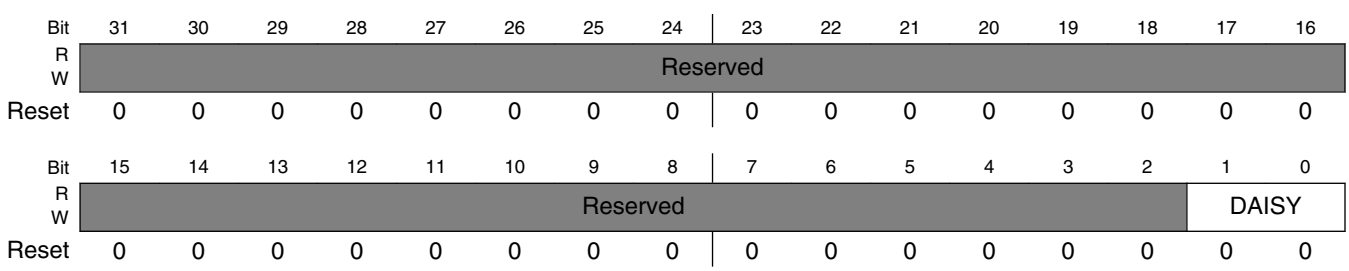
IOMUXC_GPT2_CLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt2, In Pin: CLK 0 **JTAG_MOD_ALT1** — Selecting Pad: JTAG_MOD for Mode: ALT1 1 **SD1_DATA1_ALT1** — Selecting Pad: SD1_DATA1 for Mode: ALT1

30.5.357 I2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SCL_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5A4h offset = 20E_05A4h



IOMUXC_I2C1_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c1, In Pin: scl_in 00 **GPIO1_IO02_ALT0** — Selecting Pad: GPIO1_IO02 for Mode: ALT0 01 **UART4_TX_DATA_ALT2** — Selecting Pad: UART4_TX_DATA for Mode: ALT2 10 **CSI_PIXCLK_ALT3** — Selecting Pad: CSI_PIXCLK for Mode: ALT3

30.5.358 I2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SDA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5A8h offset = 20E_05A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_I2C1_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c1, In Pin: sda_in 00 **CSI_MCLK_ALT3** — Selecting Pad: CSI_MCLK for Mode: ALT3 01 **GPIO1_IO03_ALT0** — Selecting Pad: GPIO1_IO03 for Mode: ALT0 10 **UART4_RX_DATA_ALT2** — Selecting Pad: UART4_RX_DATA for Mode: ALT2

30.5.359 I2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SCL_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5ACh offset = 20E_05ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_I2C2_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c2, In Pin: scl_in 00 **CSI_HSYNC_ALT3** — Selecting Pad: CSI_HSYNC for Mode: ALT3 01 **GPIO1_IO00_ALT0** — Selecting Pad: GPIO1_IO00 for Mode: ALT0 10 **UART5_TX_DATA_ALT2** — Selecting Pad: UART5_TX_DATA for Mode: ALT2

30.5.360 I2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SDA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5B0h offset = 20E_05B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_I2C2_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c2, In Pin: sda_in 00 **CSI_VSYNC_ALT3** — Selecting Pad: CSI_VSYNC for Mode: ALT3 01 **GPIO1_IO01_ALT0** — Selecting Pad: GPIO1_IO01 for Mode: ALT0 10 **UART5_RX_DATA_ALT2** — Selecting Pad: UART5_RX_DATA for Mode: ALT2

30.5.361 I2C3_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SCL_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5B4h offset = 20E_05B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_I2C3_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c3, In Pin: scl_in 00 **UART1_TX_DATA_ALT2** — Selecting Pad: UART1_TX_DATA for Mode: ALT2 01 **ENET2_RX_DATA0_ALT3** — Selecting Pad: ENET2_RX_DATA0 for Mode: ALT3 10 **LCD_DATA01_ALT4** — Selecting Pad: LCD_DATA01 for Mode: ALT4

30.5.362 I2C3_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SDA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5B8h offset = 20E_05B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_I2C3_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c3, In Pin: sda_in 00 **UART1_RX_DATA_ALT2** — Selecting Pad: UART1_RX_DATA for Mode: ALT2 01 **ENET2_RX_DATA1_ALT3** — Selecting Pad: ENET2_RX_DATA1 for Mode: ALT3 10 **LCD_DATA00_ALT4** — Selecting Pad: LCD_DATA00 for Mode: ALT4

30.5.363 I2C4_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SCL_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5BCh offset = 20E_05BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_I2C4_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c4, In Pin: scl_in 00 **UART2_TX_DATA_ALT2** — Selecting Pad: UART2_TX_DATA for Mode: ALT2 01 **ENET2_RX_EN_ALT3** — Selecting Pad: ENET2_RX_EN for Mode: ALT3 10 **LCD_DATA03_ALT4** — Selecting Pad: LCD_DATA03 for Mode: ALT4

30.5.364 I2C4_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SDA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5C0h offset = 20E_05C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

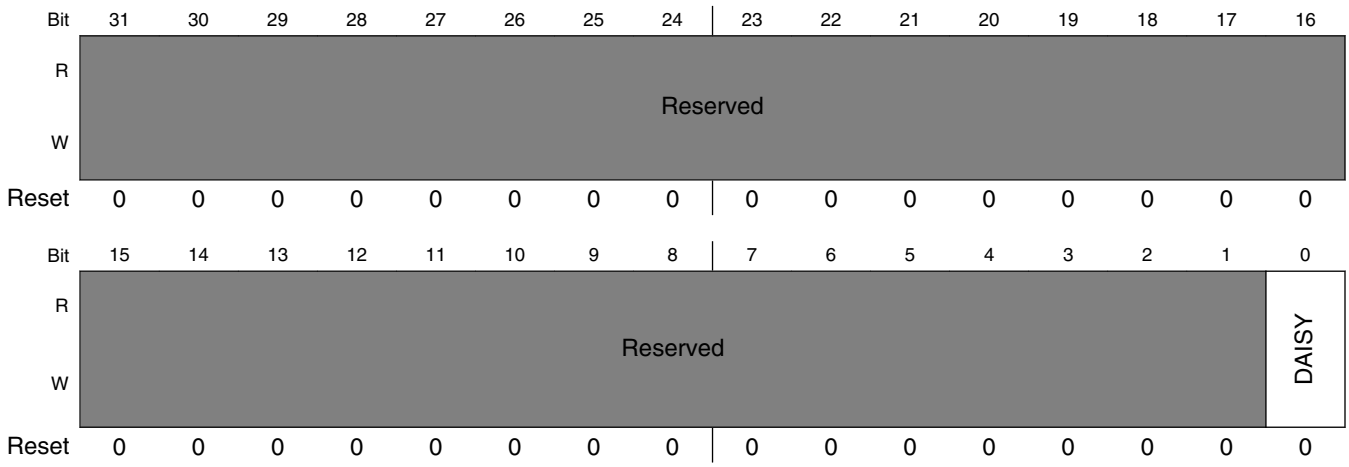
IOMUXC_I2C4_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c4, In Pin: sda_in 00 **UART2_RX_DATA_ALT2** — Selecting Pad: UART2_RX_DATA for Mode: ALT2 01 **ENET2_TX_DATA0_ALT3** — Selecting Pad: ENET2_TX_DATA0 for Mode: ALT3 10 **LCD_DATA02_ALT4** — Selecting Pad: LCD_DATA02 for Mode: ALT4

30.5.365 KPP_COL0_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL0_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5C4h offset = 20E_05C4h



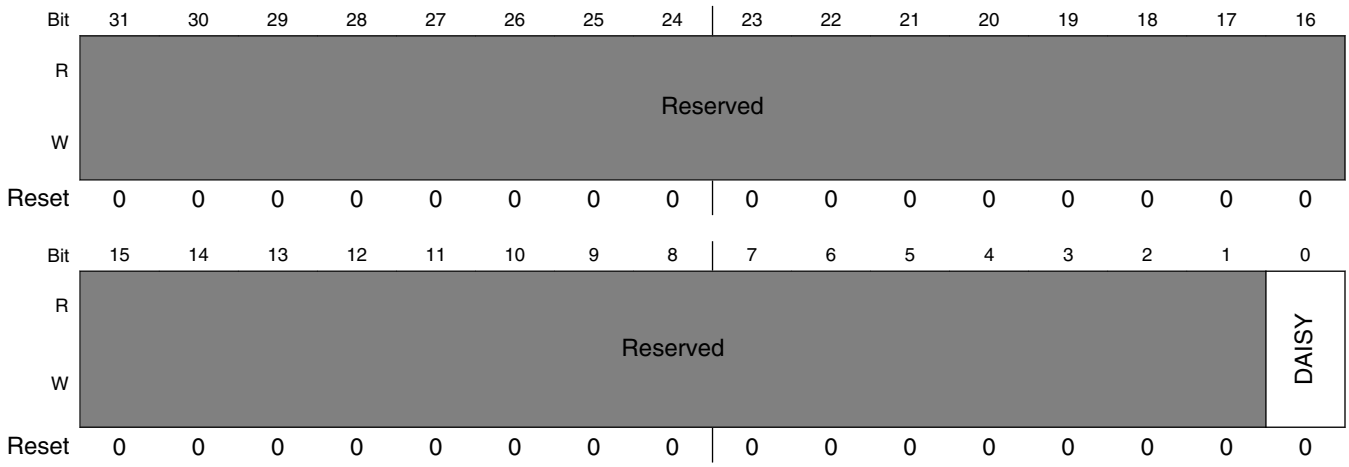
IOMUXC_KPP_COL0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: col0 0 **ENET1_RX_DATA1_ALT6** — Selecting Pad: ENET1_RX_DATA1 for Mode: ALT6 1 **NAND_WE_B_ALT3** — Selecting Pad: NAND_WE_B for Mode: ALT3

30.5.366 KPP_COL1_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL1_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5C8h offset = 20E_05C8h



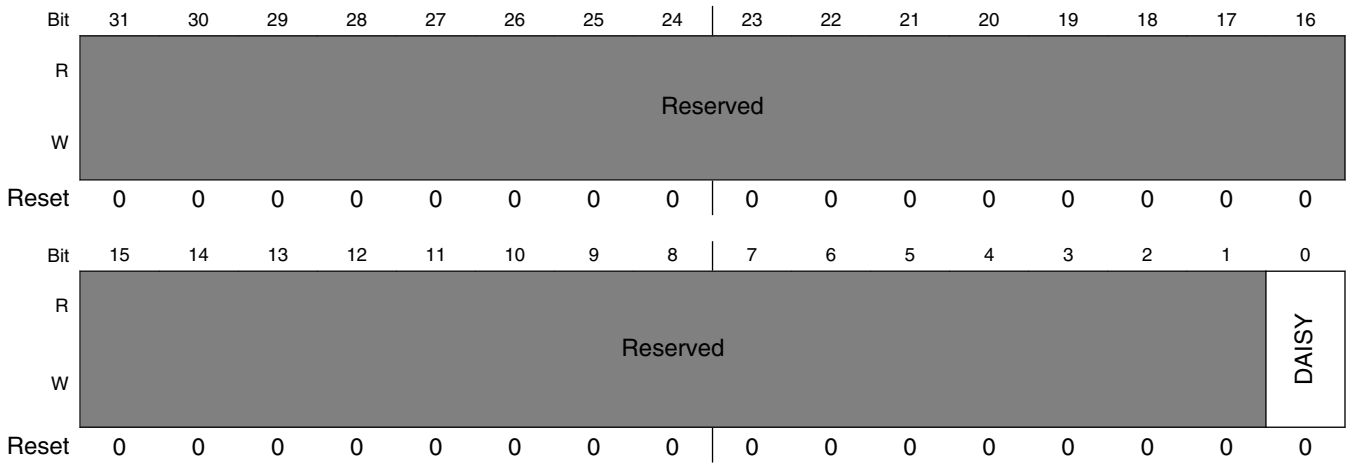
IOMUXC_KPP_COL1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: col1 0 **ENET1_TX_DATA0_ALT6** — Selecting Pad: ENET1_TX_DATA0 for Mode: ALT6 1 **NAND_DATA01_ALT3** — Selecting Pad: NAND_DATA01 for Mode: ALT3

30.5.367 KPP_COL2_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL2_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5CCh offset = 20E_05CCh



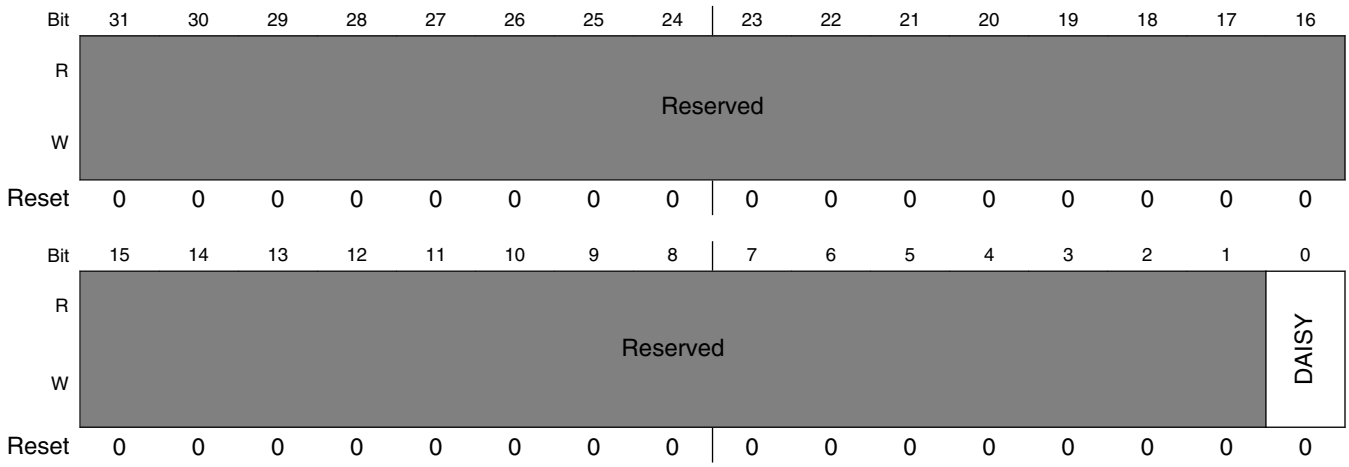
IOMUXC_KPP_COL2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: col2 0 **ENET1_TX_EN_ALT6** — Selecting Pad: ENET1_TX_EN for Mode: ALT6 1 **NAND_DATA03_ALT3** — Selecting Pad: NAND_DATA03 for Mode: ALT3

30.5.368 KPP_ROW0_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW0_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5D0h offset = 20E_05D0h



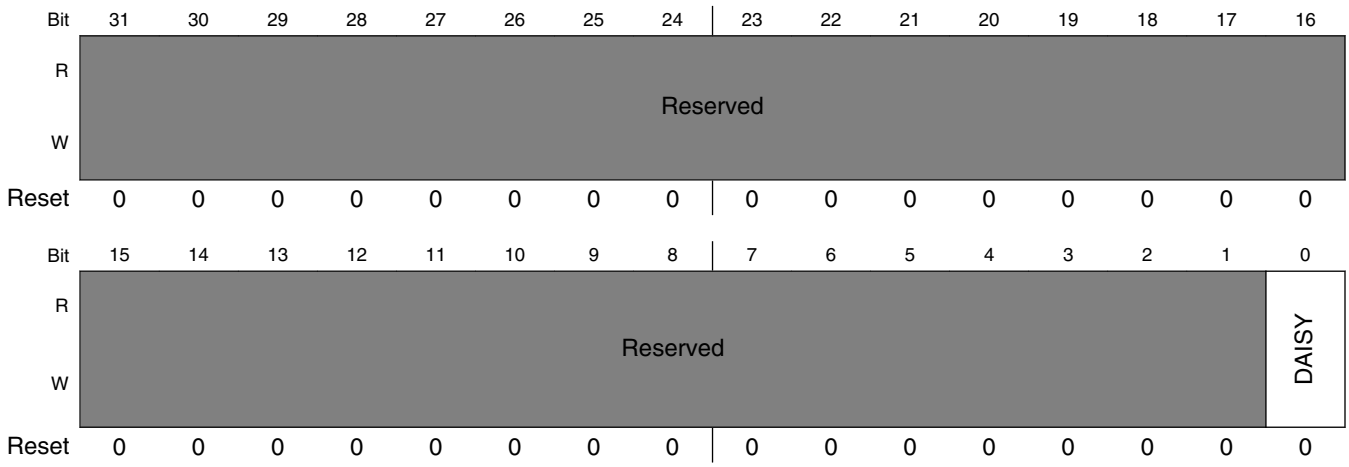
IOMUXC_KPP_ROW0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: row0 0 **ENET1_RX_DATA0_ALT6** — Selecting Pad: ENET1_RX_DATA0 for Mode: ALT6 1 **NAND_RE_B_ALT3** — Selecting Pad: NAND_RE_B for Mode: ALT3

30.5.369 KPP_ROW1_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW1_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5D4h offset = 20E_05D4h



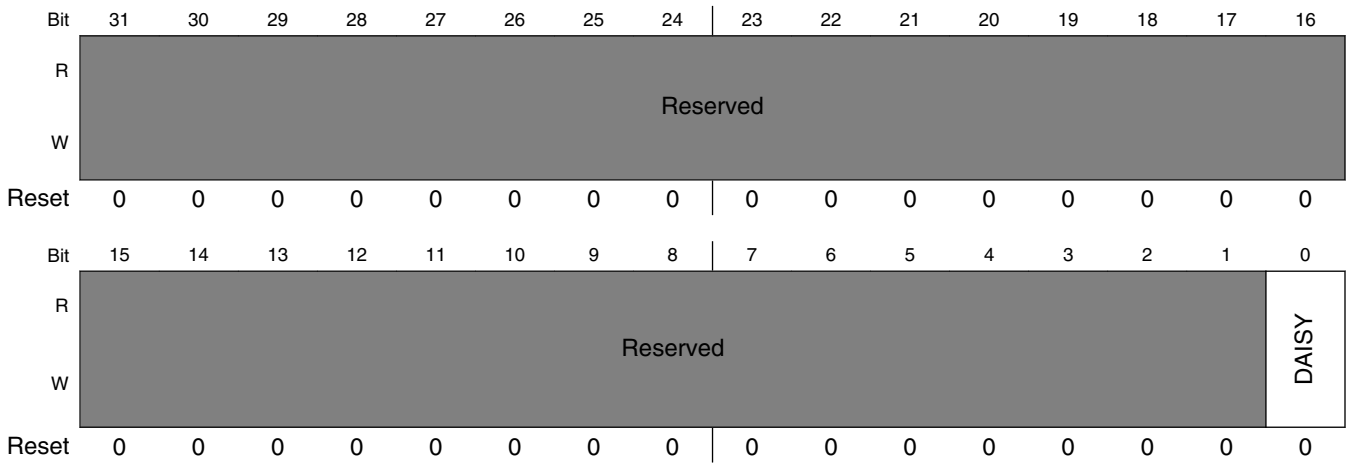
IOMUXC_KPP_ROW1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: row1 0 **ENET1_RX_EN_ALT6** — Selecting Pad: ENET1_RX_EN for Mode: ALT6 1 **NAND_DATA00_ALT3** — Selecting Pad: NAND_DATA00 for Mode: ALT3

30.5.370 KPP_ROW2_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW2_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5D8h offset = 20E_05D8h



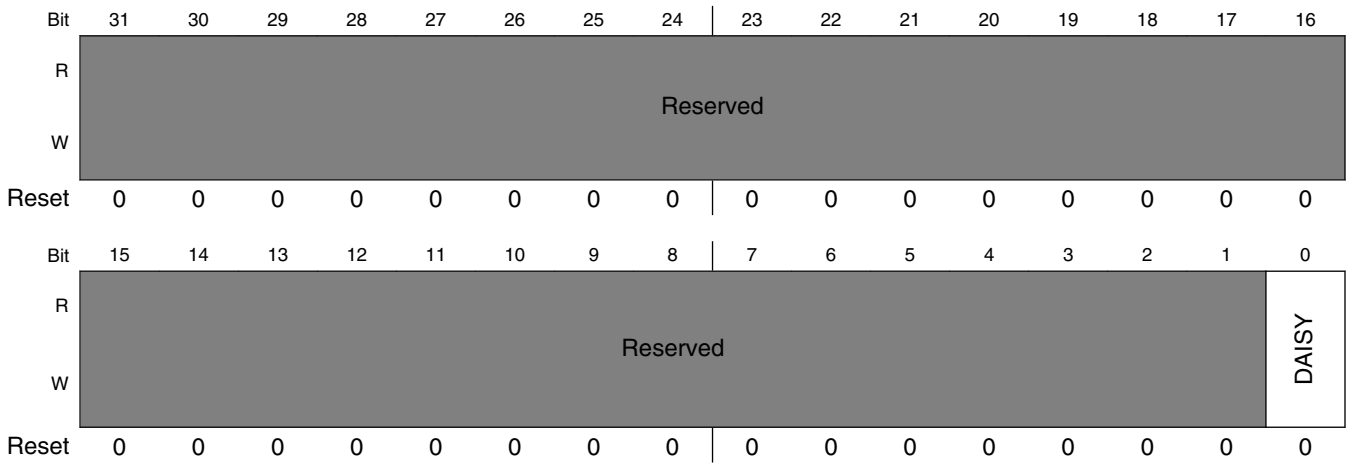
IOMUXC_KPP_ROW2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: row2 0 **ENET1_TX_DATA1_ALT6** — Selecting Pad: ENET1_TX_DATA1 for Mode: ALT6 1 **NAND_DATA02_ALT3** — Selecting Pad: NAND_DATA02 for Mode: ALT3

30.5.371 LCD_BUSY_SELECT_INPUT DAISY Register (IOMUXC_LCD_BUSY_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5DCh offset = 20E_05DCh



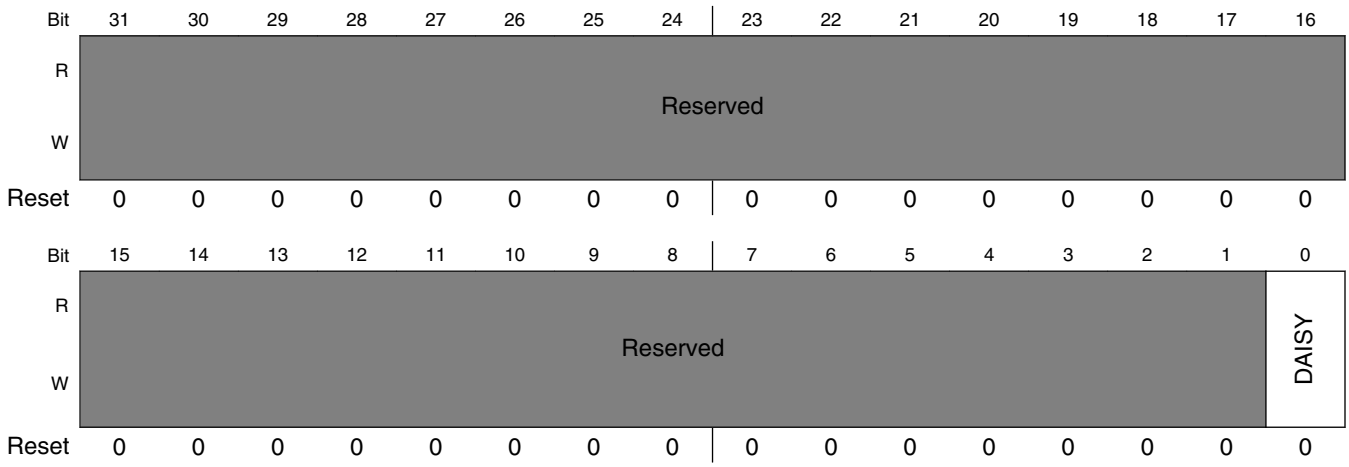
IOMUXC_LCD_BUSY_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lcdif, In Pin: lcdif_busy 0 **LCD_HSYNC_ALT0** — Selecting Pad: LCD_HSYNC for Mode: ALT0 1 **LCD_VSYNC_ALT1** — Selecting Pad: LCD_VSYNC for Mode: ALT1

30.5.372 SAI1_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_MCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5E0h offset = 20E_05E0h



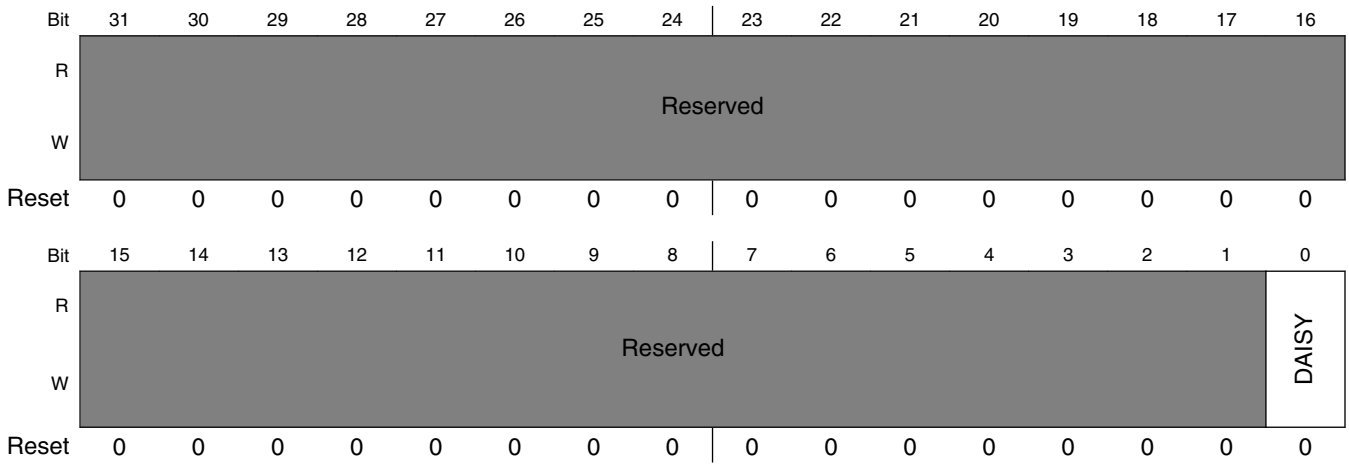
IOMUXC_SAI1_MCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: mclk2 0 **CSI_DATA01_ALT6** — Selecting Pad: CSI_DATA01 for Mode: ALT6 1 **LCD_DATA00_ALT8** — Selecting Pad: LCD_DATA00 for Mode: ALT8

30.5.373 SAI1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5E4h offset = 20E_05E4h



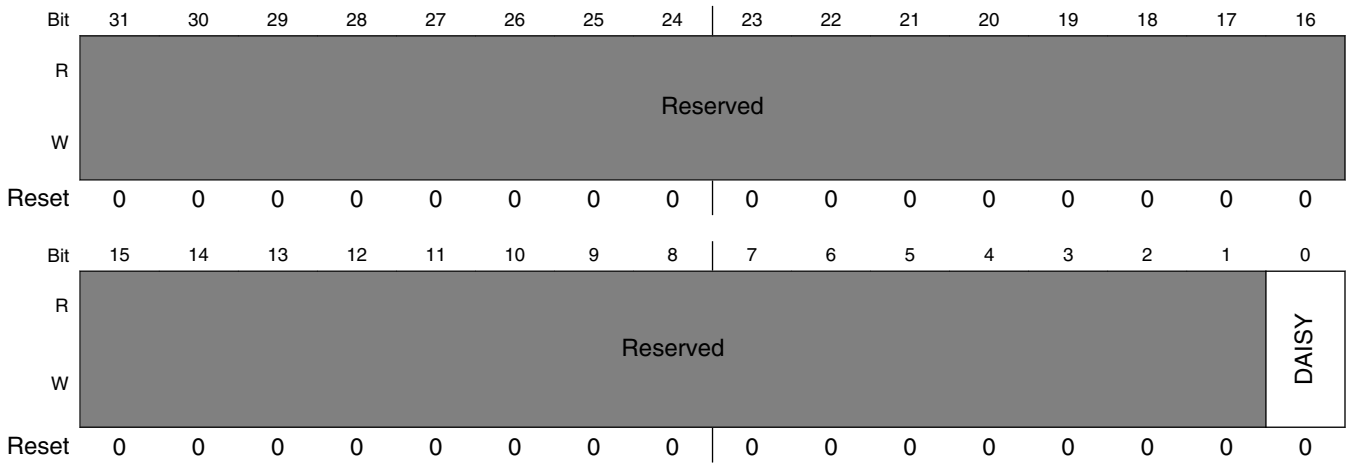
IOMUXC_SAI1_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: RX_DATA0 0 **LCD_DATA03_ALT8** — Selecting Pad: LCD_DATA03 for Mode: ALT8 1 **CSI_DATA06_ALT6** — Selecting Pad: CSI_DATA06 for Mode: ALT6

30.5.374 SAI1_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5E8h offset = 20E_05E8h



IOMUXC_SAI1_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: TX_BCLK 0 **LCD_DATA02_ALT8** — Selecting Pad: LCD_DATA02 for Mode: ALT8 1 **CSI_DATA05_ALT6** — Selecting Pad: CSI_DATA05 for Mode: ALT6

30.5.375 SAI1_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5ECh offset = 20E_05ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

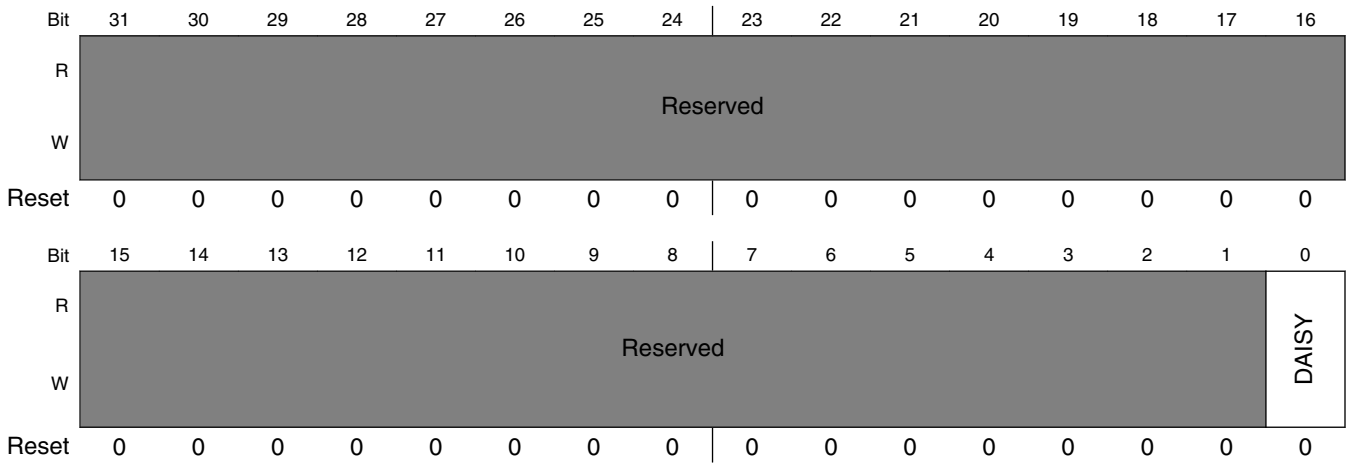
IOMUXC_SAI1_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: TX_SYNC 0 **LCD_DATA01_ALT8** — Selecting Pad: LCD_DATA01 for Mode: ALT8 1 **CSI_DATA04_ALT6** — Selecting Pad: CSI_DATA04 for Mode: ALT6

30.5.376 SAI2_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_MCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5F0h offset = 20E_05F0h



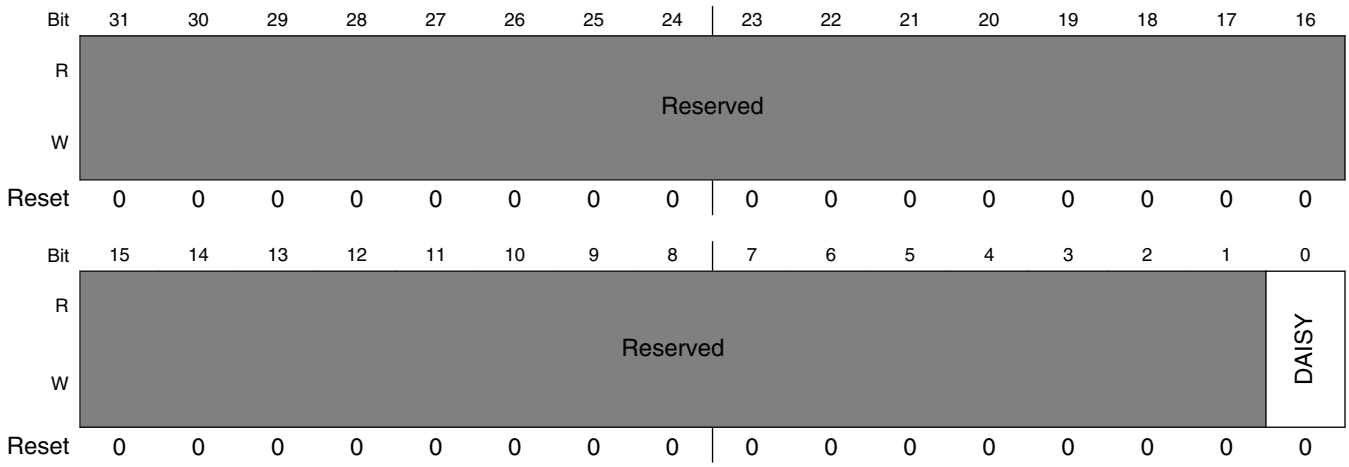
IOMUXC_SAI2_MCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: mclk2 0 **JTAG_TMS_ALT2** — Selecting Pad: JTAG_TMS for Mode: ALT2 1 **SD1_CLK_ALT2** — Selecting Pad: SD1_CLK for Mode: ALT2

30.5.377 SAI2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5F4h offset = 20E_05F4h



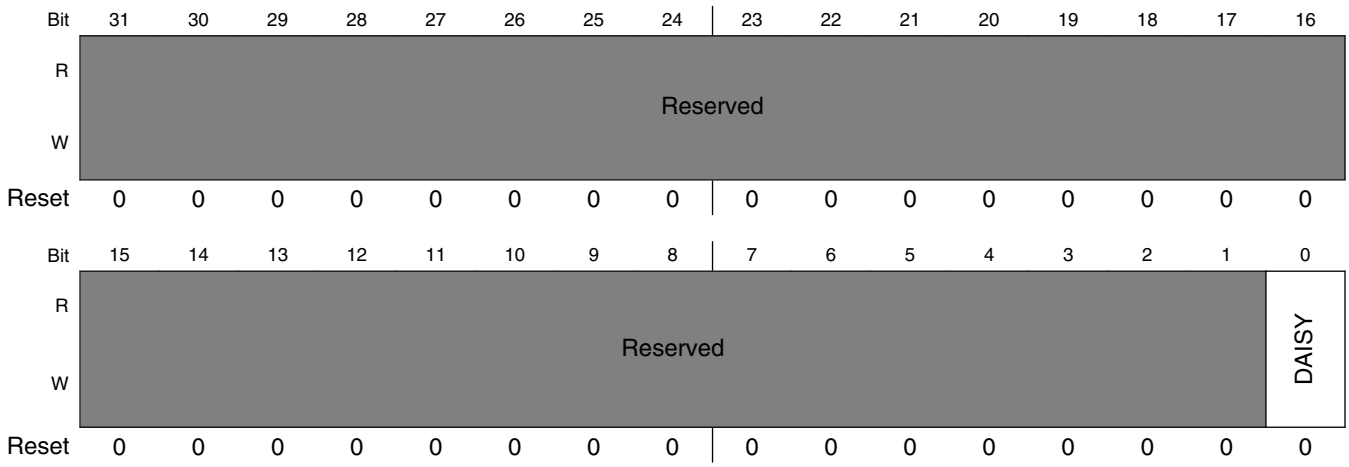
IOMUXC_SAI2_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: RX_DATA0 0 **JTAG_TCK_ALT2** — Selecting Pad: JTAG_TCK for Mode: ALT2 1 **SD1_DATA2_ALT2** — Selecting Pad: SD1_DATA2 for Mode: ALT2

30.5.378 SAI2_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5F8h offset = 20E_05F8h



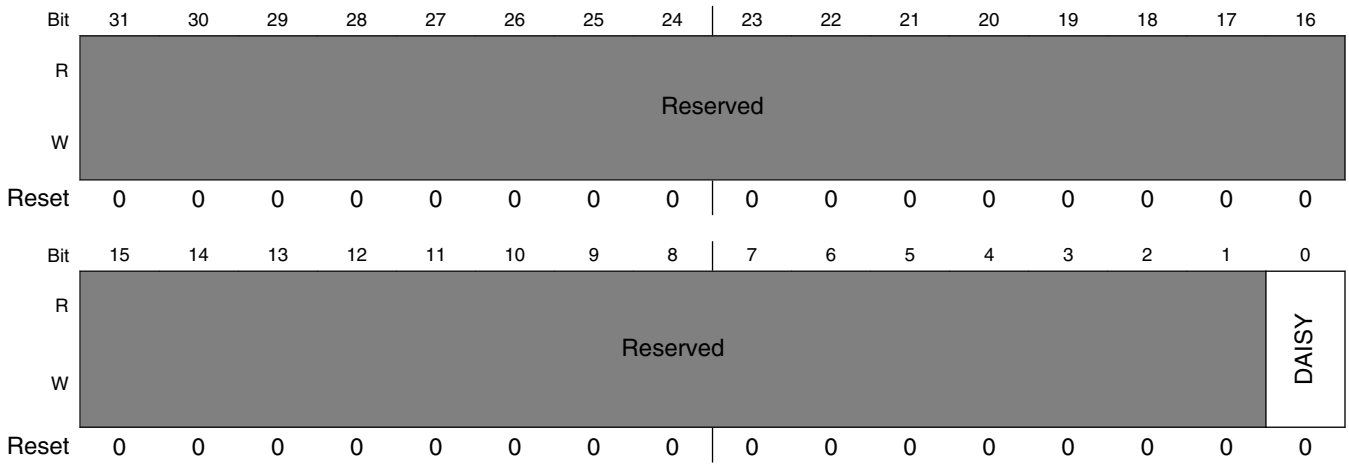
IOMUXC_SAI2_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: TX_BCLK 0 **JTAG_TDI_ALT2** — Selecting Pad: JTAG_TDI for Mode: ALT2 1 **SD1_DATA1_ALT2** — Selecting Pad: SD1_DATA1 for Mode: ALT2

30.5.379 SAI2_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 5FCh offset = 20E_05FCh



IOMUXC_SAI2_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: TX_SYNC 0 **JTAG_TDO_ALT2** — Selecting Pad: JTAG_TDO for Mode: ALT2 1 **SD1_DATA0_ALT2** — Selecting Pad: SD1_DATA0 for Mode: ALT2

30.5.380 SAI3_MCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_MCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 600h offset = 20E_0600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

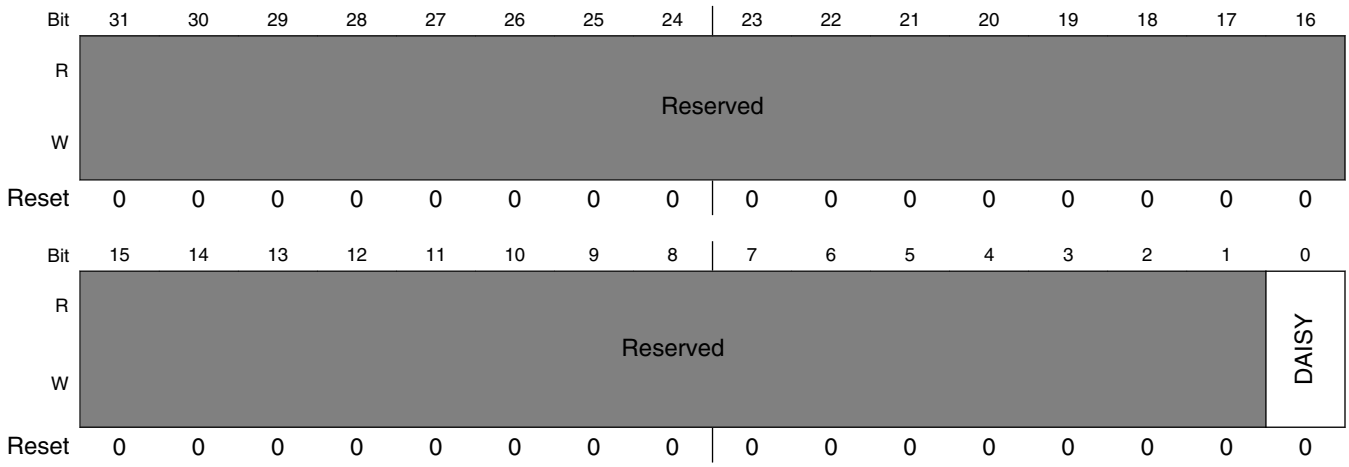
IOMUXC_SAI3_MCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: mclk2 0 **LCD_CLK_ALT3** — Selecting Pad: LCD_CLK for Mode: ALT3 1 **LCD_DATA09_ALT1** — Selecting Pad: LCD_DATA09 for Mode: ALT1

30.5.381 SAI3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 604h offset = 20E_0604h



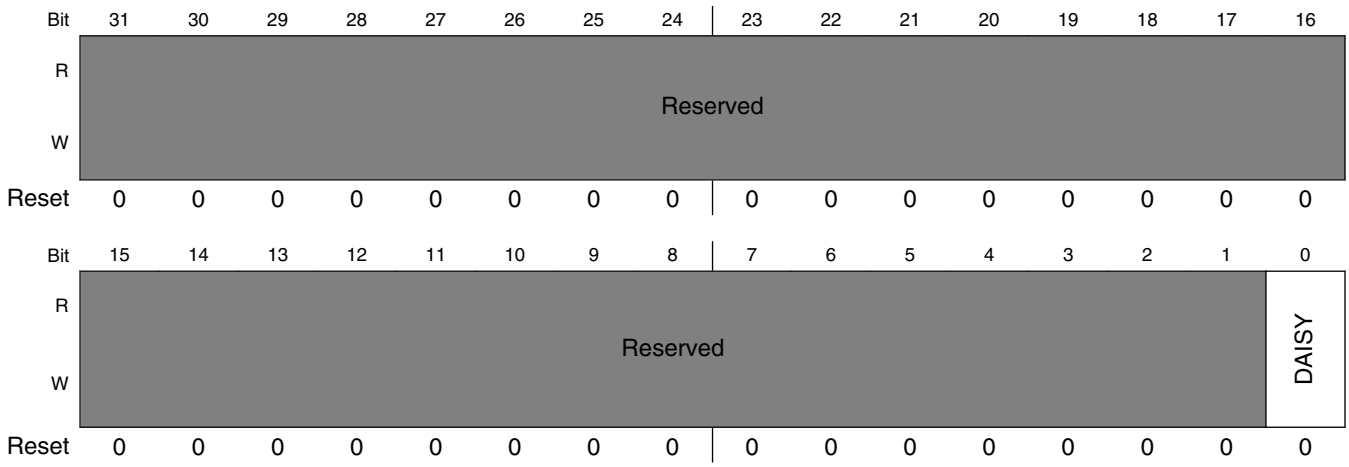
IOMUXC_SAI3_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: RX_DATA0 0 **LCD_VSYNC_ALT3** — Selecting Pad: LCD_VSYNC for Mode: ALT3 1 **LCD_DATA14_ALT1** — Selecting Pad: LCD_DATA14 for Mode: ALT1

30.5.382 SAI3_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 608h offset = 20E_0608h



IOMUXC_SAI3_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: TX_BCLK 0 **LCD_HSYNC_ALT3** — Selecting Pad: LCD_HSYNC for Mode: ALT3 1 **LCD_DATA13_ALT1** — Selecting Pad: LCD_DATA13 for Mode: ALT1

30.5.383 SAI3_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 60Ch offset = 20E_060Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_SAI3_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: TX_SYNC 0 **LCD_ENABLE_ALT3** — Selecting Pad: LCD_ENABLE for Mode: ALT3 1 **LCD_DATA12_ALT1** — Selecting Pad: LCD_DATA12 for Mode: ALT1

30.5.384 SDMA_EVENTS0_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS0_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 610h offset = 20E_0610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

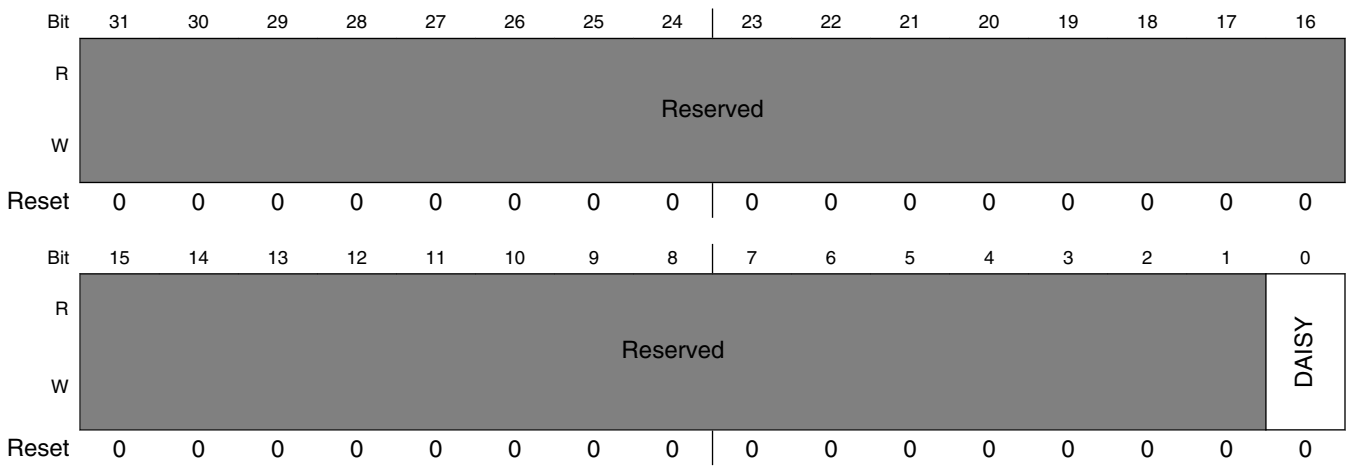
IOMUXC_SDMA_EVENTS0_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: sdma, In Pin: events14 00 **JTAG_MOD_ALT6** — Selecting Pad: JTAG_MOD for Mode: ALT6 01 **GPIO1_IO02_ALT6** — Selecting Pad: GPIO1_IO02 for Mode: ALT6 10 **SD1_CMD_ALT6** — Selecting Pad: SD1_CMD for Mode: ALT6

30.5.385 SDMA_EVENTS1_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS1_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 614h offset = 20E_0614h



IOMUXC_SDMA_EVENTS1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sdma, In Pin: events15 0 **JTAG_TMS_ALT6** — Selecting Pad: JTAG_TMS for Mode: ALT6 1 **NAND_DQS_ALT6** — Selecting Pad: NAND_DQS for Mode: ALT6

30.5.386 SPDIF_IN_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_IN_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 618h offset = 20E_0618h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

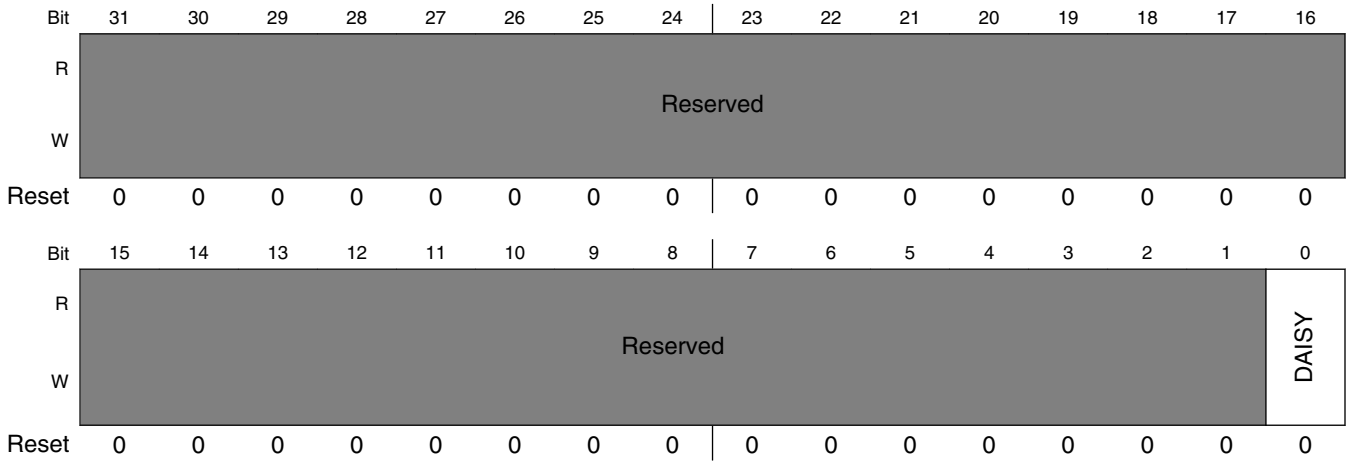
IOMUXC_SPDIF_IN_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: spdif, In Pin: spdif_in1 00 **GPIO1_IO09_ALT2** — Selecting Pad: GPIO1_IO09 for Mode: ALT2 01 **UART1_RX_DATA_ALT8** — Selecting Pad: UART1_RX_DATA for Mode: ALT8 10 **LCD_DATA08_ALT1** — Selecting Pad: LCD_DATA08 for Mode: ALT1 11 **SD1_CLK_ALT3** — Selecting Pad: SD1_CLK for Mode: ALT3

30.5.387 SPDIF_EXT_CLK_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_EXT_CLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 61Ch offset = 20E_061Ch



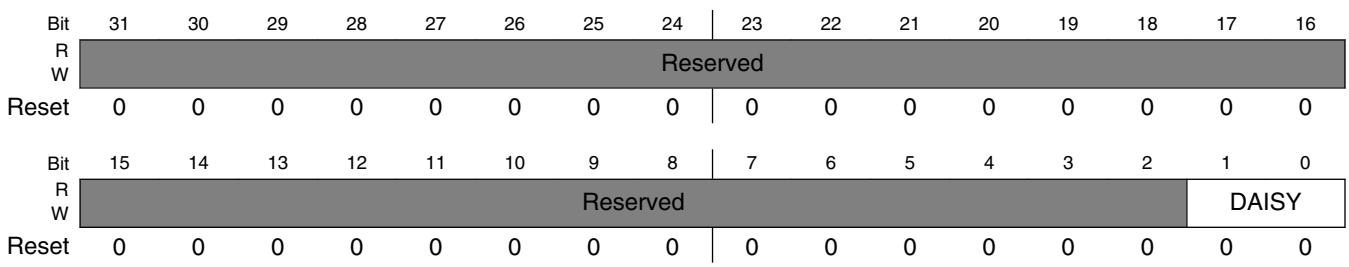
IOMUXC_SPDIF_EXT_CLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: spdif, In Pin: tx_clk2 0 **LCD_DATA07_ALT4** — Selecting Pad: LCD_DATA07 for Mode: ALT4 1 **NAND_DQS_ALT8** — Selecting Pad: NAND_DQS for Mode: ALT8

30.5.388 UART1_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART1_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 620h offset = 20E_0620h



IOMUXC_UART1_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart1, In Pin: uart_rts_b 00 **GPIO1_IO06_ALT8** — Selecting Pad: GPIO1_IO06 for Mode: ALT8 01 **GPIO1_IO07_ALT8** — Selecting Pad: GPIO1_IO07 for Mode: ALT8 10 **UART1_CTS_B_ALT0** — Selecting Pad: UART1_CTS_B for Mode: ALT0 11 **UART1_RTS_B_ALT0** — Selecting Pad: UART1_RTS_B for Mode: ALT0

30.5.389 UART1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART1_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 624h offset = 20E_0624h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART1_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart1, In Pin: uart_RX_DATA 00 **GPIO1_IO02_ALT8** — Selecting Pad: GPIO1_IO02 for Mode: ALT8 01 **GPIO1_IO03_ALT8** — Selecting Pad: GPIO1_IO03 for Mode: ALT8 10 **UART1_TX_DATA_ALT0** — Selecting Pad: UART1_TX_DATA for Mode: ALT0 11 **UART1_RX_DATA_ALT0** — Selecting Pad: UART1_RX_DATA for Mode: ALT0

30.5.390 UART2_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART2_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 628h offset = 20E_0628h

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

IOMUXC_UART2_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart2, In Pin: uart_rts_b 000 **UART2_CTS_B_ALT0** — Selecting Pad: UART2_CTS_B for Mode: ALT0 001 **UART2_RTS_B_ALT0** — Selecting Pad: UART2_RTS_B for Mode: ALT0 010 **UART3_TX_DATA_ALT4** — Selecting Pad: UART3_TX_DATA for Mode: ALT4 011 **UART3_RX_DATA_ALT4** — Selecting Pad: UART3_RX_DATA for Mode: ALT4 100 **NAND_DATA06_ALT8** — Selecting Pad: NAND_DATA06 for Mode: ALT8 101 **NAND_DATA07_ALT8** — Selecting Pad: NAND_DATA07 for Mode: ALT8

30.5.391 UART2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART2_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 62Ch offset = 20E_062Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART2_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart2, In Pin: uart_RX_DATA 00 **UART2_TX_DATA_ALT0** — Selecting Pad: UART2_TX_DATA for Mode: ALT0 01 **UART2_RX_DATA_ALT0** — Selecting Pad: UART2_RX_DATA for Mode: ALT0 10 **NAND_DATA04_ALT8** — Selecting Pad: NAND_DATA04 for Mode: ALT8 11 **NAND_DATA05_ALT8** — Selecting Pad: NAND_DATA05 for Mode: ALT8

30.5.392 UART3_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART3_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 630h offset = 20E_0630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART3_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart3, In Pin: uart_rts_b 00 **UART3_CTS_B_ALT0** — Selecting Pad: UART3_CTS_B for Mode: ALT0 01 **UART3_RTS_B_ALT0** — Selecting Pad: UART3_RTS_B for Mode: ALT0 10 **NAND_CE1_B_ALT8** — Selecting Pad: NAND_CE1_B for Mode: ALT8 11 **NAND_CLE_ALT8** — Selecting Pad: NAND_CLE for Mode: ALT8

30.5.393 UART3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART3_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 634h offset = 20E_0634h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART3_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart3, In Pin: uart_RX_DATA 00 **UART3_TX_DATA_ALT0** — Selecting Pad: UART3_TX_DATA for Mode: ALT0 01 **UART3_RX_DATA_ALT0** — Selecting Pad: UART3_RX_DATA for Mode: ALT0 10 **NAND_READY_B_ALT8** — Selecting Pad: NAND_READY_B for Mode: ALT8 11 **NAND_CE0_B_ALT8** — Selecting Pad: NAND_CE0_B for Mode: ALT8

30.5.394 UART4_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART4_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 638h offset = 20E_0638h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART4_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart4, In Pin: uart_rts_b 00 **ENET1_RX_DATA0_ALT1** — Selecting Pad: ENET1_RX_DATA0 for Mode: ALT1 01 **ENET1_RX_DATA1_ALT1** — Selecting Pad: ENET1_RX_DATA1 for Mode: ALT1 10 **LCD_HSYNC_ALT2** — Selecting Pad: LCD_HSYNC for Mode: ALT2 11 **LCD_VSYNC_ALT2** — Selecting Pad: LCD_VSYNC for Mode: ALT2

30.5.395 UART4_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART4_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 63Ch offset = 20E_063Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART4_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart4, In Pin: uart_RX_DATA 00 **UART4_TX_DATA_ALT0** — Selecting Pad: UART4_TX_DATA for Mode: ALT0 01 **UART4_RX_DATA_ALT0** — Selecting Pad: UART4_RX_DATA for Mode: ALT0 10 **LCD_CLK_ALT2** — Selecting Pad: LCD_CLK for Mode: ALT2 11 **LCD_ENABLE_ALT2** — Selecting Pad: LCD_ENABLE for Mode: ALT2

30.5.396 UART5_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART5_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 640h offset = 20E_0640h

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

IOMUXC_UART5_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart5, In Pin: uart_rts_b 000 **CSI_DATA03_ALT8** — Selecting Pad: CSI_DATA03 for Mode: ALT8 001 **GPIO1_IO08_ALT8** — Selecting Pad: GPIO1_IO08 for Mode: ALT8 010 **GPIO1_IO09_ALT8** — Selecting Pad: GPIO1_IO09 for Mode: ALT8 011 **ENET1_RX_EN_ALT1** — Selecting Pad: ENET1_RX_EN for Mode: ALT1 100 **ENET1_TX_DATA0_ALT1** — Selecting Pad: ENET1_TX_DATA0 for Mode: ALT1 101 **CSI_DATA02_ALT8** — Selecting Pad: CSI_DATA02 for Mode: ALT8

30.5.397 UART5_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART5_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 644h offset = 20E_0644h

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

IOMUXC_UART5_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart5, In Pin: uart_RX_DATA

Table continues on the next page...

IOMUXC_UART5_RX_DATA_SELECT_INPUT field descriptions (continued)

Field	Description
	000 **CSI_DATA00_ALT8** — Selecting Pad: CSI_DATA00 for Mode: ALT8 001 **CSI_DATA01_ALT8** — Selecting Pad: CSI_DATA01 for Mode: ALT8 010 **GPIO1_IO04_ALT8** — Selecting Pad: GPIO1_IO04 for Mode: ALT8 011 **GPIO1_IO05_ALT8** — Selecting Pad: GPIO1_IO05 for Mode: ALT8 100 **UART5_TX_DATA_ALT0** — Selecting Pad: UART5_TX_DATA for Mode: ALT0 101 **UART5_RX_DATA_ALT0** — Selecting Pad: UART5_RX_DATA for Mode: ALT0

30.5.398 UART6_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART6_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 648h offset = 20E_0648h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART6_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart6, In Pin: uart_rts_b 00 **CSI_VSYNC_ALT8** — Selecting Pad: CSI_VSYNC for Mode: ALT8 01 **CSI_HSYNC_ALT8** — Selecting Pad: CSI_HSYNC for Mode: ALT8 10 **ENET1_TX_DATA1_ALT1** — Selecting Pad: ENET1_TX_DATA1 for Mode: ALT1 11 **ENET1_TX_EN_ALT1** — Selecting Pad: ENET1_TX_EN for Mode: ALT1

30.5.399 UART6_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART6_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 64Ch offset = 20E_064Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART6_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart6, In Pin: uart_RX_DATA 00 **CSI_MCLK_ALT8** — Selecting Pad: CSI_MCLK for Mode: ALT8 01 **ENET2_RX_DATA0_ALT1** — Selecting Pad: ENET2_RX_DATA0 for Mode: ALT1 10 **ENET2_RX_DATA1_ALT1** — Selecting Pad: ENET2_RX_DATA1 for Mode: ALT1 11 **CSI_PIXCLK_ALT8** — Selecting Pad: CSI_PIXCLK for Mode: ALT8

30.5.400 UART7_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART7_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 650h offset = 20E_0650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART7_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart7, In Pin: uart_rts_b 00 **ENET1_TX_CLK_ALT1** — Selecting Pad: ENET1_TX_CLK for Mode: ALT1 01 **ENET1_RX_ER_ALT1** — Selecting Pad: ENET1_RX_ER for Mode: ALT1 10 **LCD_DATA06_ALT1** — Selecting Pad: LCD_DATA06 for Mode: ALT1 11 **LCD_DATA07_ALT1** — Selecting Pad: LCD_DATA07 for Mode: ALT1

30.5.401 UART7_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART7_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 654h offset = 20E_0654h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART7_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart7, In Pin: uart_RX_DATA 00 **ENET2_RX_EN_ALT1** — Selecting Pad: ENET2_RX_EN for Mode: ALT1 01 **ENET2_TX_DATA0_ALT1** — Selecting Pad: ENET2_TX_DATA0 for Mode: ALT1 10 **LCD_DATA16_ALT1** — Selecting Pad: LCD_DATA16 for Mode: ALT1 11 **LCD_DATA17_ALT1** — Selecting Pad: LCD_DATA17 for Mode: ALT1

30.5.402 UART8_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART8_RTS_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 658h offset = 20E_0658h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART8_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart8, In Pin: uart_rts_b 00 **ENET2_TX_CLK_ALT1** — Selecting Pad: ENET2_TX_CLK for Mode: ALT1 01 **ENET2_RX_ER_ALT1** — Selecting Pad: ENET2_RX_ER for Mode: ALT1 10 **LCD_DATA04_ALT1** — Selecting Pad: LCD_DATA04 for Mode: ALT1 11 **LCD_DATA05_ALT1** — Selecting Pad: LCD_DATA05 for Mode: ALT1

30.5.403 UART8_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART8_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 65Ch offset = 20E_065Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_UART8_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart8, In Pin: uart_RX_DATA 00 **ENET2_TX_DATA1_ALT1** — Selecting Pad: ENET2_TX_DATA1 for Mode: ALT1 01 **ENET2_TX_EN_ALT1** — Selecting Pad: ENET2_TX_EN for Mode: ALT1 10 **LCD_DATA20_ALT1** — Selecting Pad: LCD_DATA20 for Mode: ALT1 11 **LCD_DATA21_ALT1** — Selecting Pad: LCD_DATA21 for Mode: ALT1

30.5.404 USB_OTG2_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_OC_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 660h offset = 20E_0660h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USB_OTG2_OC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usb, In Pin: otg2_oc 00 **GPIO1_IO03_ALT2** — Selecting Pad: GPIO1_IO03 for Mode: ALT2 01 **ENET2_TX_EN_ALT8** — Selecting Pad: ENET2_TX_EN for Mode: ALT8 10 **SD1_DATA2_ALT8** — Selecting Pad: SD1_DATA2 for Mode: ALT8

30.5.405 USB_OTG_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG_OC_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 664h offset = 20E_0664h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USB_OTG_OC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usb, In Pin: otg_oc 00 **GPIO1_IO01_ALT2** — Selecting Pad: GPIO1_IO01 for Mode: ALT2 01 **ENET2_RX_DATA1_ALT8** — Selecting Pad: ENET2_RX_DATA1 for Mode: ALT8 10 **SD1_CLK_ALT8** — Selecting Pad: SD1_CLK for Mode: ALT8

30.5.406 USDHC1_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_CD_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 668h offset = 20E_0668h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC1_CD_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc1, In Pin: card_det 00 **GPIO1_IO03_ALT4** — Selecting Pad: GPIO1_IO03 for Mode: ALT4 01 **UART1_RTS_B_ALT2** — Selecting Pad: UART1_RTS_B for Mode: ALT2 10 **CSI_DATA05_ALT8** — Selecting Pad: CSI_DATA05 for Mode: ALT8

30.5.407 USDHC1_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_WP_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 66Ch offset = 20E_066Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC1_WP_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc1, In Pin: wp_on 00 **GPIO1_IO02_ALT4** — Selecting Pad: GPIO1_IO02 for Mode: ALT4 01 **UART1_CTS_B_ALT2** — Selecting Pad: UART1_CTS_B for Mode: ALT2 10 **CSI_DATA04_ALT8** — Selecting Pad: CSI_DATA04 for Mode: ALT8

30.5.408 USDHC2_CLK_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CLK_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 670h offset = 20E_0670h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_CLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: CLK_in 00 **CSI_VSYNC_ALT1** — Selecting Pad: CSI_VSYNC for Mode: ALT1 01 **LCD_DATA19_ALT8** — Selecting Pad: LCD_DATA19 for Mode: ALT8 10 **NAND_RE_B_ALT1** — Selecting Pad: NAND_RE_B for Mode: ALT1

30.5.409 USDHC2_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CD_B_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 674h offset = 20E_0674h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_CD_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: card_det 00 **CSI_MCLK_ALT1** — Selecting Pad: CSI_MCLK for Mode: ALT1 01 **GPIO1_IO07_ALT4** — Selecting Pad: GPIO1_IO07 for Mode: ALT4 10 **UART1_RTS_B_ALT8** — Selecting Pad: UART1_RTS_B for Mode: ALT8

30.5.410 USDHC2_CMD_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CMD_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 678h offset = 20E_0678h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_CMD_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: cmd_in 00 **CSI_HSYNC_ALT1** — Selecting Pad: CSI_HSYNC for Mode: ALT1 01 **LCD_DATA18_ALT8** — Selecting Pad: LCD_DATA18 for Mode: ALT8 10 **NAND_WE_B_ALT1** — Selecting Pad: NAND_WE_B for Mode: ALT1

30.5.411 USDHC2_DATA0_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA0_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 67Ch offset = 20E_067Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA0_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA0_in 00 **CSI_DATA00_ALT1** — Selecting Pad: CSI_DATA00 for Mode: ALT1 01 **LCD_DATA20_ALT8** — Selecting Pad: LCD_DATA20 for Mode: ALT8 10 **NAND_DATA00_ALT1** — Selecting Pad: NAND_DATA00 for Mode: ALT1

30.5.412 USDHC2_DATA1_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA1_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 680h offset = 20E_0680h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA1_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA1_in 00 **CSI_DATA01_ALT1** — Selecting Pad: CSI_DATA01 for Mode: ALT1 01 **LCD_DATA21_ALT8** — Selecting Pad: LCD_DATA21 for Mode: ALT8 10 **NAND_DATA01_ALT1** — Selecting Pad: NAND_DATA01 for Mode: ALT1

30.5.413 USDHC2_DATA2_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA2_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 684h offset = 20E_0684h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA2_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA2_in 00 **LCD_DATA22_ALT8** — Selecting Pad: LCD_DATA22 for Mode: ALT8 01 **NAND_DATA02_ALT1** — Selecting Pad: NAND_DATA02 for Mode: ALT1 10 **CSI_DATA02_ALT1** — Selecting Pad: CSI_DATA02 for Mode: ALT1

30.5.414 USDHC2_DATA3_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA3_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 688h offset = 20E_0688h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA3_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA3_in 00 **CSI_DATA03_ALT1** — Selecting Pad: CSI_DATA03 for Mode: ALT1 01 **LCD_DATA23_ALT8** — Selecting Pad: LCD_DATA23 for Mode: ALT8 10 **NAND_DATA03_ALT1** — Selecting Pad: NAND_DATA03 for Mode: ALT1

30.5.415 USDHC2_DATA4_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA4_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 68Ch offset = 20E_068Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA4_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA4_in 00 **LCD_DATA14_ALT8** — Selecting Pad: LCD_DATA14 for Mode: ALT8 01 **NAND_DATA04_ALT1** — Selecting Pad: NAND_DATA04 for Mode: ALT1 10 **CSI_DATA04_ALT1** — Selecting Pad: CSI_DATA04 for Mode: ALT1

30.5.416 USDHC2_DATA5_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA5_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 690h offset = 20E_0690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA5_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA5_in 00 **LCD_DATA15_ALT8** — Selecting Pad: LCD_DATA15 for Mode: ALT8 01 **NAND_DATA05_ALT1** — Selecting Pad: NAND_DATA05 for Mode: ALT1 10 **CSI_DATA05_ALT1** — Selecting Pad: CSI_DATA05 for Mode: ALT1

30.5.417 USDHC2_DATA6_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA6_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 694h offset = 20E_0694h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA6_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA6_in 00 **LCD_DATA16_ALT8** — Selecting Pad: LCD_DATA16 for Mode: ALT8 01 **NAND_DATA06_ALT1** — Selecting Pad: NAND_DATA06 for Mode: ALT1 10 **CSI_DATA06_ALT1** — Selecting Pad: CSI_DATA06 for Mode: ALT1

30.5.418 USDHC2_DATA7_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA7_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 698h offset = 20E_0698h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_DATA7_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: DATA7_in 00 **LCD_DATA17_ALT8** — Selecting Pad: LCD_DATA17 for Mode: ALT8 01 **NAND_DATA07_ALT1** — Selecting Pad: NAND_DATA07 for Mode: ALT1 10 **CSI_DATA07_ALT1** — Selecting Pad: CSI_DATA07 for Mode: ALT1

30.5.419 USDHC2_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_WP_SELECT_INPUT)

DAISY Register

Address: 20E_0000h base + 69Ch offset = 20E_069Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

IOMUXC_USDHC2_WP_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: wp_on 00 **GPIO1_IO06_ALT4** — Selecting Pad: GPIO1_IO06 for Mode: ALT4 01 **UART1_CTS_B_ALT8** — Selecting Pad: UART1_CTS_B for Mode: ALT8 10 **CSI_PIXCLK_ALT1** — Selecting Pad: CSI_PIXCLK for Mode: ALT1

Chapter 31
Keypad Port (KPP)

31.1 Overview

The Keypad Port (KPP) is a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O).

The figure below shows the KPP block diagram. The KPP provides interface for the keypad matrix with 2-point contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously on the keypad.

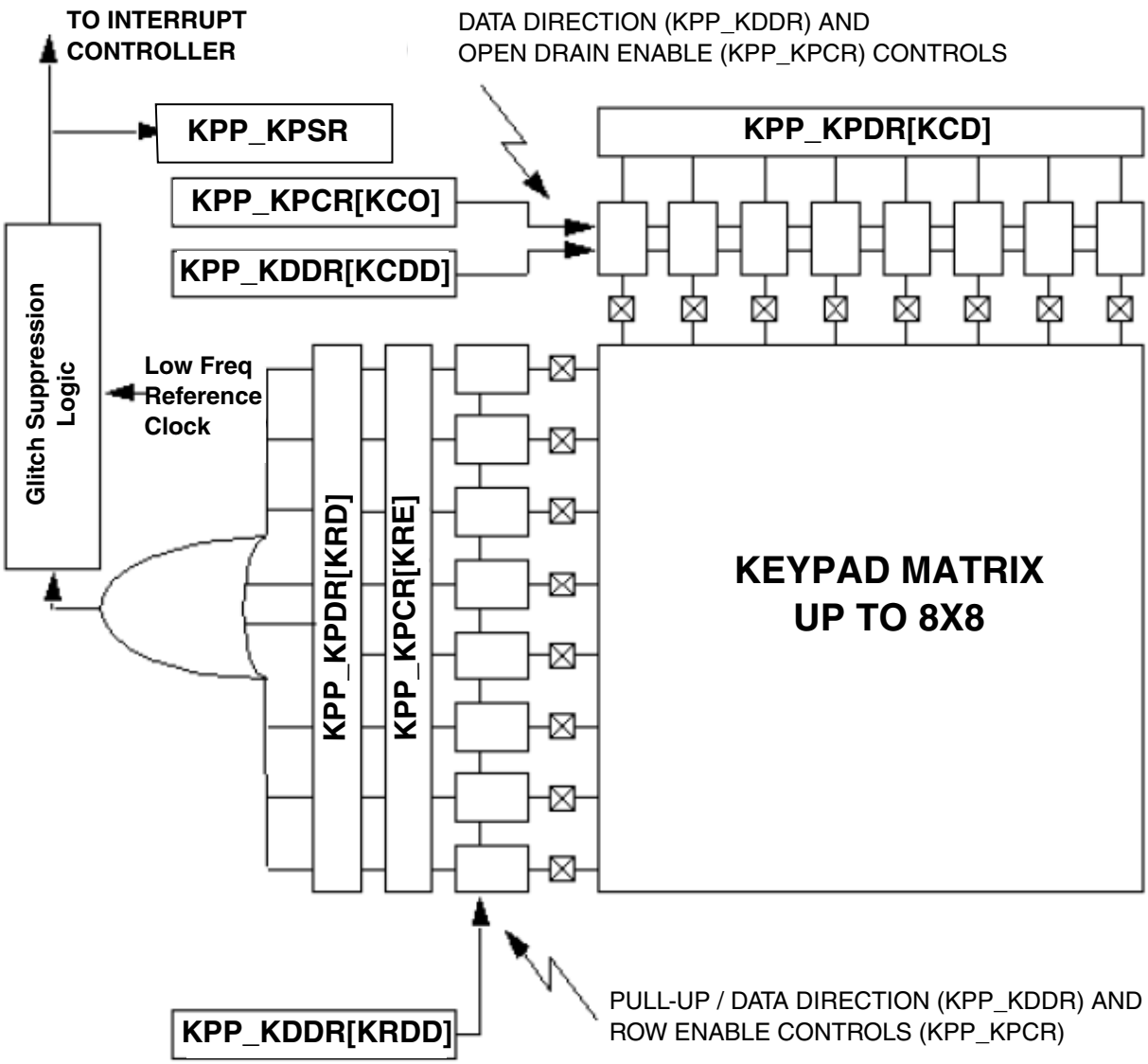


Figure 31-1. KPP Peripheral Block Diagram

31.1.1 Features

The KPP includes these distinctive features:

- Supports up to an 8 x 8 external key pad matrix
- Port pins can be used as general purpose I/O
- Open drain design
- Glitch suppression circuit design
- Multiple-key detection
- Long key-press detection
- Standby key-press detection
- Synchronizer chain clear
- Supports a 2-point and 3-point contact key matrix

31.1.2 Modes and Operations

This block supports the following modes:

- Run Mode-This is the normal functional mode in which the KPP can detect any key press event.
- Low Power Mode-The keypad can detect any key press even in low power modes (when there is no MCU clock).

31.2 Clocks

The table found here describes the clock sources for KPP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 31-1. KPP Clocks

Clock name	Clock Root	Description
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32 kHz)
ipg_clk_s	ipg_clk_root	Peripheral access clock

31.3 External Signals

There are several pins dedicated to the KPP. Keypads of any configuration up to eight rows and eight columns are supported through the software configuration of the peripheral pins. Any pins not used for the keypad are available as general purpose I/O. The registers are configured such that the pins can be treated as an I/O port up to 16 bits wide.

See the table below for the list of external signals.

Table 31-2. KPP External Signals

Signal	Description	Pad	Mode	Direction
KPP_COL0	Column input or output pin, from chip	ENET1_RX_DATA1	ALT6	IO
		NAND_WE_B	ALT3	
KPP_COL1	Column input or output pin, from chip	ENET1_TX_DATA0	ALT6	IO
		NAND_DATA01	ALT3	
KPP_COL2	Column input or output pin, from chip	ENET1_TX_EN	ALT6	IO
		NAND_DATA03	ALT3	
KPP_COL3	Column input or output pin, from chip	ENET1_RX_ER	ALT6	IO
KPP_COL4	Column input or output pin, from chip	ENET2_RX_DATA1	ALT6	IO
KPP_COL5	Column input or output pin, from chip	ENET2_TX_DATA0	ALT6	IO
KPP_COL6	Column input or output pin, from chip	ENET2_TX_EN	ALT6	IO
KPP_COL7	Column input or output pin, from chip	ENET2_RX_ER	ALT6	IO
KPP_ROW0	Row input or output pin, from chip	ENET1_RX_DATA0	ALT6	IO
		NAND_RE_B	ALT3	
KPP_ROW1	Row input or output pin, from chip	ENET1_RX_EN	ALT6	IO
		NAND_DATA00	ALT3	
KPP_ROW2	Row input or output pin, from chip	ENET1_RX_DATA1	ALT6	IO
		NAND_DATA02	ALT3	
KPP_ROW3	Row input or output pin, from chip	ENET1_TX_CLK	ALT6	IO
KPP_ROW4	Row input or output pin, from chip	ENET2_RX_DATA0	ALT6	IO
KPP_ROW5	Row input or output pin, from chip	ENET2_RX_EN	ALT6	IO
KPP_ROW6	Row input or output pin, from chip	ENET2_TX_DATA1	ALT6	IO
KPP_ROW7	Row input or output pin, from chip	ENET2_TX_CLK	ALT6	IO

31.3.1 Input Pins

Any of the 16 pins associated with the KPP can be configured as inputs by writing a "0" to the appropriate bits in the KPP_KDDR. Additionally, the least significant 8 bits (ROW inputs) corresponding to KPP_KDDR[KRDD] have internal pull-ups, which are enabled when the pin is used as an input.

31.3.2 Output Pins

Any of the 16 pins associated with the KPP can be configured as outputs by writing the appropriate bits in the KPP_KDDR to a "1". Additionally, the 8 most significant bits (15-8) can be designated as open drain outputs by writing a "1" to the appropriate bits in the KPP_KPCR. The lower 8 bits (7-0) are always in "totem pole" style, driven when configured as outputs.

See the table below.

Table 31-3. Keypad Port Column Modes

KPP_KDDR (15:8)	KPP_KPCR (15:8)	Pin Function
0	x	Input
1	0	Totem-Pole Output
1	1	Open-Drain Output

NOTE

Totem pole capability should be provided for column pins. Totem pole configuration helps for a faster discharge of keypad capacitance when all columns need to be quickly brought to a "1" during the scan routine. With this configuration, delay between the scanning of two subsequent columns is reduced.

31.3.3 Generation of Transfer Error Signal on Peripheral Bus

If there is an access to an address which is not implemented, then the KPP asserts a transfer error signal on Peripheral Bus.

31.4 Functional Description

The Keypad Port (KPP) is designed to simplify the software task of scanning a keypad matrix. With appropriate software support and matrix organization, the KPP is capable of detecting, debouncing, and decoding one or more keys pressed simultaneously on the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes provided that a low frequency reference clock is on. The KPP may generate an ARM platform interrupt any time a key press or key release is detected. This interrupt is capable of forcing the processor out of a low power mode.

31.4.1 Keypad Matrix Construction

The KPP is designed to interface to a keypad matrix, which shorts the intersecting row and column lines together whenever a key is depressed. The interface is not optimized for any other switch configuration.

31.4.2 Keypad Port Configuration

The software must initialize the KPP for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip, pull-up resistors should be implemented for active keypad rows.

In addition to enabled row inputs in the Keypad Control register, corresponding interrupt (depress or/and release) must also be enabled to generate an interrupt.

Discrete switches that are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware detects closures of these switches without the need for software polling.

31.4.3 Keypad Matrix Scanning

Keypad scanning is performed by a software loop that walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, with the results of each pass optionally compared to those from the previous pass. When several (3 or 4) consecutive scans yield the same key closures, a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next higher software layer.

The basic debouncing period, which must be defined in the software routine, may be controlled with an internal timer. The basic period is the period between the scan of two consecutive columns, so the debouncing time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period.

31.4.4 Keypad Standby

There is no need for the ARM platform to continually scan the keypad. Between key presses, the keypad can be left in a state that requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point, the ARM platform can attend to other tasks or revert to a low power standby mode. The KPP will interrupt the ARM platform if any key is pressed.

Upon receiving a keypad interrupt, the ARM platform should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used when scanning to prevent a possible DC path between power and ground through two or more switches.

31.4.5 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the ARM platform. The circuit is a 4-state synchronizer clocked from a low frequency reference clock source.

This clock must continue to run in any low power mode where the keypad is a wake-up source, as the ARM platform interrupt is generated from the synchronized input. An interrupt is not generated until all four synchronizer stages have latched a valid key assertion. This guarantees the filtering out of any noise less than three clock periods in duration of a low frequency reference clock. Noise filtering of the duration between three to four clock periods cannot be guaranteed. The interrupt output is latched in an S-R latch and remains asserted until cleared by the software. The Set input of the latch is rising-edge clocked. See the figure below.

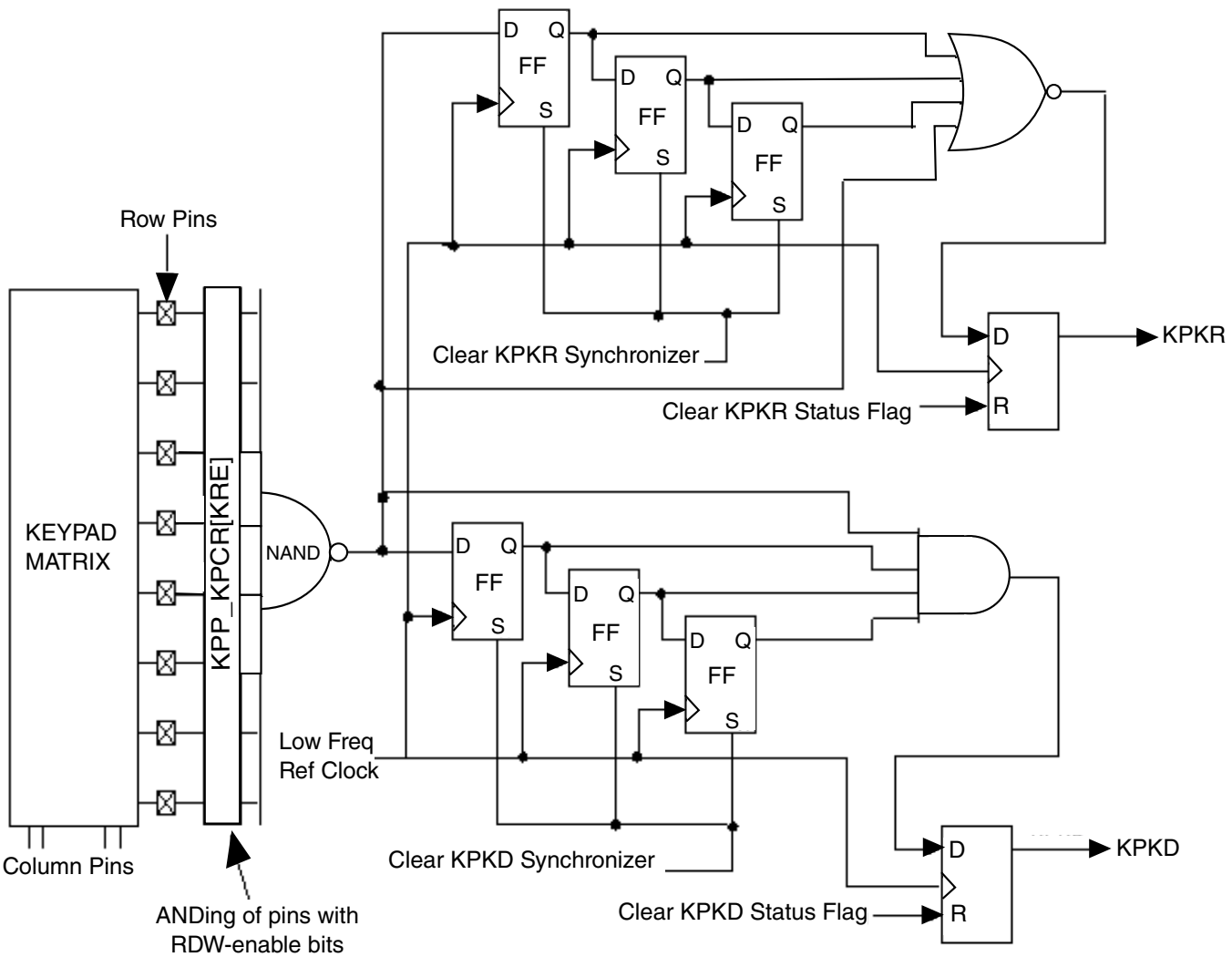


Figure 31-2. Keypad Synchronizer Functional Diagram

31.4.6 Multiple Key Closures

Using the key press and Key release interrupts, the software can detect multiple keys or achieve n key rollover. The key scanning routine can be programmed accordingly.

See the following figures for illustrations of the interfacing of a 2-contact keypad matrix with the KPP controller. With proper enabling of row lines and the performing scan-routine, multiple key presses can be detected. When keys present on the same row are pressed, corresponding row lines (multiple lines) become low when the column is driven low during a scan-routine. By reading the data-register, pressed keys can be detected. Similarly, when keys present on same row line are pressed, the corresponding row line (only one line) becomes low when logic "0" is driven on the column line during a scan-routine.

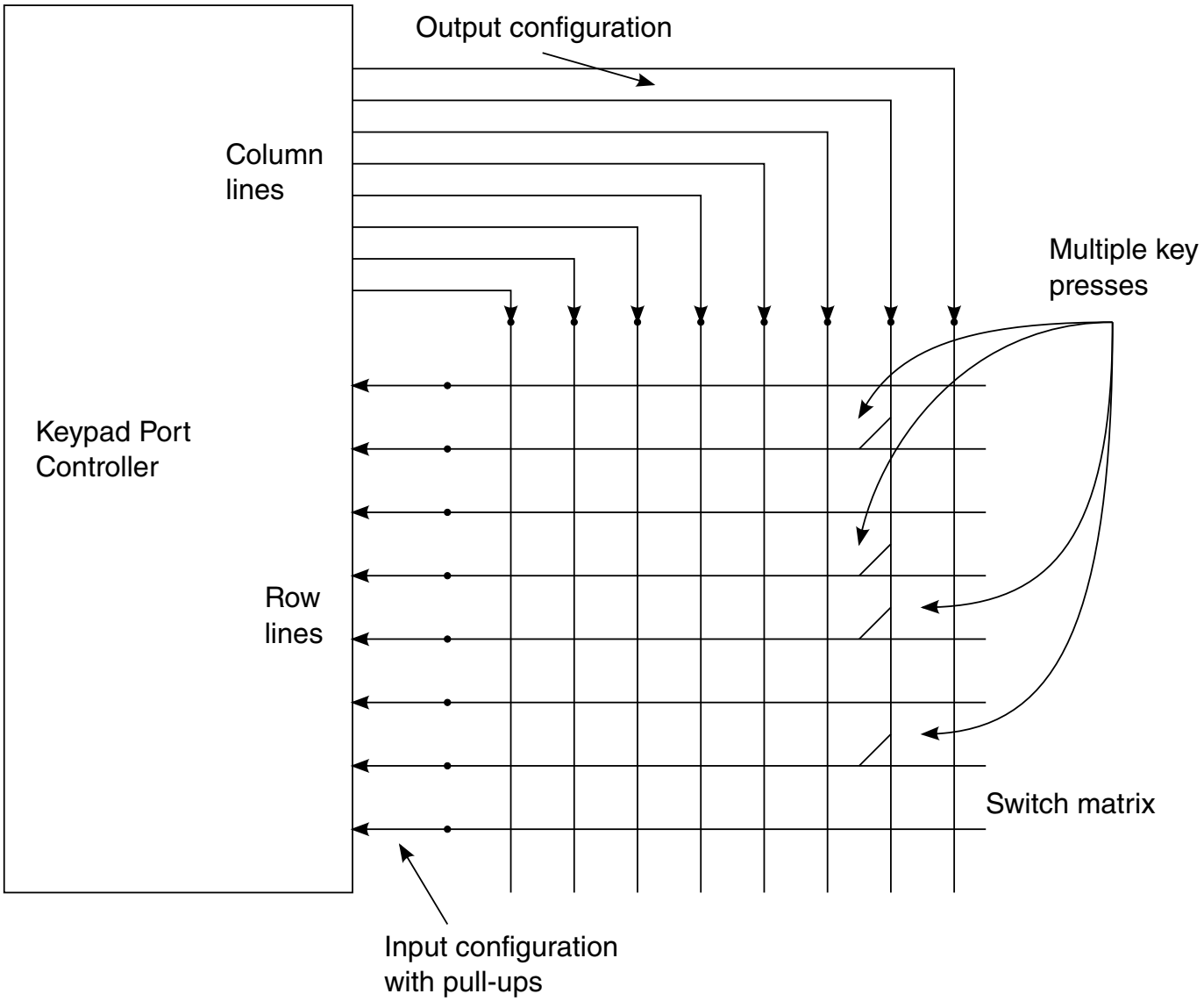


Figure 31-3. Multiple Key Presses on Same Column Line (Simplified View)

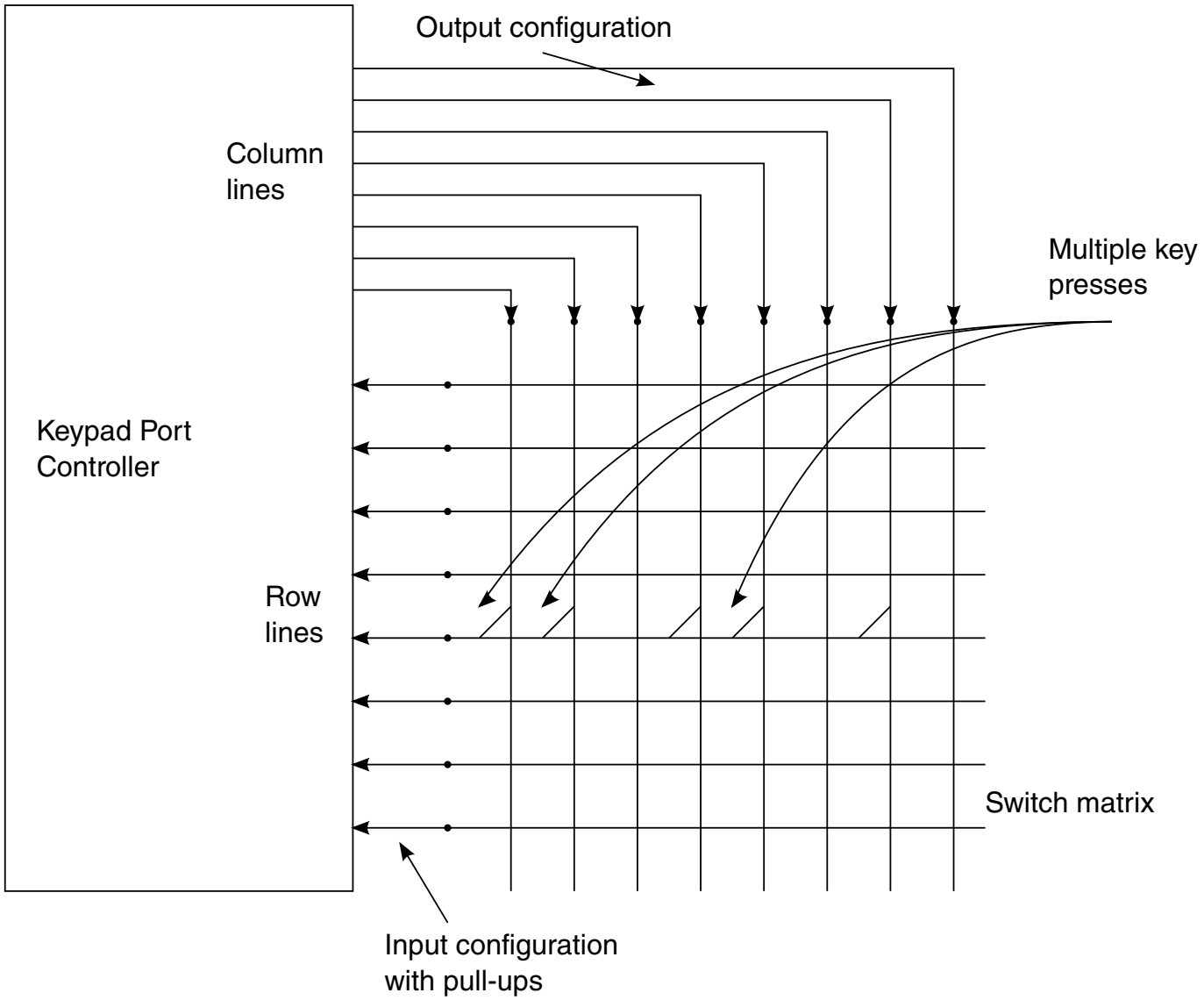


Figure 31-4. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An n key rollover is a technique with which the system can recognize the order in which keys are pressed.

31.4.6.1 Ghost Key Problem and Correction

The KPP detects if one or multiple keys are pressed or released. In the case where a simple keypad matrix with two-contact switches is used, there is a chance of "ghost" key detection when three or more keys are pressed. This is a limitation imposed by such a keypad matrix.

As can be seen in Figure 31-5, three keys pressed simultaneously can cause a short between the column currently "scanned" by the software and another column. Depending on the location of the third key pressed, a "ghost" key press may be detected.

However, this can be corrected by using a keypad matrix that provides "ghost" key protection. Such a matrix implements a one-way "diode" at all keypad points between rows and columns. This way, the multiple pressing of three keys will not cause a short at a fourth key (see Figure 31-6).

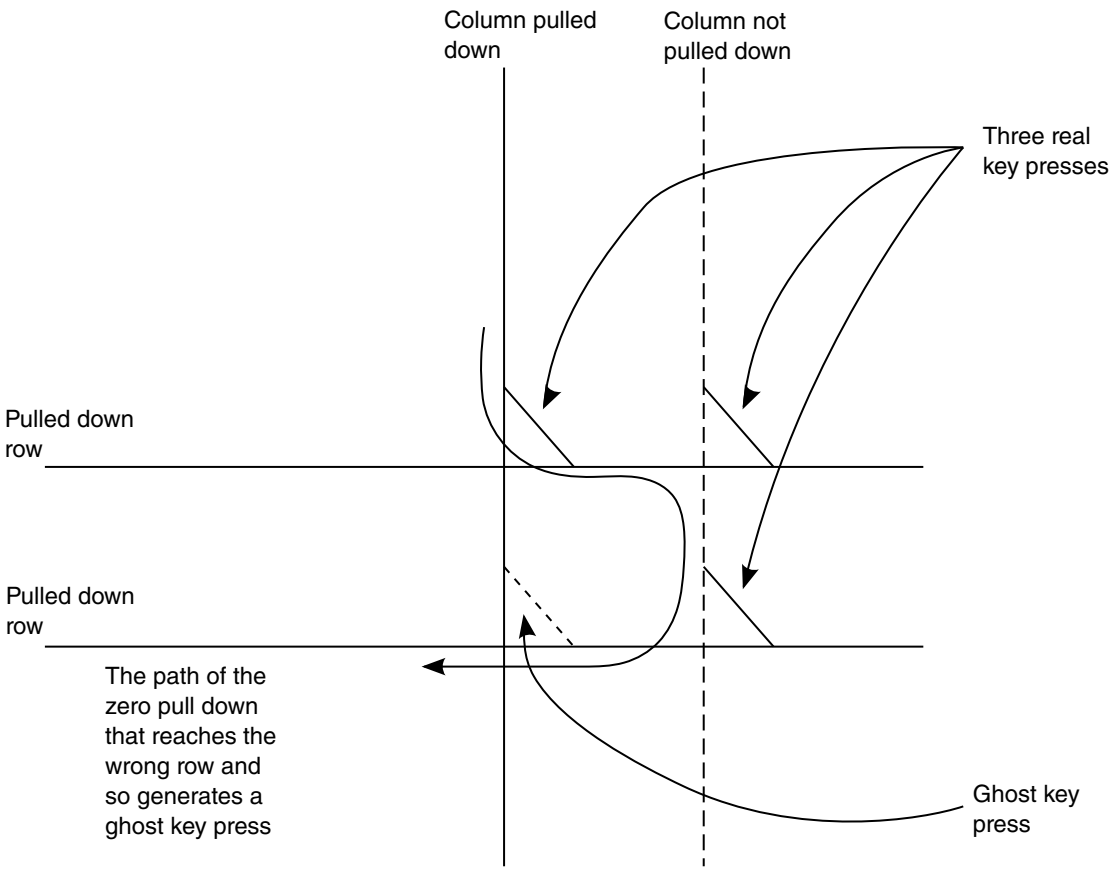


Figure 31-5. Decoding Wrong Three- Key-Presses

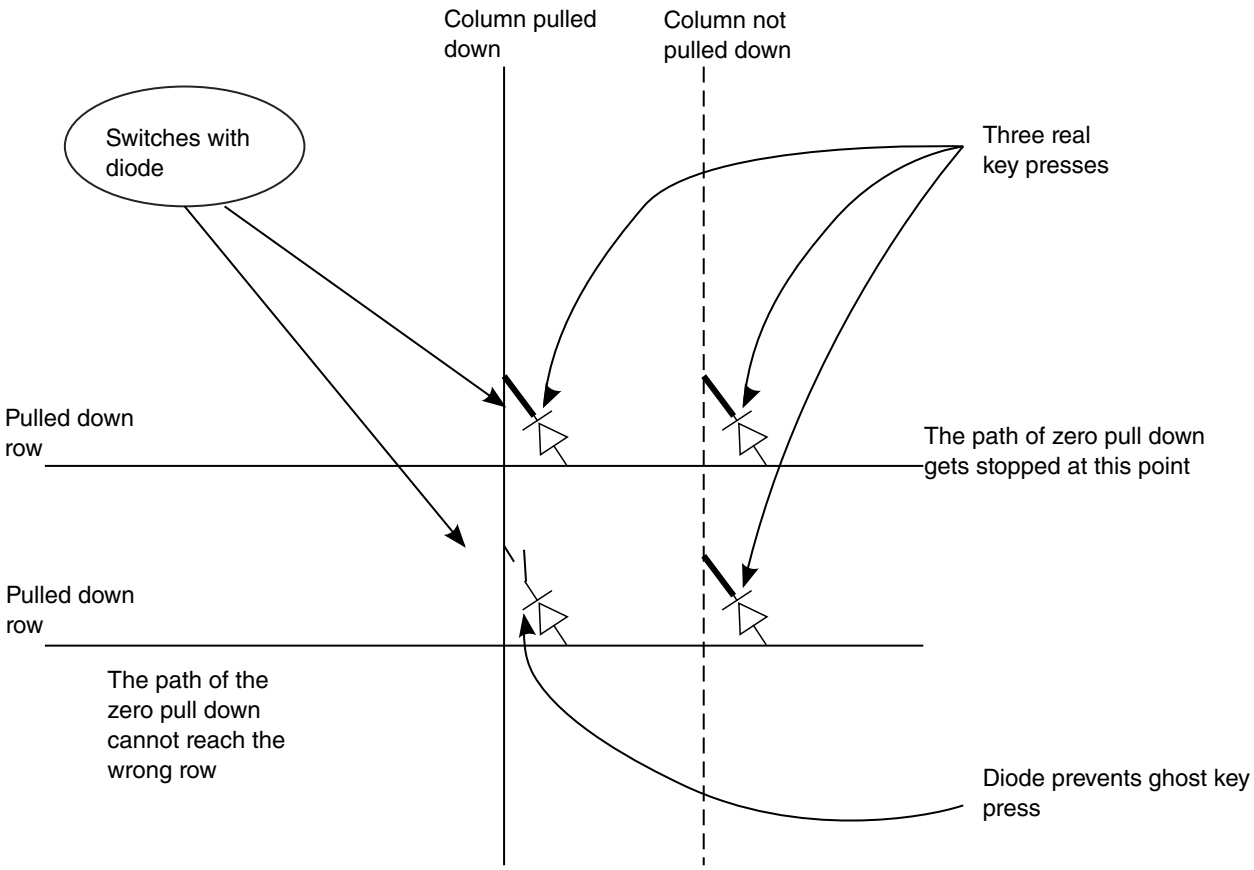


Figure 31-6. Matrix with "Ghost" Key Protections

31.4.7 3-Point Contact Keys Support

The KPP supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 31-7, two points of such a key are connected to keypad lines, while a third point is connected to ground (low logic).

The keypad lines should be configured as input and a pull-up should to be present on these lines. When such a key is pressed, corresponding keypad lines go low and an interrupt is generated. There is no need to perform a scanning routine for identification of pressed key as it can be done by reading the keypad data-register. A limitation with such a matrix is that for every key at least one keypad row line should be used.

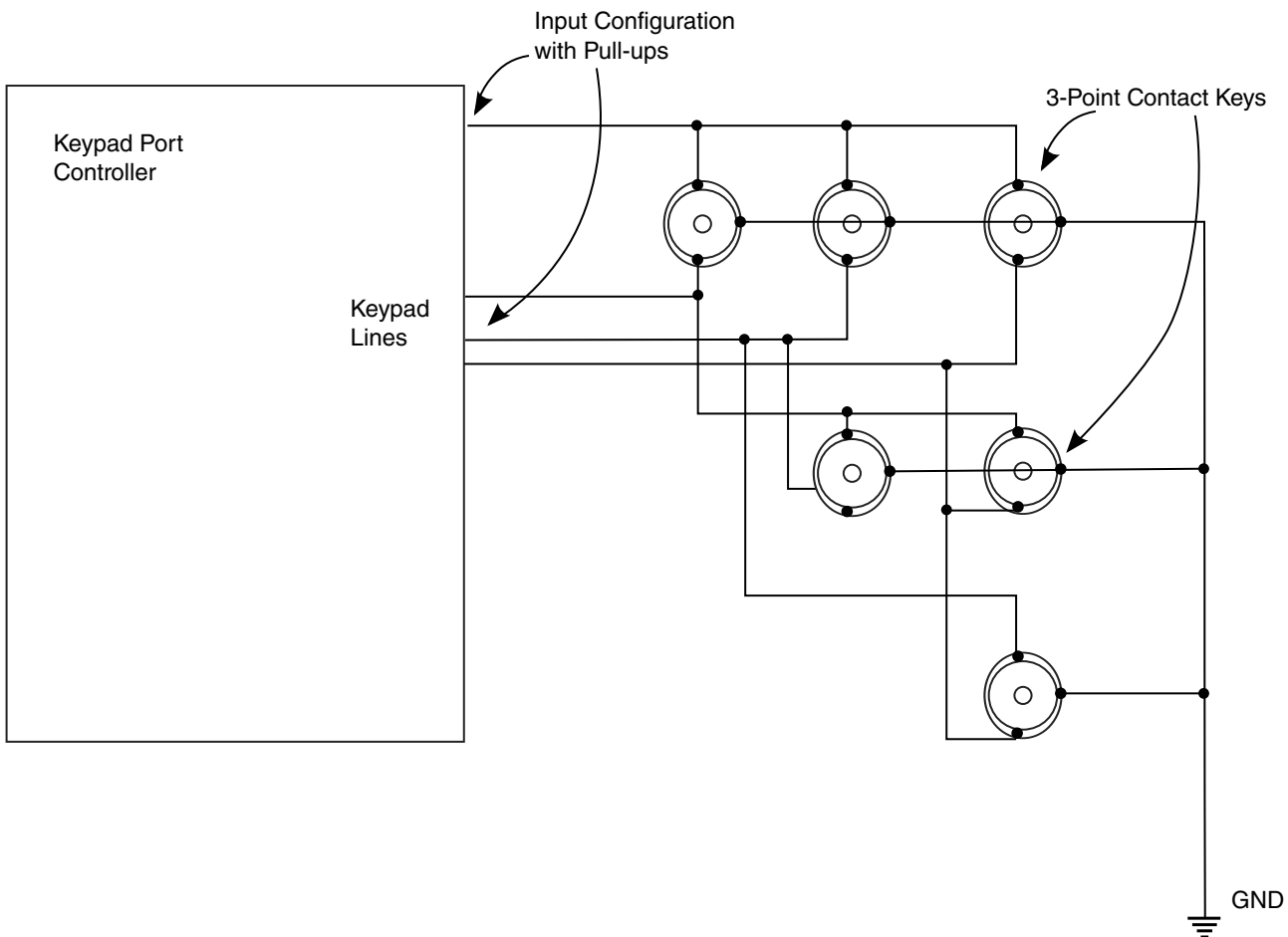


Figure 31-7. KPP Interface with 3-point Contact Key Matrix (Simplified View)

31.5 Initialization/Application Information

31.5.1 Typical Keypad Configuration and Scanning Sequence

Perform the following steps to configure the keypad:

1. Enable the number of rows in the keypad (KPP_KPCR[KRE]).
2. Write 0s to KPP_KPDR[KCD].
3. Configure the keypad columns as open-drain (KPP_KPCR[KCO]).
4. Configure columns as output (KPP_KDDR[KCDD]) and rows as input (KPP_KDDR[KRDD]).
5. Clear the KPKD Status Flag and Synchronizer chain.
6. Set the KDIE control bit, and clear the KRIE control bit (avoid false release events).
7. (The system is now in standby mode, and awaiting a key press.)

31.5.2 Key Press Interrupt Scanning Sequence

Perform the following steps to perform a keypad scanning routine:

1. Disable both (depress and release) keypad interrupts.
2. Write 1s to KPP_KPDR[KCD], setting column data to 1s.
3. Configure columns as totem pole outputs (for quick discharging of keypad capacitance).
4. Configure columns as open-drain.
5. Write a single column to 0, and other columns to 1.
6. Sample row inputs and save data. Multiple key presses can be detected on a single column.
7. Repeat Steps 2-6 for remaining columns.
8. Return all columns to 0 in preparation for standby mode.
9. Clear KPKD and KPKR status bit(s) by writing to a "1"; set the KPKR synchronizer chain by writing a "1" to the KPP_KRSS register; and clear the KPKD synchronizer chain by writing a "1" to the KDSC register.
10. Re-enable the appropriate keypad interrupt(s) so that the KDIE detects a key hold condition, or the KRIE detects a key-release event.

31.5.3 Additional Comments

The order of key press detection can be done in software only. Therefore, the software may need to run the scan routines at very short intervals of time per the application's demands. The reason that such functionality cannot be put in the KPP is that the block is limited by the number of external pins.

For the keys that require a very precise order (such as game keys), individual GPIO pins may be more useful.

31.6 KPP Memory Map/Register Definition

The KPP contains four registers.

KPP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_8000	Keypad Control Register (KPP_KPCR)	16	R/W	0000h	31.6.1/1922
20B_8002	Keypad Status Register (KPP_KPSR)	16	R/W	0400h	31.6.2/1923
20B_8004	Keypad Data Direction Register (KPP_KDDR)	16	R/W	0000h	31.6.3/1925
20B_8006	Keypad Data Register (KPP_KPDR)	16	R/W	0000h	31.6.4/1925

31.6.1 Keypad Control Register (KPP_KPCR)

The Keypad Control Register determines which of the eight possible column strobes are to be open drain when configured as outputs, and which of the eight row sense lines are considered in generating an interrupt to the core.

It is up to the programmer to ensure that pins being used for functions other than the keypad are properly disabled. The KPP_KPCR register is byte- or half-word-addressable.

Address: 20B_8000h base + 0h offset = 20B_8000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCO								KRE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPCR field descriptions

Field	Description
15–8 KCO	Keypad Column Strobe Open-Drain Enable. Setting a column open-drain enable bit (KCO7-KCO0) disables the pull-up driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input. **NOTE:** Configuration of external port control logic (for example, IOMUX) should be done properly so that the KPP controls an open-drain enable of the pin. 0 **TOTEM_POLE** — Column strobe output is totem pole drive. 1 **OPEN_DRAIN** — Column strobe output is open drain.

Table continues on the next page...

KPP_KPCR field descriptions (continued)

Field	Description
KRE	Keypad Row Enable. Setting a row enable control bit in this register enables the corresponding row line to participate in interrupt generation. Likewise, clearing a bit disables that row from being used to generate an interrupt. This register is cleared by a reset, disabling all rows. The row-enable logic is independent of the programmed direction of the pin. Writing a "0" to the data register of the pins configured as outputs will cause a keypad interrupt to be generated if the row enable associated with that bit is set. 0 Row is not included in the keypad key press detect. 1 Row is included in the keypad key press detect.

31.6.2 Keypad Status Register (KPP_KPSR)

The Keypad Status Register reflects the state of the key press detect circuit. The KPP_KPSR register is byte- or half-word-addressable.

Address: 20B_8000h base + 2h offset = 20B_8002h

Bit	15	14	13	12	11	10	9	8
Read	0						KRIE	KDIE
Write								
Reset	0	0	0	0	0	1	0	0

Bit	7	6	5	4	3	2	1	0
Read	0				0	0	KPKR	KPKD
Write					KRSS	KDSC	w1c	w1c
Reset	0	0	0	0	0	0	0	0

KPP_KPSR field descriptions

Field	Description
15–10 Reserved	This read-only field is reserved and always has the value 0.
9 KRIE	Keypad Release Interrupt Enable. The software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKR is set. 1 An interrupt request is generated when KPKR is set.
8 KDIE	Keypad Key Depress Interrupt Enable. Software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice-versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain.

Table continues on the next page...

KPP_KPSR field descriptions (continued)

Field	Description
	0 No interrupt request is generated when KPKD is set. 1 An interrupt request is generated when KPKD is set.
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 KRSS	Key Release Synchronizer Set. Self-clear bit. The Key release synchronizer is set by writing a logic one into this bit. Reads return a value of "0". 0 No effect 1 Set bits which sets keypad release synchronizer chain
2 KDSC	Key Depress Synchronizer Clear. Self-clear bit. The Key depress synchronizer is cleared by writing a logic "1" into this bit. Reads return a value of "0". 0 No effect 1 Set bits that clear the keypad depress synchronizer chain
1 KPKR	Keypad Key Release. The keypad key release (KPKR) status bit is set when all enabled rows are detected high after synchronization (the KPKR status bit will be set when cleared by a reset). The KPKR bit may be used to generate a maskable key release interrupt. The key release synchronizer may be set high by software after scanning the keypad to ensure a known state. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by software prior to the system exiting the state it represents. Reset value of register is "0" as long as reset is asserted. However when reset is de-asserted, the value of the register depends upon the external row pins and can become "1". 0 No key release detected 1 All keys have been released
0 KPKD	Keypad Key Depress. The keypad key depress (KPKD) status bit is set when one or more enabled rows are detected low after synchronization. The KPKD status bit remains set until cleared by the software. The KPKD bit may be used to generate a maskable key depress interrupt. If desired, the software may clear the key press synchronizer chain to allow a repeated interrupt to be generated while a key remains pressed. In this case, a new interrupt will be generated after the synchronizer delay (4 cycles of the low frequency reference clock elapses if a key remains pressed. This functionality can be used to detect a long key press. This allows detection of additional key presses of the same key or other keys. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by the software prior to the system exiting the state it represents. 0 No key presses detected 1 A key has been depressed

31.6.3 Keypad Data Direction Register (KPP_KDDR)

The bits in the KPP_KDDR control the direction of the keypad port pins. The upper eight bits in the register affect the pins designated as column strobes, while the lower eight bits affect the row sense pins. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding port pin as an input. For the Keypad Row DDR, an internal pull-up is enabled if the corresponding bit is clear. This register is cleared by a reset, configuring all pins as inputs. The KPP_KDDR register is byte- or half-word addressable.

NOTE

When a pin is used as row pin for keypad purposes, all corresponding pull-ups should be enabled at the upper level (for example, IOMUX) when the bit in KRDD is cleared.

Address: 20B_8000h base + 4h offset = 20B_8004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCDD								KRDD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KDDR field descriptions

Field	Description
15–8 KCDD	Keypad Column Data Direction Register. Setting a bit configures the corresponding COL*n* pin as an output (where *n* = 7 through 0). 0 **INPUT** — COLn pin is configured as an input. 1 **OUTPUT** — COLn pin is configured as an output.
KRDD	Keypad Row Data Direction. Setting a bit configures the corresponding ROW*n* pin as an output (where *n* = 7 through 0). 0 **INPUT** — ROWn pin configured as an input. 1 **OUTPUT** — ROWn pin configured as an output.

31.6.4 Keypad Data Register (KPP_KPDR)

This 16-bit register is used to access the column and row data. Data written to this register is stored in an internal latch, and for each pin configured as an output, the stored data is driven onto the pin. A read of this register returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register.

The KPP_KPDR register is byte- or half-word addressable. This register is not initialized by a reset. Valid data should be written to this register before any bits are configured as outputs.

Address: 20B_8000h base + 6h offset = 20B_8006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCD								KRD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPDR field descriptions

Field	Description
15–8 KCD	Keypad Column Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to column ports 1 Read/Write "1" from/to column ports
KRD	Keypad Row Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to row ports 1 Read/Write "1" from/to row ports

Chapter 32
Enhanced LCD Interface (eLCDIF)

32.1 Overview

The eLCDIF is a general purpose display controller used to drive a wide range of display devices varying in size and capability.

Many of these displays have had an asynchronous parallel MPU interface for command and data transfer to an integrated frame buffer. There are other popular displays that support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) or the VSYNC mode for high-speed data transfers. In addition to these displays, it is also common to provide support for digital video encoders that accept ITU-R BT.656 format 4:2:2 YCbCr digital component video and convert it to analog TV signals. The eLCDIF block supports these interfaces by providing fully programmable functionality.

The block has several major features:

- Bus master interface to source frame buffer data for display refresh. This interface can also be used to drive data for "Smart" displays.
- PIO interface to manage data transfers between "Smart" displays and SoC.
- 8/16/18/24 bit LCD data bus support available depending on I/O mux options.
- Programmable timing and parameters for MPU, VSYNC and DOTCLK LCD interfaces to support a wide variety of displays.
- ITU-R BT.656 mode (called Digital Video Interface or DVI mode here) including progressive-to-interlace feature and RGB to YCbCr 4:2:2 color space conversion to support 525/60 and 625/50 operation.

32.2 External Signals

The following table describes the external signals of LCD:

Table 32-1. LCD External Signals

Signal	Description	Pad	Mode	Direction
LCD_BUSY	Busy Signal	LCD_VSYNC	ALT1	I
LCD_CLK	Clock signal	LCD_CLK	ALT0	I
LCD_CS	Chip select	LCD_RESET	ALT1	O
LCD_DATA0	Data signals	LCD_DATA00	ALT0	IO
LCD_DATA1		LCD_DATA01	ALT0	
LCD_DATA2		LCD_DATA02	ALT0	
LCD_DATA3		LCD_DATA03	ALT0	
LCD_DATA4		LCD_DATA04	ALT0	
LCD_DATA5		LCD_DATA05	ALT0	
LCD_DATA6		LCD_DATA06	ALT0	
LCD_DATA7		LCD_DATA07	ALT0	
LCD_DATA8		LCD_DATA08	ALT0	
LCD_DATA9		LCD_DATA09	ALT0	
LCD_DATA10		LCD_DATA10	ALT0	
LCD_DATA11		LCD_DATA11	ALT0	
LCD_DATA12		LCD_DATA12	ALT0	
LCD_DATA13		LCD_DATA13	ALT0	
LCD_DATA14		LCD_DATA14	ALT0	
LCD_DATA15		LCD_DATA15	ALT0	
LCD_DATA16		LCD_DATA16	ALT0	
LCD_DATA17		LCD_DATA17	ALT0	
LCD_DATA18		LCD_DATA18	ALT0	
LCD_DATA19		LCD_DATA19	ALT0	
LCD_DATA20		LCD_DATA20	ALT0	
LCD_DATA21		LCD_DATA21	ALT0	
LCD_DATA22		LCD_DATA22	ALT0	
LCD_DATA23		LCD_DATA23	ALT0	
LCD_ENABLE	Enable signal	LCD_ENABLE	ALT0	IO
LCD_HSYNC	HSYNC signal	LCD_HSYNC	ALT0	I
LCD_RD_E	RD_E signal	LCD_ENABLE	ALT1	IO
LCD_RESET	Reset signal	LCD_RESET	ALT0	IO
LCD_RS	RS signal	LCD_HSYNC	ALT1	O
LCD_VSYNC	VSYNC signal	LCD_VSYNC	ALT0	I
LCD_WR_RWN	WR signal	LCD_CLK	ALT1	IO

32.3 Clocks

The following table describes the clock sources for eLCDIF. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 32-2. eLCDIF Clocks

Clock name	Clock Root	Description
apb_clk	axi_clk_root	AXI clock
pix_clk	lcdif_pix_clk_root	Pixel clock

32.4 Functional Description

Bus Interface Mechanisms through Initializing the eLCDIF, describe the internal pipeline for the MPU write/read interface, VSYNC, DOTCLK, and DVI interfaces. Differences for each mode are then described in separate sections, as follows:

- MPU Interface
- VSYNC Interface
- DOTCLK Interface
- ITU-R BT.656 Digital Video Interface (DVI)

eLCDIF pin usage by interface mode is described in eLCDIF Pin Usage by Interface Mode.

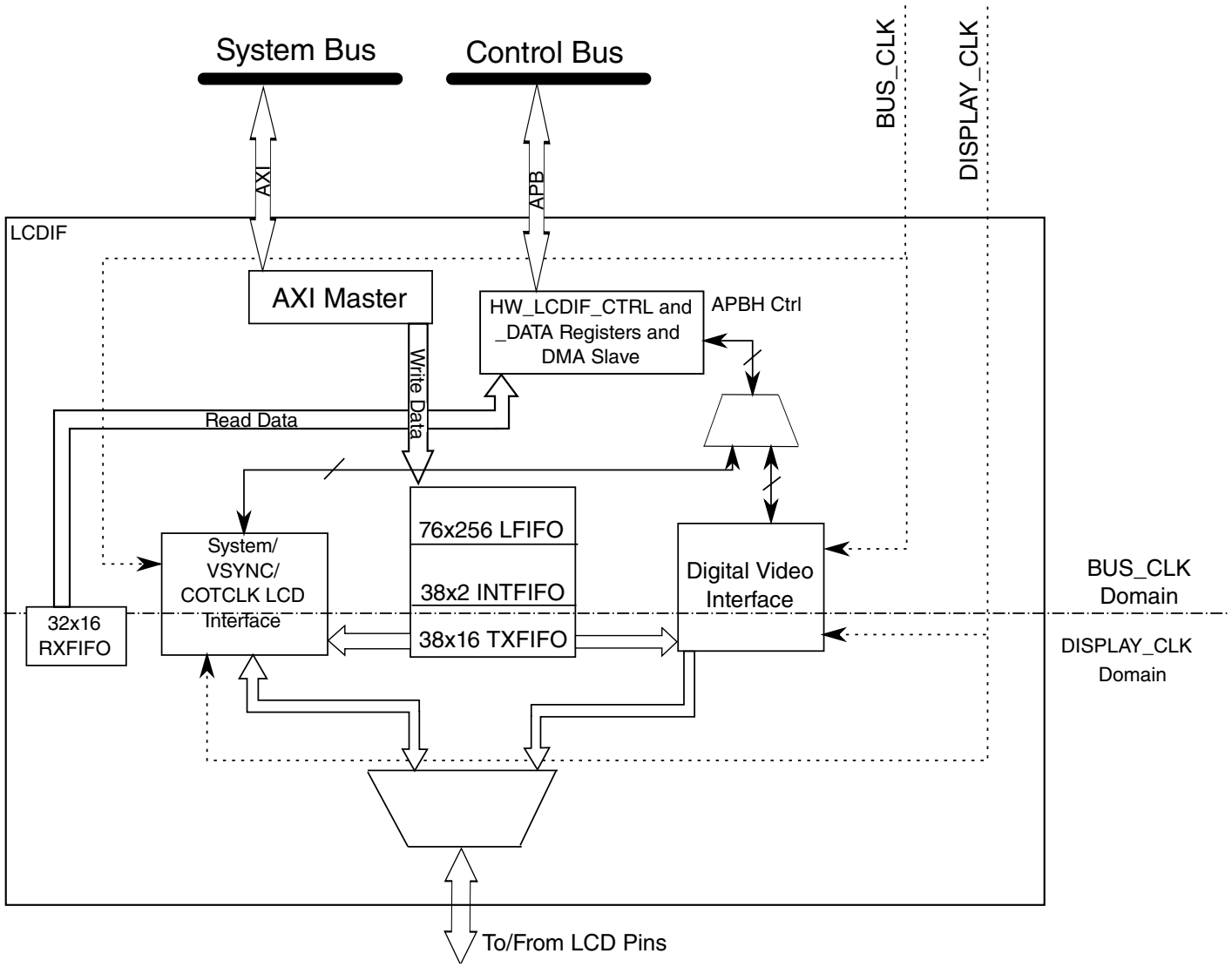


Figure 32-1. Top-Level Block Diagram of eLCDIF subsystem

32.4.1 Bus Interface Mechanisms

The LCDIF module has memory-mapped control, data and status registers. It provides several interfaces to transfer data between the display and SoC.

The bus master interface is used to initiate the requests to transfer data from external memory to the display. It is completely autonomous, or no CPU intervention is required, to manage the cyclical nature of refreshing standard display types. Bus mastering can also be used for MPU mode data writes.

The PIO interface is used to interface to "Smart" displays to transfer frame buffer data and control information to/from the external display. The host CPU executes display drivers to manage the display solution.

The following sections describe the system bus interface mechanisms.

32.4.1.1 Bus Master Operation in Write/Display Modes

The eLCDIF block has a bus master interface that initiates requests for data to drive the display. The LCDIF_MASTER bit must be set to 1 to enable the bus master interface. Software should program all control registers required to transfer the frame sequence.

In the MPU and VSYNC modes, single frames are transferred. When a complete frame is transferred, eLCDIF enters idle and clears the RUN bit in the CTRL register. For subsequent frame transmission, the eLCDIF setup sequence should be repeated.

The DOTCLK and DVI modes are used to refresh the display at the desired refresh rate and resolution. These modes are used to drive displays that do not integrate a display buffer memory. When the display is refreshed, the eLCDIF will automatically update the LCDIF_CUR_BUF_ADDR register with the value in LCDIF_NEXT_BUF_ADDR at the end of current frame and start fetching the next frame from the new address. If the LCDIF_NEXT_BUF_ADDR register was not updated within a frame refresh cycle, eLCDIF will keep transmitting the last frame until a new value is programmed into that register.

eLCDIF also provides the capability of interlacing a progressive frame by fetching odd lines in the first field and then fetching even lines in the second field. This feature can be used in the DVI mode and can be turned on by setting the INTERLACE_FIELDS bit in the LCDIF_CTRL1 register.

32.4.1.2 System Bus Master Performance

The performance of the eLCDIF block can be controlled by changing the burst length and the outstanding cycle issuing capability depending on the memory bandwidth requirements. Two fields in the LCDIF_CTRL2 register will throttle system memory requests. The LCDIF_CTRL2_OUTSTANDING_REQS field will control how many requests the eLCDIF can have in flight on any given clock cycle. This should be programmed based on the expected system bus latency for returned read data. Also, the LCDIF_CTRL2_BURST_LEN_8 bit will set the number of 64 bit words requested for each eLCDIF system bus request to either 8 or 16 QWORDS. Generally, 4 outstanding requests of length 16 will provide enough performance to drive any standard display resolution. These configuration bits are intended to change the access pattern of the eLCDIF to optimize system bus throughput when other system masters will contend for system memory resources.

The LCDIF_THRES register can also be used to optimize bus throughput and power consumption.

The LCDIF_THRES_PANIC value can be used to raise the priority of requests initiated by the eLCDIF to alter how the eLCDIF requests are arbitrated by the system bus infrastructure. The panic output control signal is raised when the number of 32bpp pixel equivalents in the LFIFO is less than this programmed value. Since the LFIFO is arranged as a 256x64bit quadword FIFO, it contains two 32bpp pixels per quadword, or 512 32bpp pixels total. To set the panic output when 3/4s of the LFIFO is empty, set the LCDIF_THRES_PANIC value to 3/4 * 512, or 128. The panic signal output is used to assess higher priority to eLCDIF system requests to avoid eLCDIF under run errors during periods of high system bandwidth utilization.

The features available with the LCDIF_THRES register require support from system clocking and dynamic priority control. Refer to the appropriate block documentation to assess the system support for these features.

32.4.2 Write Data Path

eLCDIF supports raster based frame buffers and there is no support for tiled buffers.

There are several options to accommodate endianness of display buffers in memory before the data is processed for the external display. The INPUT_DATA_SWIZZLE field in the LCDIF_CTRL register provides the following options for data word multiplexing:

```
00 (0): No swizzle (little-endian)
01 (1): Swap bytes 0 and 3, swap bytes 1 and 2 (big-endian)
10 (2): Swap half-words
11 (3): Swap bytes within each half-word
```

The WORD_LENGTH field of LCDIF_CTRL register indicates the input data/pixel format. LCDIF_TRANSFER_COUNT register denotes how much data is contained in each frame. The H_COUNT field of this register indicates the number of pixels per line and V_COUNT indicates the total number of lines per frame. A special bit field in the LCDIF_CTRL1 register, called the BYTE_PACKING_FORMAT, can be used to specify which bytes within the 32-bit word are going to be valid. For example, if the entire 32-bit word is valid, BYTE_PACKING_FORMAT should be set to 0xF, if only lower 3 bytes of each word in the frame buffer are valid, then BYTE_PACKING_FORMAT should be set to 0x7.

The LCD_DATABUS_WIDTH field in LCDIF_CTRL register suggests the width of the bus going to the external display controller. There is an option to source all 32 bits of the input word and transfer it to the output I/O display interface. Refer to the system I/O muxing options for support of this feature. If the LCD_DATABUS_WIDTH is not the

same as WORD_LENGTH, eLCDIF will perform RGB to RGB color space conversion. For example, if the input frame has fewer bits per pixel than the display, as in a 16 bpp input frame going to 24 bpp LCD, eLCDIF will pad the MSBs of each color to the LSBs of the same color for each pixel. If the input frame has more bits per pixel than the display, for example, 24 bpp input frame going to 16 bpp LCD, eLCDIF will drop the LSBs of each color channel to convert to the lower color depth. eLCDIF also has the capability to support delta pixel displays by swizzling the R, G and B colors of each pixel in the odd and even lines of the frame separately by programming the ODD_LINE_PATTERN and the EVEN_LINE_PATTERN bit fields. This operation occurs after the RGB-to-RGB color space conversion operation.

eLCDIF also supports RGB to YCbCr 4:2:2 color space conversion. This is useful in the DVI mode since the TV encoder requires input in YCbCr 4:2:2 format. The LCDIF_CSC* registers have complete programmability over the CSC coefficients and offsets. The values must be written into these registers in the signed two's complement format.

The following list shows how the different input/output combinations can be obtained:

- WORD_LENGTH=1 indicates that the input is 8-bit data. This is most likely going to be used for sending commands in MPU interface, or maybe a gray scale image. Any combination of BYTE_PACKING_FORMAT [3:0] is permissible.

  Limitation: H_COUNT must be a multiple of the sum of BYTE_PACKING_FORMAT [3], BYTE_PACKING_FORMAT [2], BYTE_PACKING_FORMAT [1] and BYTE_PACKING_FORMAT [0]. LCD_DATABUS_WIDTH must be 1, indicating an 8-bit data bus.

- WORD_LENGTH=0 implies the input frame buffer is RGB 16 bits per pixel. DATA_FORMAT_16_BIT field determines the pixels are RGB 555 or RGB 565.

  Limitation: BYTE_PACKING_FORMAT [3:0] should be 0x3 or 0xC if there is only one pixel per word. If there are two pixels per word, it should be 0xF and H_COUNT will be restricted to be a multiple of 2 pixels.

- WORD_LENGTH=2 indicates that input frame buffer is RGB 18 bits per pixel, that is, RGB 666. The valid RGB values can be left-aligned or right-aligned within a 32-bit word. The alignment of the valid 18 bits within a word is indicated by the DATA_FORMAT_18_BIT bit.

  Limitation: BYTE_PACKING_FORMAT can be 0x7, 0xE or 0xF. Packed pixels are not supported in this case. H_COUNT can be any number.

- WORD_LENGTH=3 indicates that the input frame-buffer is RGB 24 bits per pixel (RGB 888). If BYTE_PACKING_FORMAT [3:0] is 0x7, it indicates that there is only one pixel per 32-bit word and there is no restriction on H_COUNT. This is also

the option that provides 32 bit output depending on the I/O muxing options available. The fourth byte, or bits [31:24], and connected to the I/Os if this muxing is available in the chip package.

Limitation: If BYTE_PACKING_FORMAT [3:0] is 0xF, it indicates that the pixels are packed, that is, there are 4 pixels in 3 words or 12 bytes and H_COUNT must be a multiple of 4 pixels.

- YCBCR422_INPUT=1 implies that the input frame is in YCbCr 4:2:2 format. BYTE_PACKING_FORMAT must be 0xF.

  Limitation: LCD_DATABUS_WIDTH must be 8-bit and H_COUNT must be a multiple of 2 pixels.

  ODD_LINE_PATTERN and EVEN_LINE_PATTERN must be 0 when any of RGB_TO_YCBCR422_CSC or INTERLACE_FIELDS or YCBCR422_INPUT bits is 1.

After the RGB to RGB or RGB to YCbCr 4:2:2 color space conversions, there is one more opportunity to swizzle the data before sending it out to the display or the encoder. This can be done with the CSC_DATA_SWIZZLE field in the LCDIF_CTRL register, and it provides the same options as the INPUT_DATA_SWIZZLE register.

Finally, there is an option to shift the output data before sending it out to the display. This is done based on the SHIFT_DIR and SHIFT_NUM_BITS fields in LCDIF_CTRL register.

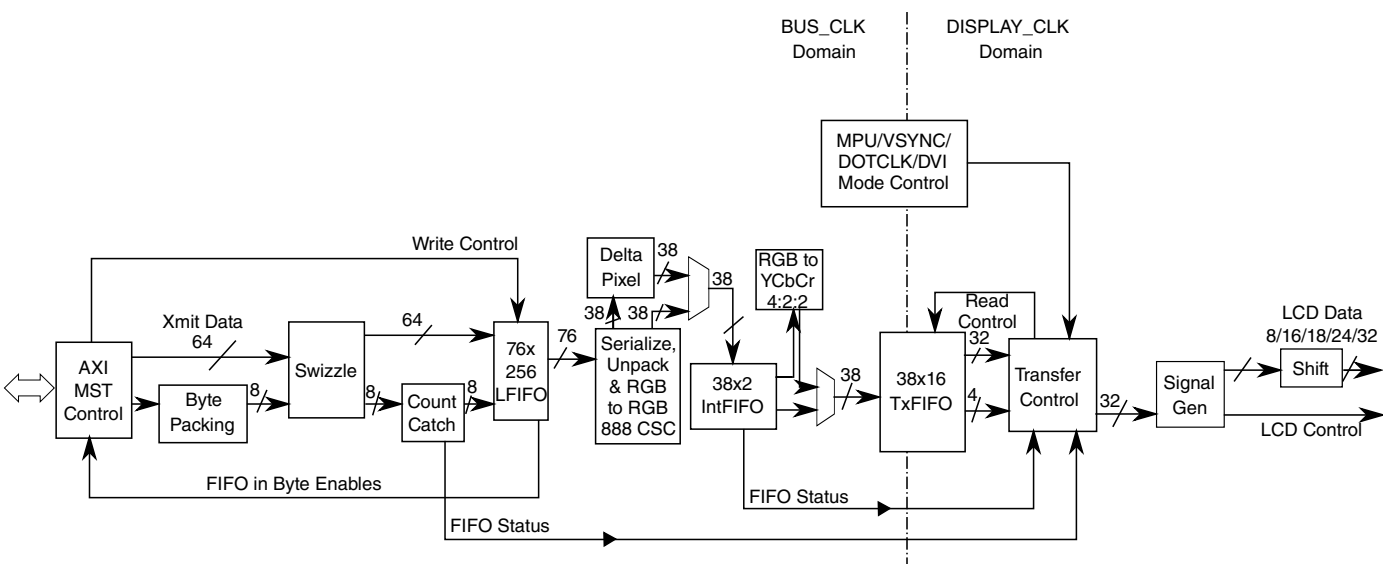


Figure 32-2. General Operations in Write Data Path

The examples in the following figures illustrate some different combinations of register programming for write mode. Assume that the data transferred over the system bus within a 32 bit word is organized as {A7-A0, B7-B0, C7-C0, D7-D0} in 8-bit mode and {A15-A0, B15-B0} in 16-bit mode.

In this example, all 32 bits of the input word are transferred out over an 8 bit display bus. Each byte within the 32 bit word is shifted to the right with zeros appended to I/O bits D[7:6]. The input data bits [7:2] are shifted to the right by 2 bits and presented on the D[5:0].

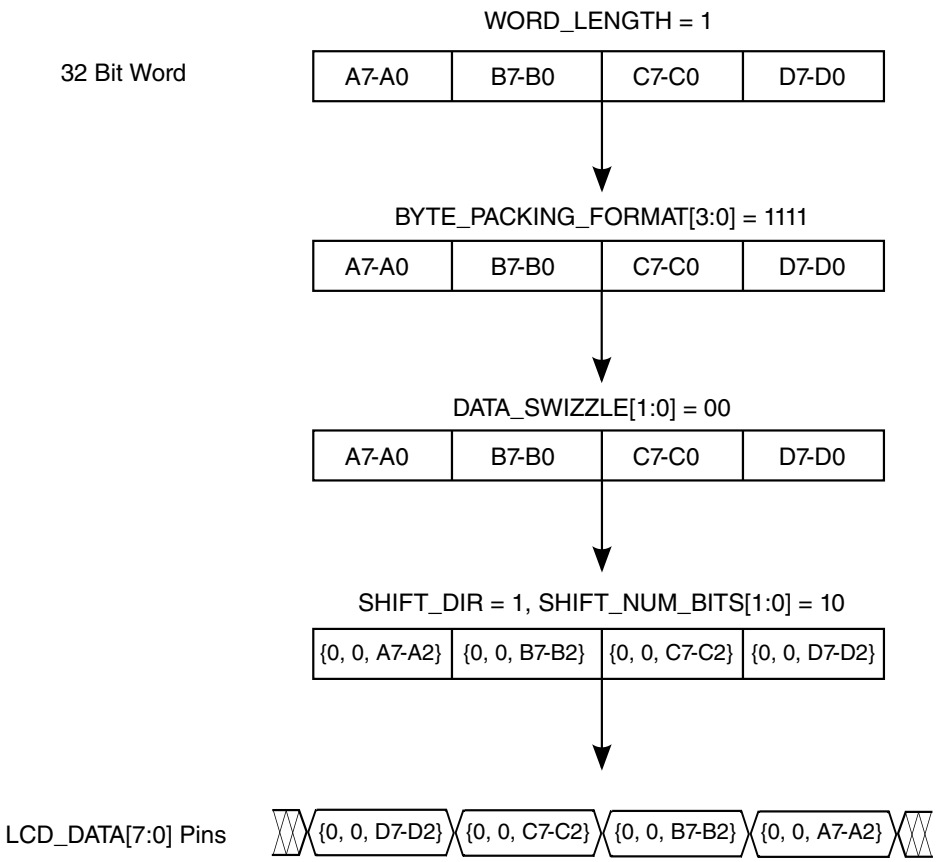


Figure 32-3. Register programming for write mode

In this 8 bit display interface example, one byte of the input word is deleted and not transferred over the external 8 bit display interface. This mode could be used to transfer 24bpp pixels over the 8 bit interface. In this case, the 4th unused byte is not transferred.

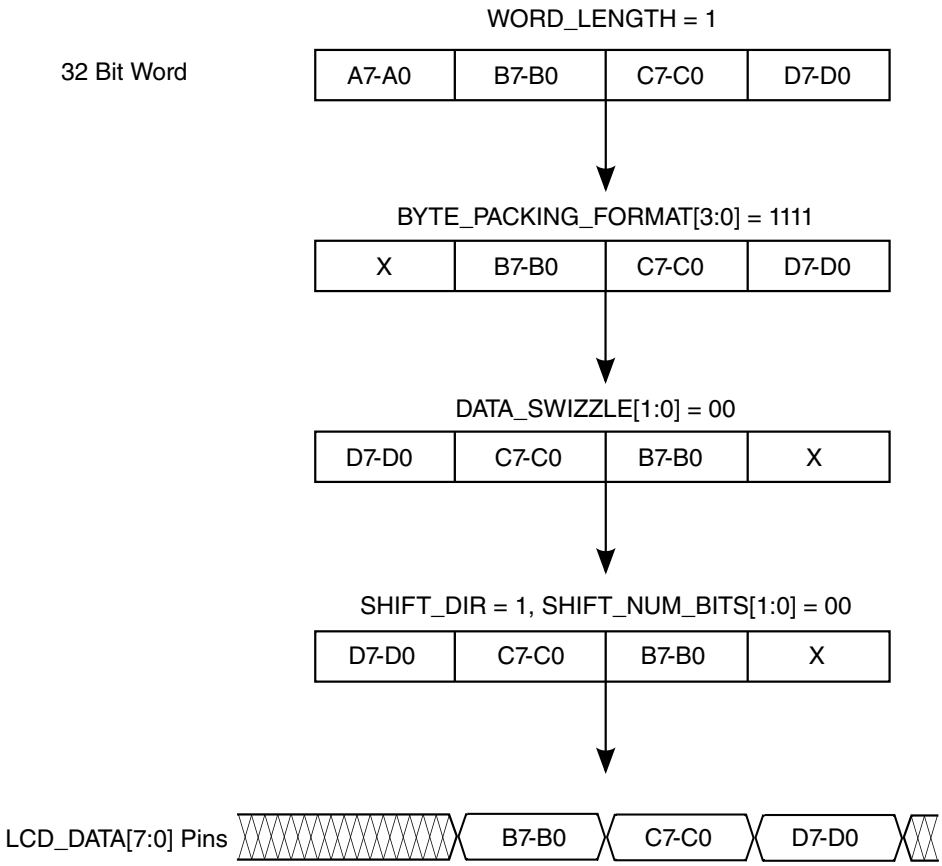


Figure 32-4. Register programming for write mode

The following example uses a 16 bit display interface. Each 16 bit half word is shifted to the right by two bits with zeros appended to the most significant two bits.

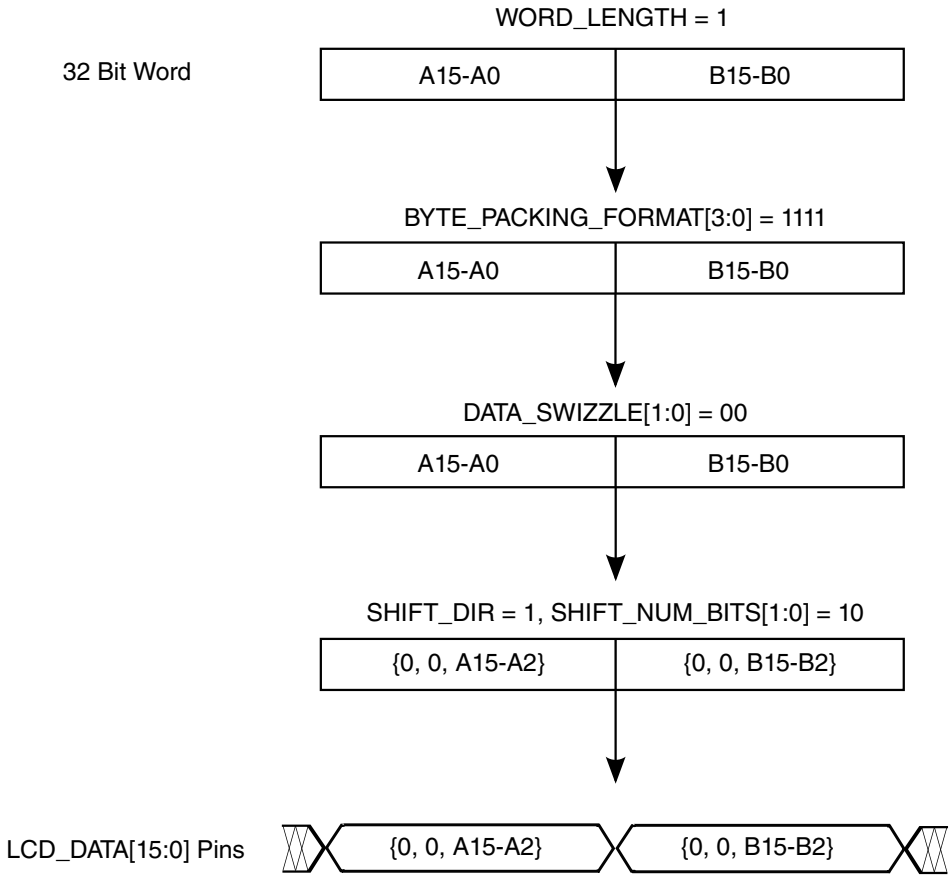


Figure 32-5. Register programming for write mode

This example indicates how an unpacked frame buffer can sourced for display. Only a single 16 bit half word within the 32 bit word is tranferred out via the 16 display bus.

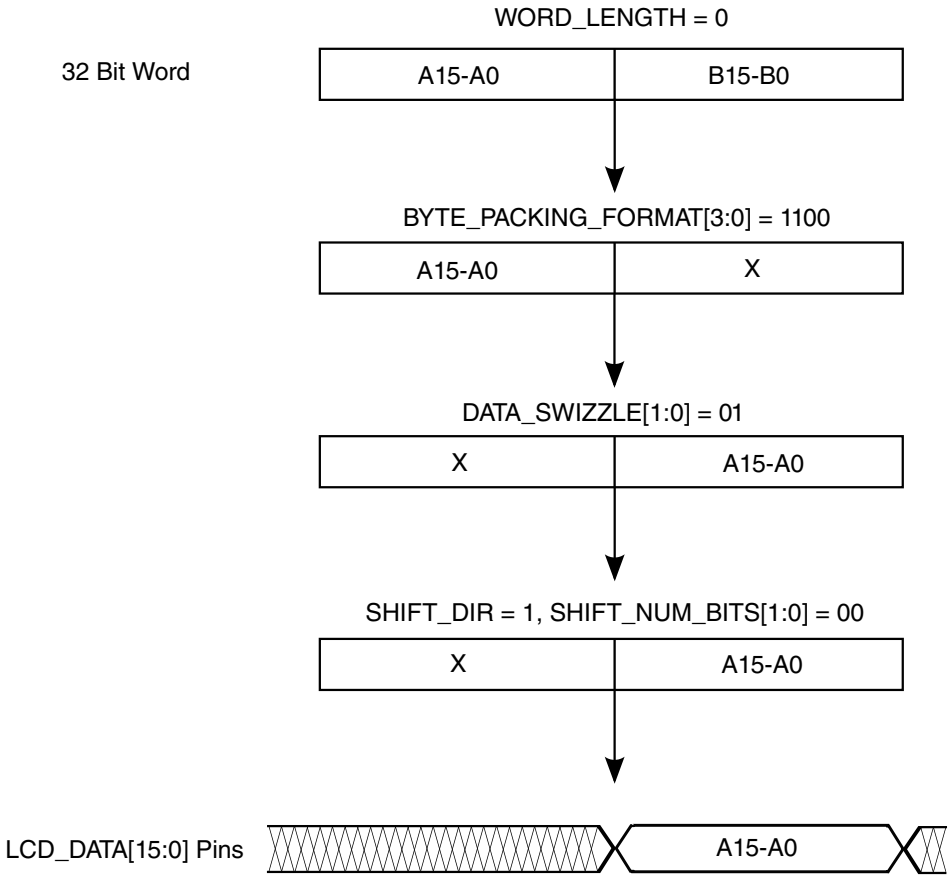


Figure 32-6. Register programming for write mode

32.4.3 Read Data Path

Figure 32-7 shows the MPU read data path in detail.

eLCDIF can read from an external display that follows the 6800/8080 MPU protocol.

The display bus width is determined by the LCD_DATABUS_WIDTH bit field. The data sampled at every read strobe is called a subword and the number of subwords that can be packed in a 32-bit word is given by the READ_MODE_NUM_PACKED_SUBWORDS bit field. The INITIAL_DUMMY_READ bit field directs the eLCDIF to skip the number of programmed subwords before starting to process read data. This feature is useful in the case of an LCD controller that returns the last written data the first time a read is issued, and then sends the correct data thereafter. SHIFT_DIR and SHIFT_NUM_BITS bit fields indicate whether the data needs to be shifted before getting stored in the internal registers. For example, a value of 2 in READ_MODE_NUM_PACKED_SUBWORDS if lcd

databus width is 8 bits indicates two bytes should be packed in a 32-bit word, while if the lcd databus width is 16 bits, it indicates that two half words (or 4 bytes) should be packed.

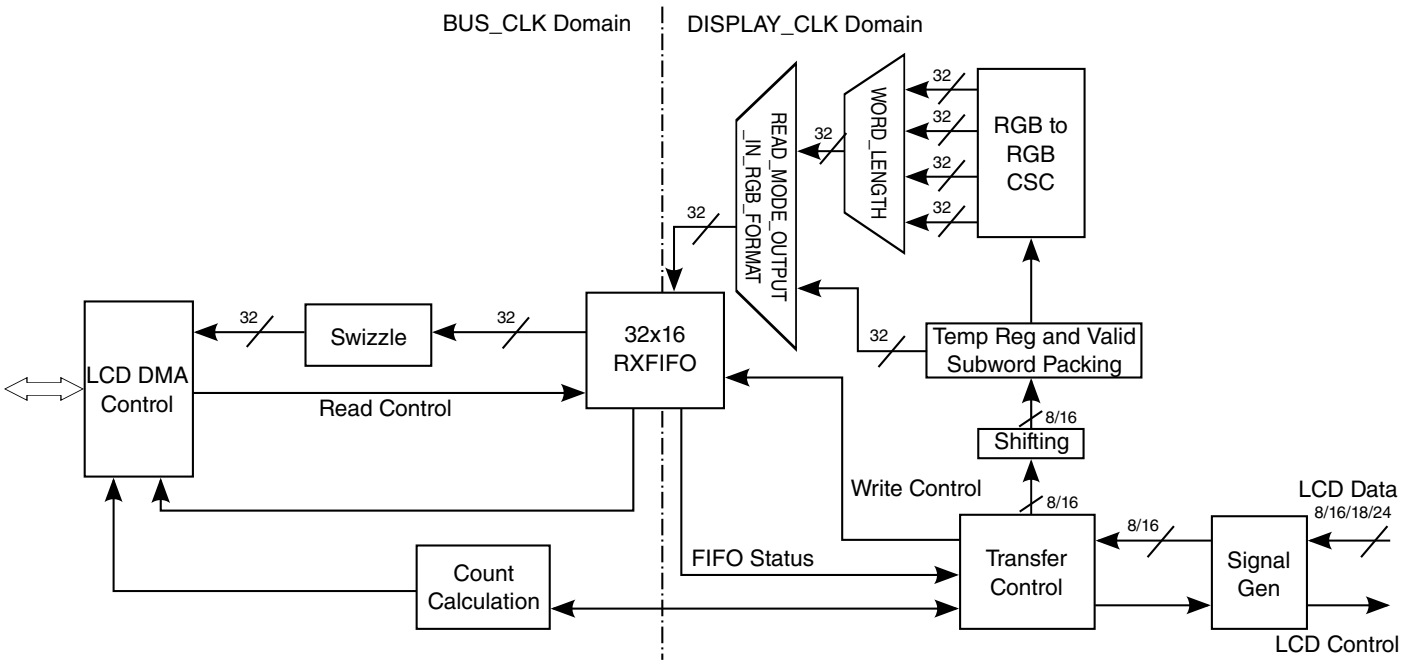


Figure 32-7. MPU Read Data Path

After the last subword within a word is reached, the block looks at the READ_PACK_DIR in the HW_LCDIF_CTRL2 register. If this bit is set, the block will swizzle the data, but only within the valid bytes, unlike in the write mode, where swizzle occurs across all 4 bytes. If the READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, eLCDIF will convert the data obtained from the READ_PACK_DIR operation into 24-bit unpacked RGB and then re-convert it into 16/18/24 bpp RGB depending on the WORD_LENGTH field. The DATA_FORMAT_16/18/24_BIT bit fields are also considered while converting to 24-bit unpacked RGB format. For example, if DATA_FORMAT_18_BIT is 1, the RGB666 data will be packed in the upper bits [31:4]of a 32-bit word, and that bit is 0, the data will be packed in the lower bits [17:0]. After all these operations, the data gets written into the RXFIFO.

The following figures show some examples of how data is handled in different MPU read modes.

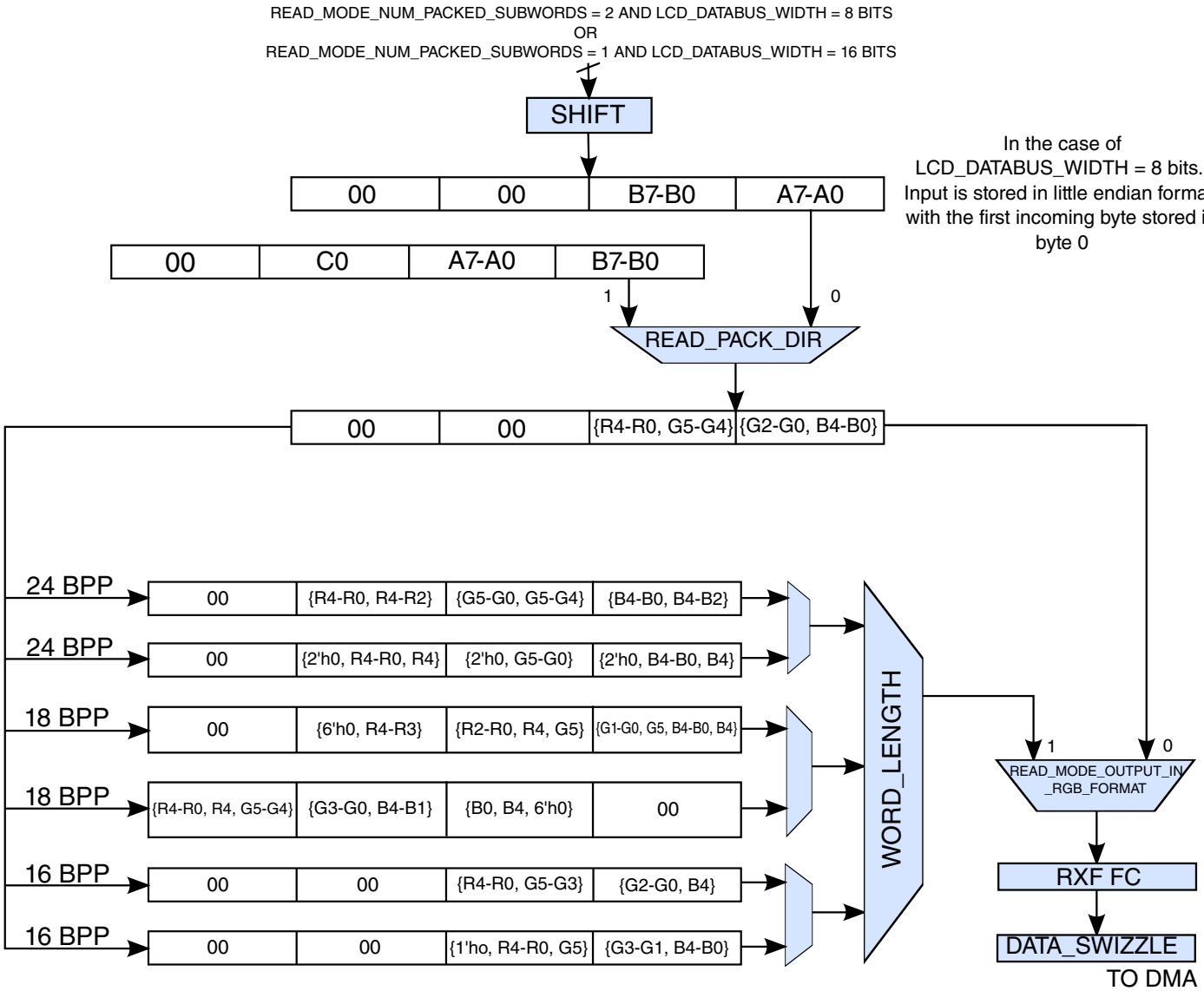


Figure 32-8. Data in MPU read mode

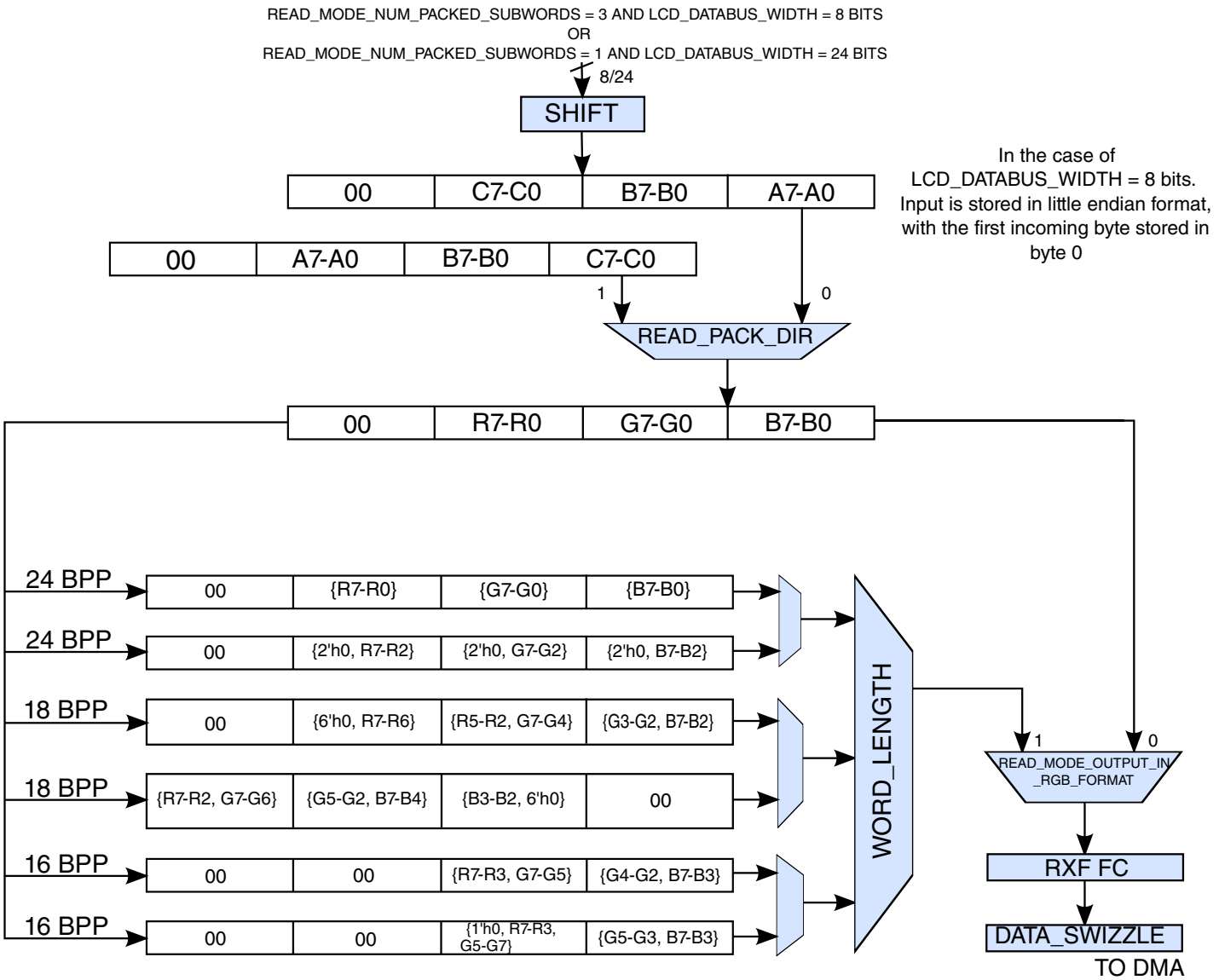


Figure 32-9. Data in MPU read mode

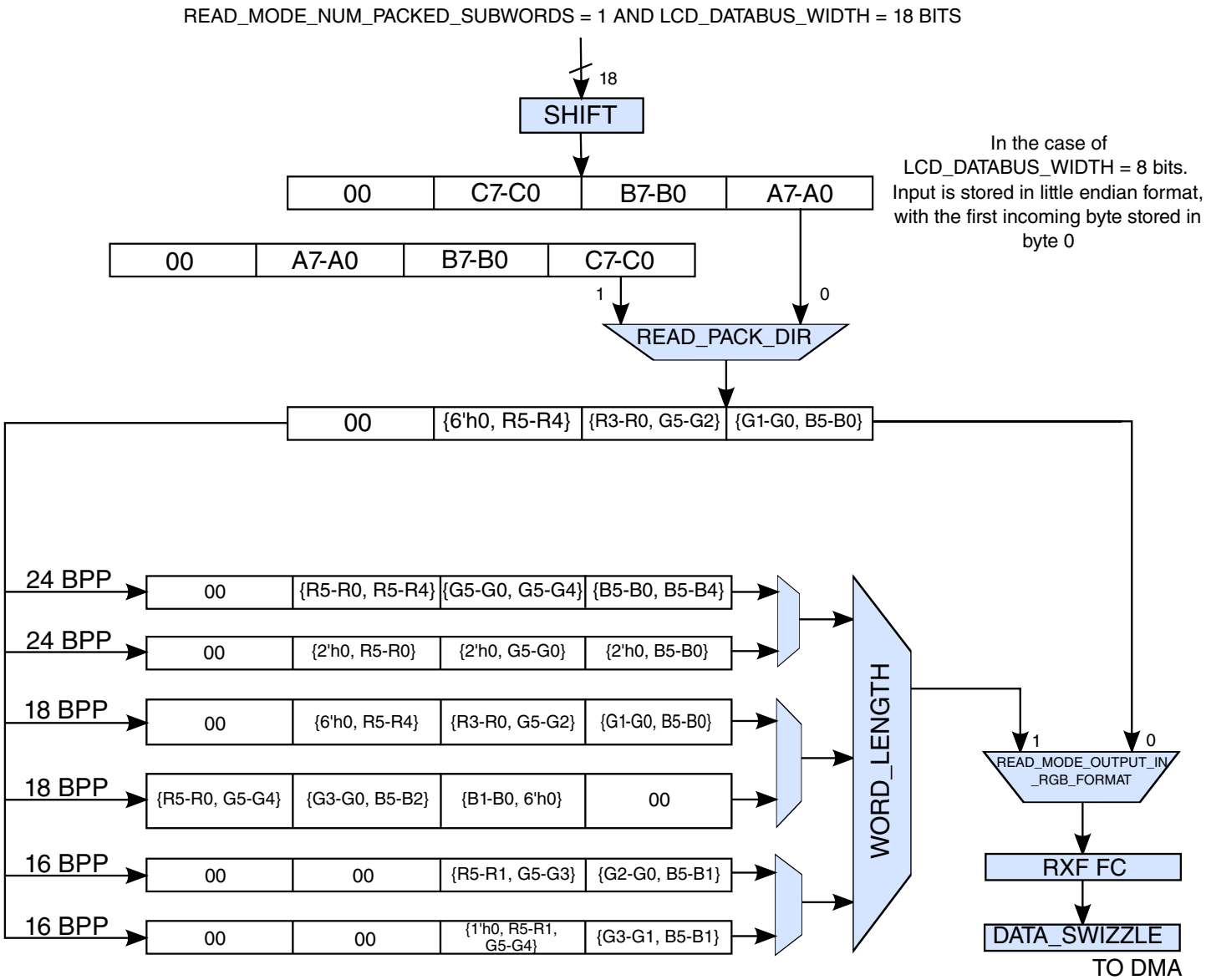


Figure 32-10. Data in MPU read mode

Restrictions:

READ_PACK_DIR should only be used if it is required to swizzle the subwords before doing RGB to RGB CSC, otherwise the DATA_SWIZZLE field should be used to swizzle across bytes.

READ_PACK_DIR must be 0 if LCD_DATABUS_WIDTH is 8 bits and READ_MODE_NUM_PACKED_SUBWORDS =1

If READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, the following restrictions should be followed:

- If LCD_DATABUS_WIDTH = 8 bits, then READ_MODE_NUM_PACKED_SUBWORDS <= 3.
- If LCD_DATABUS_WIDTH = 16/18/24 bits, then READ_MODE_NUM_PACKED_SUBWORDS = 1.

32.4.4 eLCDIF Interrupts

eLCDIF supports a number of interrupts to aid controlling and status reporting of the block.

All the interrupts have individual mask bits for enabling or disabling each of them. They all get funneled through a single interrupt line connected to the interrupt collector (ICOLL).

The following list describes the different interrupts supported by eLCDIF:

- Underflow interrupt is asserted when the clock domain crossing FIFO (TXFIFO) becomes empty but the block is in active display portion during that time. Software should take corrective action to make sure that this does not happen.
- In the bus master mode, the overflow interrupt will be asserted if the block has requested more data than it's FIFOs could hold. In the read mode, it will be asserted if the RxFIFO becomes full and the block reads more data.
- VSYNC edge interrupt will be asserted every time a leading VSYNC edge occurs.
- Cur_frame_done interrupt occurs at the end of every frame in all modes except DVI. In DVI mode, if IRQ_ON_ALTERNATE_FIELDS bit is set, it will occur at the end of every frame, otherwise it will occur at the end of every field.

32.4.5 Initializing the eLCDIF

This section describes write modes and MPU read mode.

32.4.5.1 Write Modes

The following initialization steps are common to all eLCDIF write modes of operation before entering any particular mode.

Initialization steps:

1. Configure the external I/Os to correctly interface the external display.
2. Start the DISPLAY_CLK clock and set the appropriate frequency by programming the registers in CCM.

3. Start the BUS_CLK and set the appropriate frequency by programming the registers in CCM.
4. Bring the eLCDIF out of soft reset and disable the clock gate bit.
5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller. See Behavior During Reset for more information on Reset requirements.
6. Make sure READ_WRITEB bit in HW_LCDIF_CTRL register is 0.
7. Select the transfer mode of operation. The LCDIF_MASTER bit in HW_LCDIF_CTRL register determines the transfer mode selected. Bus master (LCDIF_MASTER =1) or PIO (LCDIF_MASTER =0) mode are the transfer modes to select.
8. Set the INPUT_DATA_SWIZZLE according to the endianness of the LCD controller. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.
9. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24/32-bit input. Also, select the correct 16/18/24 bit data format with the corresponding fields in HW_LCDIF_CTRL register.
10. Set the BYTE_PACKING_FORMAT field in HW_LCDIF_CTRL1 according to the input frame.
11. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24/32-bit output.
12. Enable the necessary IRQs.

32.4.5.2 MPU Read Mode

The following initialization steps should be done to enter the MPU read mode of operation:

Initialization steps:

1. Configure the external I/Os to correctly interface the external display.
2. Start the DISPLAY_CLK and set the appropriate frequency by programming the registers in CCM.
3. Start the BUS_CLK and set the appropriate frequency by programming the registers in CCM.
4. Bring the eLCDIF out of soft reset and clock gate.
5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller.
6. Set the READ_WRITEB bit in LCDIF_CTRL register to 1.
7. Set the LCDIF_MASTER bit in LCDIF_CTRL register to 0. Bus master mode is not supported for reading data from the display.

8. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.
9. Indicate if the read data needs to color-space-converted and stored in a different RGB format by setting the READ_MODE_OUTPUT_IN_RGB_FORMAT field accordingly.
10. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24-bit input if READ_MODE_OUTPUT_IN_RGB_FORMAT is required. Also, select the correct 16/18/24 bit data format with the corresponding fields in LCDIF_CTRL register.
11. Set the READ_MODE_NUM_PACKED_SUBWORDS field in LCDIF_CTRL2 according to the number of subwords per word required to be packed.
12. Set the READ_PACK_DIR to 1 if it is required to store the data in big-endian format.
13. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24-bit output.
14. Enable the necessary IRQs.

32.4.6 MPU Interface

The MPU interface is used to transfer data and commands between the SoC via the eLCDIF and the external display at modest data rates.

Bus master or PIO transactions using the LCDIF_DATA register can be used for MPU mode write operations. For MPU mode read operations, only PIO can be used. eLCDIF can support the 6800 as well as the 8080 MPU protocol. If DOTCLK_MODE, DVI_MODE and VSYNC_MODE bits in LCDIF_CTRL registers are 0, it implies that the block is in MPU interface mode of operation. The LCDIF MPU mode has four basic timing parameters: Setup and Hold for the Command/Data register selection (TCS, TCH) and Setup and Hold for the Data bus (TDS, TDH). These parameters are expressed in DISPLAY_CLK cycles. The LCD_WR signal is used as the write strobe while LCD_RS signal is typically used to switch between command and data modes.

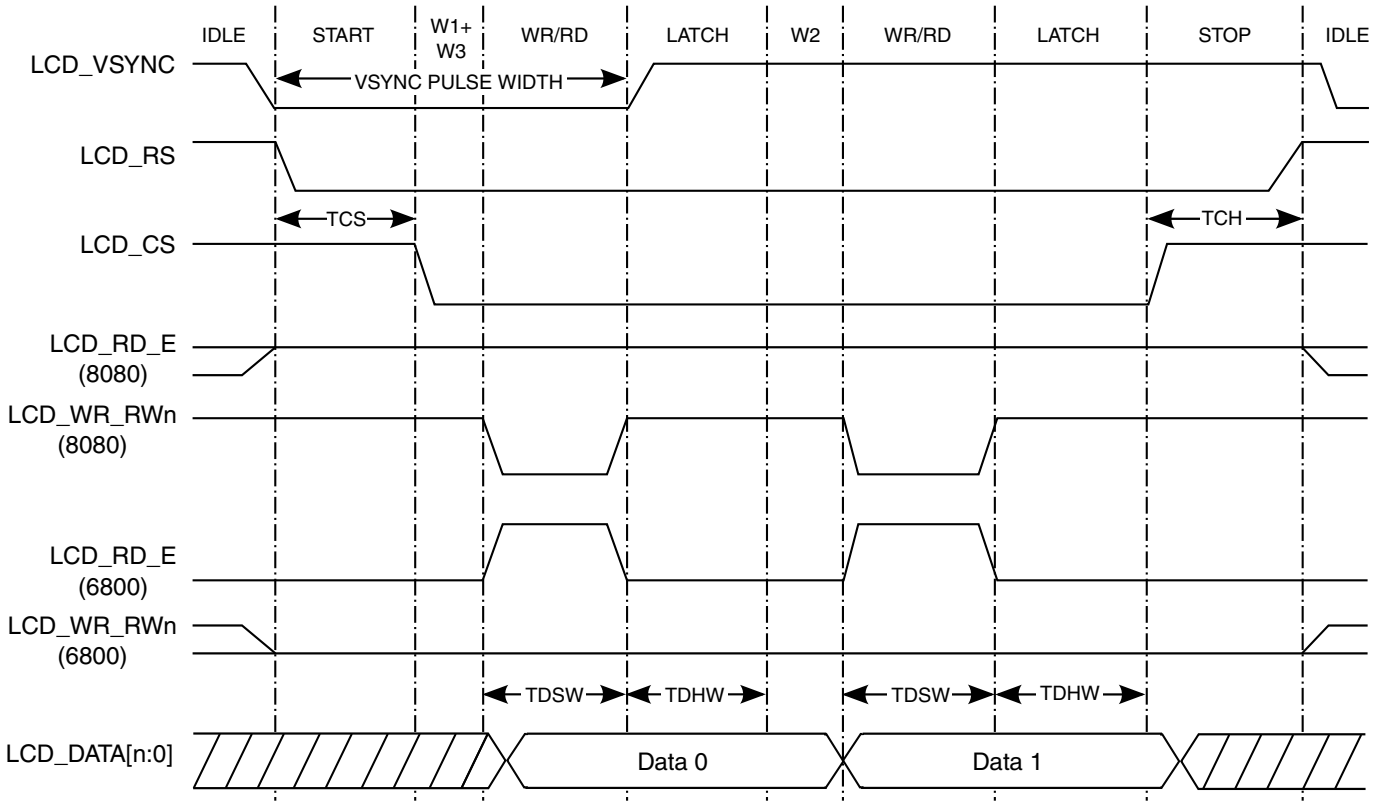


Figure 32-11. Timing in write mode of 6800 and 8080 protocols

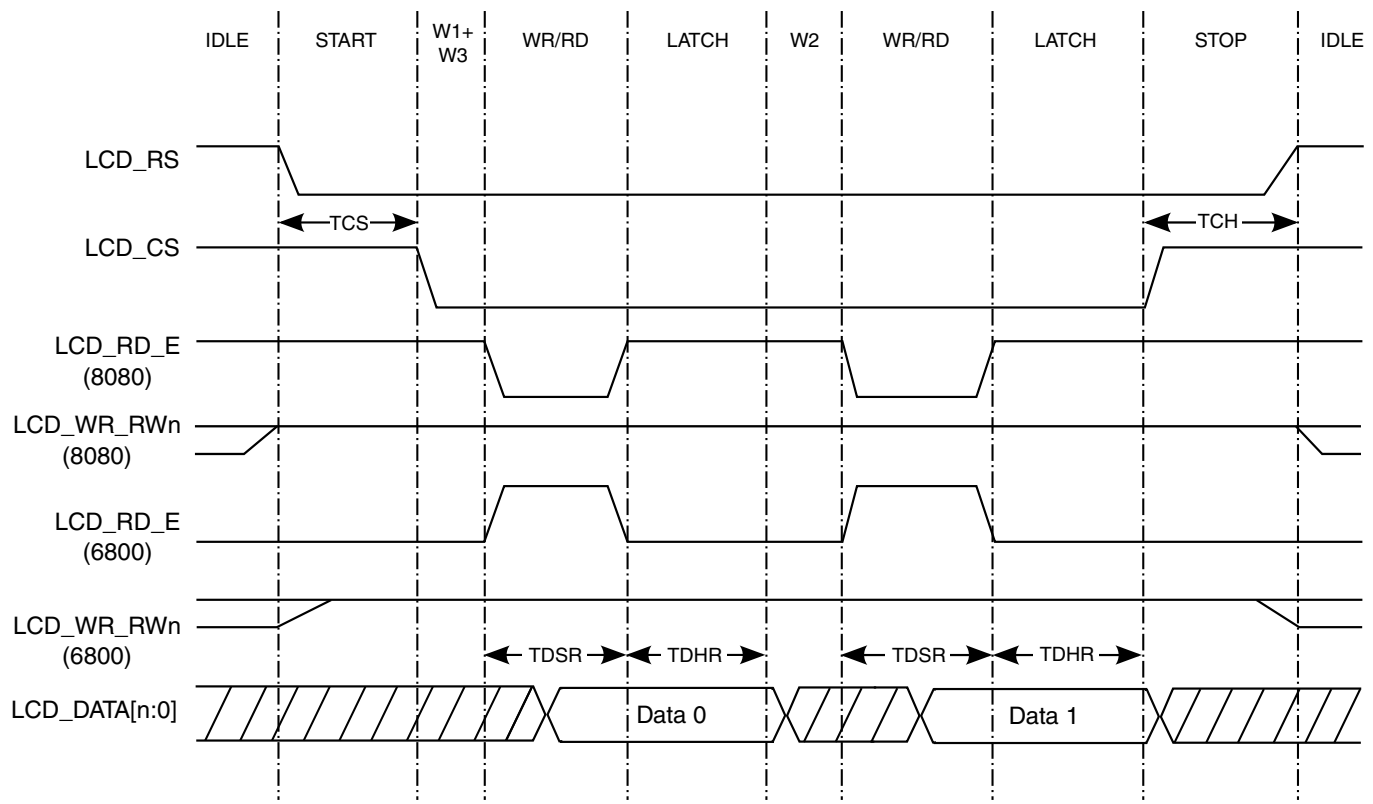


Figure 32-12. Read timing interface in 6800 and 8080 protocols

The eLCDIF has flexible pin and strobe timings which enable it to optimally support a wide range of LCDs. The minimum cycle time is two DISPLAY_CLK cycles (TDS=TDH=1). For example, this results in a maximum LCD data rate of 12 MB/s when DISPLAY_CLK is 24 MHz. TDS and TDH are 8-bit values, so the minimum eLCDIF period is 510 DISPLAY_CLK cycles (47 KHz with a 24 MHz DISPLAY_CLK). The timings are not automatically adjusted if the DISPLAY_CLK frequency changes, so it may be necessary to adjust the timings if DISPLAY_CLK changes.

In the MPU interface mode, the LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the eLCDIF has received/transmitted all the data as per the LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

32.4.6.1 Code Example to Initialize the eLCDIF in MPU Write Mode

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1(LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that the
                                    // idle state for LCD_RS signal is high, regardless of the
                                     // programming of the DATA_SELECT register.
BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, READ_WRITEB, 0);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 1);//Only if LCD controller implements a busy line
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values
based
                               // on DISPLAY_CLK frequency and timing requirements of
controller.
                              // Note that these register must be non-zero for correct
operation.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240); //For a 320 RGB x 240 display
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The eLCDIF is now ready to receive data via bus master PIO write transactions using the LCDIF_DATA register. Note that when using the PIO write operations to the LCDIF_DATA register, the software will need to poll the FIFO STATUS bits to ensure that it does not overflow the eLCDIF data buffers. When eLCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

32.4.7 VSYNC Interface

The VSYNC interface uses the same protocol as the MPU interface, with an additional signal VSYNC at the frame rate of the display, as shown in the figure given in MPU Interface section.

It is used in the moving picture display mode where data has to be written to the internal LCD buffer at a speed higher than the display rate and displayed in synchronization with the VSYNC signal. This mode is selected by setting the VSYNC_MODE bit in LCDIF_CTRL register. The VSYNC signal is programmable for period, polarity and direction. Many other programmable parameters are shared with the MPU interface. The VSYNC_OEB bit in LCDIF_VDCTRL0 register indicates whether the display controller will send the VSYNC signal, or whether it should be generated by eLCDIF. The timing of the VSYNC signal is based on the DISPLAY_CLK (make sure VSYNC_PULSE_WIDTH_UNIT = VSYNC_PERIOD_UNIT = 0 and VSYNC_ONLY = 1) and it is determined by the VSYNC_PERIOD, VSYNC_PULSE_WIDTH and VSYNC_POL fields in LCDIF_VDCTRL0-4 registers. The SYNC_SIGNALS_ON bit in LCDIF_VDCTRL4 register must be set if the target requires the VSYNC signal to be generated by eLCDIF. If the WAIT_FOR_VSYNC_EDGE bit in LCDIF_CTRL register is set, it indicates that the hardware should wait until it sees the leading VSYNC edge before starting the data transfer. The VERITCAL_WAIT_CNT indicates the number of DISPLAY_CLK cycles from the leading VSYNC edge after which data transfer will be started on the interface.

In the VSYNC interface mode, the LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the eLCDIF has received/transmitted all the data as per the LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

32.4.7.1 Code Example to Initialize eLCDIF in VSYNC Mode

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that
//the idle state for LCD_RS signal is high, regardless of the programming of the DATA_SELECT
//register.

BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 0);
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values
//based on DISPLAY_CLK frequency and timing requirements of controller. Note that these
//register must be non-zero for the MPU and VSYNC modes.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240);//For a 320 RGB x 240 display
//The following section indicates setting up the VSYNC signal timing when VSYNC is an output
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Making VSYNC signal an output
BF_CS1 (LCDIF_VDCTRL4, VSYNC_ONLY, 1); //Only need to generate VSYNC signal
BF_CS1 (VDCTRL0, VSYNC_POL, 0); //Setting the polarity of VSYNC signal to be low during
//VSYNC_PULSE_WIDTH time
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 0, VSYNC_PULSE_WIDTH_UNIT, 0);
BF_CS2 (LCDIF_VDCTRL1, VSYNC_PERIOD, 400000, VSYNC_PULSE_WIDTH, 100);//Frame display rate in
//terms of number of DISPLAY_CLKs.
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 0, HSYNC_PERIOD, 0);
BF_CS1 (LCDIF_VDCTRL3, VERTICAL_WAIT_CNT, 50);
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
```

```
BF_CS2 (LCDIF_CTRL, VSYNC_MODE, 1, WAIT_FOR_VSYNC_EDGE, 1); //set WAIT_FOR_VSYNC_EDGE if
//software wishes to transfer the next frame after the VSYNC edge occurs.
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The eLCDIF is now ready to receive data via bus master requests or PIO writes to the LCDIF_DATA register. When eLCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

32.4.8 DOTCLK Interface

The DOTCLK interface is another mode used in moving picture displays.

It includes the VSYNC, HSYNC, DOTCLK and (optional) ENABLE signals. The interface is popularly called the RGB interface if the ENABLE signal is present.

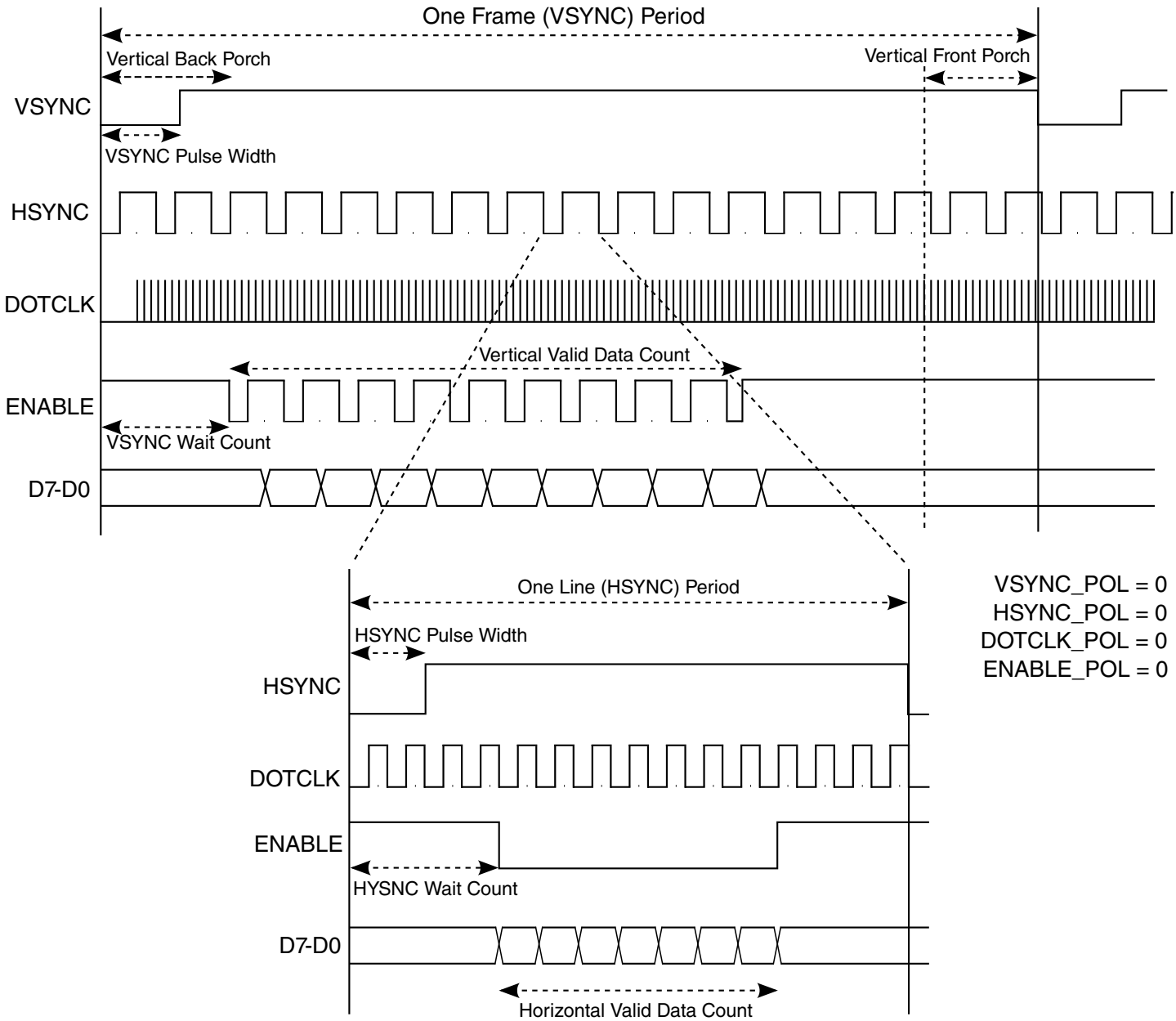


Figure 32-13. DOTCLK protocol with programmable parameters

The DOTCLK mode writes data at high speed to the LCD, and the display operation is synchronized with the VSYNC, HSYNC, ENABLE and DOTCLK signals. The polarities, periods and pulse-widths of the sync signals are programmable using the LCDIF_VDCTRL0-4 registers. The units for the VSYNC signal must be number of horizontal lines and can be selected using the VSYNC_PULSE_WIDTH_UNIT and VSYNC_PERIOD_UNIT bit fields. The VERTICAL_WAIT_CNT is by default given the same unit as the VSYNC_PERIOD. The DISPLAY_CLK frequency is managed by the CCM.

In DOTCLK mode, LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DOTCLK_MODE bit 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and issue the cur_frame_done interrupt.

32.4.8.1 Code Example

The following code shows an example for programming a 320x240 display. Note that setting up the display must be done through the MPU mode or via SPI.

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DOTCLK_MODE, 1);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 1); //Always for DOTCLK mode
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Vsync is always an output in the DOTCLK mode
BF_CS4 (LCDIF_VDCTRL0, VSYNC_POL, 0, HSYNC_POL, 0, DOTCLK_POL, 0, ENABLE_POL, 0);
BF_CS1 (LCDIF_VDCTRL0, ENABLE_PRESENT, 1);
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 1, VSYNC_PULSE_WIDTH_UNIT, 1);
BF_CS1 (LCDIF_VDCTRL0, VSYNC_PULSE_WIDTH, 2);
BF_CS1 (LCDIF_VDCTRL1, VSYNC_PERIOD, 280);
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 10, HSYNC_PERIOD, 360); //Assuming
                                                    //  LCD_DATABUS_WIDTH is 24bit
BF_CS2 (LCDIF_VDCTRL3, VSYNC_ONLY, 0);
BF_CS2 (LCDIF_VDCTRL3, HORIZONTAL_WAIT_CNT, 20, VERTICAL_WAIT_CNT, 20);
BF_CS1 (LCDIF_VDCTRL4, DOTCLK_H_VALID_DATA_CNT, 320);//Note that DOTCLK_V_VALID_DATA_CNT is
                              //implicitly assumed to be HW_LCDIF_TRANSFER_COUNT_V_COUNT
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

To stop the transfer completely, the ideal way is to make DOTCLK_MODE = 0. In that case, the block will transmit the contents in the FIFO and reset the RUN bit.

32.4.9 CSI HANDSHAKE INTERFACE

The LCDIF and CSI support a pipeline mode to use double buffers inside OCRAM for the video pass through from CSI to LCDIF, the pipeline buffer size can be 8 line or 16 line as configurable. The pipeline will be handle by hardware handshake signals between CSI and LCDIF. The LCDIF will have the capability to synchronize display with CSI based on the VSYNC signal from CSI. When LCDIF is enabled to start display,it can optionally wait for the VSYNC edge from CSI before it starts the display for next frame,The delay from VSYNC input to the start of next frame is programmable by LCDIF_SYNC_DELAY register.

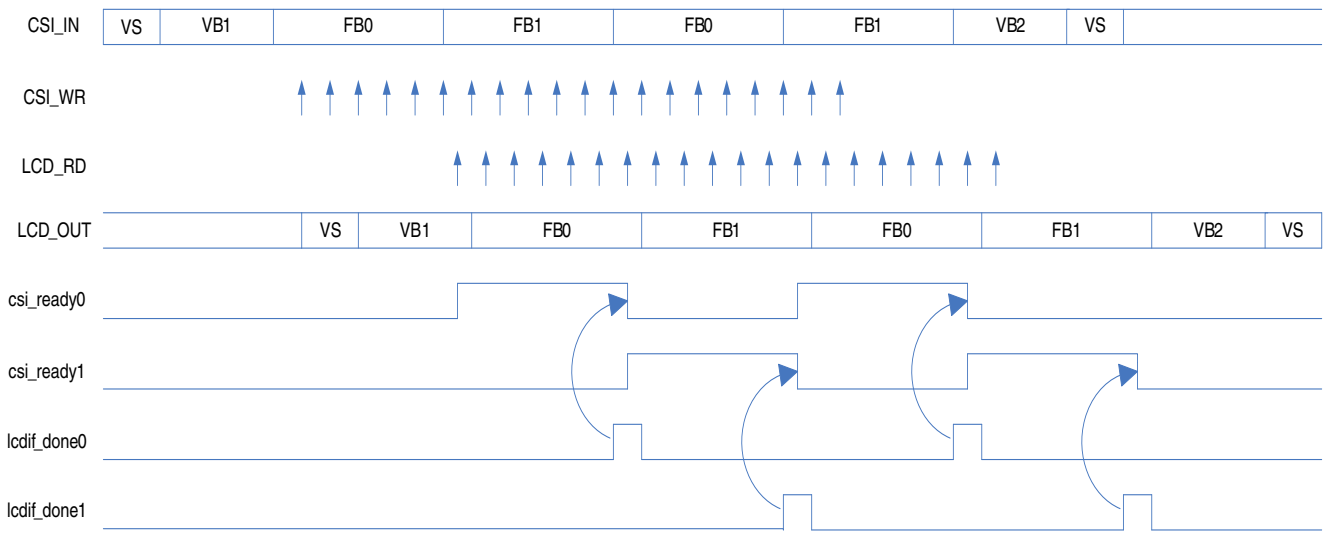


Figure 32-14. CSI HandShake Interface

When this mode is enabled ,The hardware handshake protocol will process. At the start of CSI with camera input,the CSI will put the input data to one frame buffer.When the frame buffer is ready for LCDIF display,CSI will assert csi_ready signal to indicate LCDIF to read the buffer and LCDIF will display data in this buffer.When one frame buffer reading finished ,LCDIF will set the lcdif_done signal to indicate CSI this buffer has been displayed and can be written again.With this double buffer handshake mode,the LCDIF can display the video input within very short delay and minimize DRAM bandwidth.

32.4.10 Alpha Blending Interface

The LCDIF have the capability to add an extra overlay on the normal display buffer,LCDIF can fetch data from two buffers and combine them before display,one buffer data can have the alpha value with the RGB pixels. With LCDIF_AS_CTRL[AS_ENABLE] is set, the LCDIF will start fetching alpha surface buffer data in bus master mode and combine it with another buffer.

The LCDIF_AS_CTRL[ALPHA_CTRL] bits determines how the alpha value is constructed for the alpha surface and alpha blending process is as same as in PXP block, for the alpha blend and color key process refer to the PXP block descriptions.

32.4.11 ITU-R BT.656 Digital Video Interface (DVI)

ITU-R BT.656 Digital Video Interface shown below transmits 4:2:2 YCbCr digital component video to a digital video encoder that can translate it into 525/60 or 625/50 analog TV signal.

Unique timing codes (timing reference signals) are embedded within the video stream to indicate the different timing events that would have been otherwise indicated by VSYNC, HSYNC and BLANK signals. The hardware supports 8-bit data transfers; the pins are shared with the lower 8 bits of LCD data bus. The LCD_RS pin is shared with the clock signal of the interface (called CCIRCLK here for uniqueness). CCIRCLK also can be obtained on the LCD_DOTCK pin. The mode shares the write FIFO with the LCD interface and the associated pipeline. The programmable parameters in registers LCDIF_DVICTRL0-3 allow setting the total number of horizontal lines per frame, vertical and horizontal blanking interval, odd and even field start and end positions, and so on. In short, these parameters are provided to ensure that the hardware has enough flexibility to generate the right 525/60 or 625/50 data streams. Most of the initialization steps in Initializing the eLCDIF such as data shifting, swizzle, and so on, are applicable to DVI mode also. The register descriptions in the programmable registers section at the end of this chapter include example code for programming the DVICTRL0-3 registers.

In DVI mode, LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DVI_MODE bit the value 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and assert the cur_frame_done interrupt.

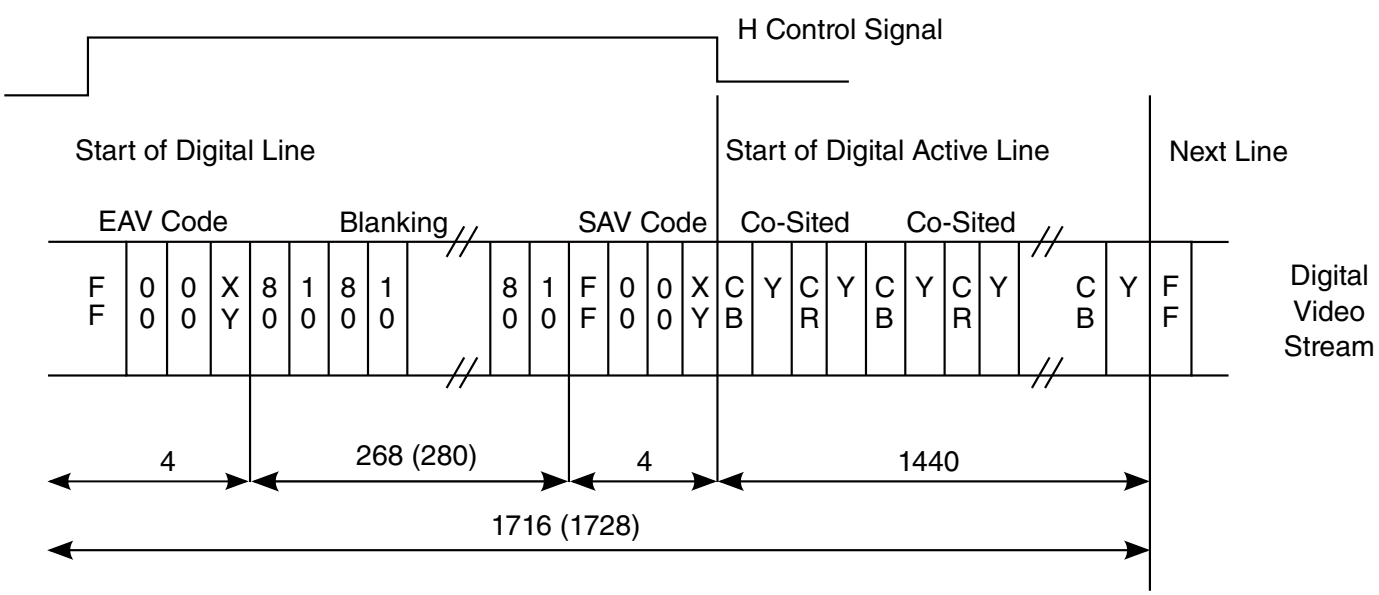


Figure 32-15. Digital Video Interface

32.4.12 eLCDIF Pin Usage by Interface Mode

Table 32-3 and Table 32-4 indicates how the eLCDIF level interface pins are used based on the desired mode of operation. The chip level I/Os should also be configured to be consistent with the desired eLCDIF operating mode.

The VSYNC signal has been mapped onto two pins, LCD_BUSY and LCD_VSYNC. The pin multiplexing can be programmed to select either of those pins to function as VSYN.

NOTE

There is an option to internally mux the HSYNC, DOTCLK and ENABLE signals in the DOTCLK mode by setting the MUX_SYNC_SIGNALS bit in the VDCTRL0 register. There is also an option to internally mux the LCD_WR_RWn and LCD_RD_E pins in the CTRL1 register for backward compatibility.

Table 32-3. Pin use in MPU Mode and VSYNC Mode

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_RS	LCD_ RS	LCD_ RS	LCD_ RS	LCD_RS	LCD_RS	LCD_ RS	LCD_RS	LCD_RS
LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS
LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn
LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E
LCD_VSYNC * (Two options)	X	X	X	X	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC
LCD_HSYNC	X	X	X	X	X	X	X	X
LCD_DOTCLK	X	X	X	X	X	X	X	X
LCD_ENABLE	X	X	X	X	X	X	X	X
LCD_D23	X	X	X	LCD_D23	X	X	X	LCD_D23
LCD_D22	X	X	X	LCD_D22	X	X	X	LCD_D22
LCD_D21	X	X	X	LCD_D21	X	X	X	LCD_D21
LCD_D20	X	X	X	LCD_D20	X	X	X	LCD_D20
LCD_D19	X	X	X	LCD_D19	X	X	X	LCD_D19
LCD_D18	X	X	X	LCD_D18	X	X	X	LCD_D18

Table continues on the next page...

Table 32-3. Pin use in MPU Mode and VSYNC Mode (continued)

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_D17	X	X	LCD_D17	LCD_D17	X	X	LCD_D17	LCD_D17
LCD_D16	X	X	LCD_D16	LCD_D16	X	X	LCD_D16	LCD_D16
LCD_D15 / VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	VSYNC (optional)	LCD_D15	VSYNC (optional)	LCD_D15
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X	LCD_D14	X	LCD_D14
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X	LCD_D13	X	LCD_D13
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X	LCD_D12	X	LCD_D12
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X	LCD_D11	X	LCD_D11
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X	LCD_D10	X	LCD_D10
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X	LCD_D9	X	LCD_D9
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X	LCD_D8	X	LCD_D8
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET
LCD_BUSY / LCD_VSYNC	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)

Table 32-4. Pin use in DOTCLK Mode and DVI Mode

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_RS	X	X	X	X	CCIR_CLK
LCD_CS	X	X	X	X	X
LCD_WR_RWn	X	X	X	X	X
LCD_RD_E	X	X	X	X	X

Table continues on the next page...

Table 32-4. Pin use in DOTCLK Mode and DVI Mode (continued)

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_VSYNC* (Two options)	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	X
LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	X
LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	X
LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	X
LCD_D23	X	X	X	LCD_D23	X
LCD_D22	X	X	X	LCD_D22	X
LCD_D21	X	X	X	LCD_D21	X
LCD_D20	X	X	X	LCD_D20	X
LCD_D19	X	X	X	LCD_D19	X
LCD_D18	X	X	X	LCD_D18	X
LCD_D17	X	X	LCD_D17	LCD_D17	X
LCD_D16	X	X	LCD_D16	LCD_D16	X
LCD_D15/ VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	X
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	X
LCD_BUSY / LCD_VSYNC	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	X

32.5 Behavior During Reset

BUS_CLK and DISPLAY_CLK must be running before making any changes to SFTRST or CLKGATE bits.

A soft reset (SFTRST) can take multiple clock periods to complete, so do not set CLKGATE when setting SFTRST.

The reset process gates the clocks automatically.

32.6 ELCDIF Memory Map/Register Definition

eLCDIF Hardware Register Format Summary

LCDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_8000	eLCDIF General Control Register (LCDIF_RL)	32	R/W	C000_0000h	32.6.1/1960
21C_8004	eLCDIF General Control Register (LCDIF_RL_SET)	32	R/W	C000_0000h	32.6.1/1960
21C_8008	eLCDIF General Control Register (LCDIF_RL_CLR)	32	R/W	C000_0000h	32.6.1/1960
21C_800C	eLCDIF General Control Register (LCDIF_RL_TOG)	32	R/W	C000_0000h	32.6.1/1960
21C_8010	eLCDIF General Control1 Register (LCDIF_CTRL1)	32	R/W	000F_0000h	32.6.2/1963
21C_8014	eLCDIF General Control1 Register (LCDIF_CTRL1_SET)	32	R/W	000F_0000h	32.6.2/1963
21C_8018	eLCDIF General Control1 Register (LCDIF_CTRL1_CLR)	32	R/W	000F_0000h	32.6.2/1963
21C_801C	eLCDIF General Control1 Register (LCDIF_CTRL1_TOG)	32	R/W	000F_0000h	32.6.2/1963
21C_8020	eLCDIF General Control2 Register (LCDIF_CTRL2)	32	R/W	0020_0000h	32.6.3/1965
21C_8024	eLCDIF General Control2 Register (LCDIF_CTRL2_SET)	32	R/W	0020_0000h	32.6.3/1965
21C_8028	eLCDIF General Control2 Register (LCDIF_CTRL2_CLR)	32	R/W	0020_0000h	32.6.3/1965
21C_802C	eLCDIF General Control2 Register (LCDIF_CTRL2_TOG)	32	R/W	0020_0000h	32.6.3/1965
21C_8030	eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT)	32	R/W	0001_0000h	32.6.4/1968
21C_8040	LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF)	32	R/W	0000_0000h	32.6.5/1968
21C_8050	LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF)	32	R/W	0000_0000h	32.6.6/1969
21C_8060	LCD Interface Timing Register (LCDIF_TIMING)	32	R/W	0000_0000h	32.6.7/1969
21C_8070	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0)	32	R/W	0000_0000h	32.6.8/1970

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_8074	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_SET)	32	R/W	0000_0000h	32.6.8/1970
21C_8078	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_CLR)	32	R/W	0000_0000h	32.6.8/1970
21C_807C	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_TOG)	32	R/W	0000_0000h	32.6.8/1970
21C_8080	eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1)	32	R/W	0000_0000h	32.6.9/1972
21C_8090	LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2)	32	R/W	0000_0000h	32.6.10/ 1972
21C_80A0	eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3)	32	R/W	0000_0000h	32.6.11/ 1973
21C_80B0	eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4)	32	R/W	0000_0000h	32.6.12/ 1974
21C_80C0	Digital Video Interface Control0 Register (LCDIF_DVICTRL0)	32	R/W	0000_0000h	32.6.13/ 1975
21C_80D0	Digital Video Interface Control1 Register (LCDIF_DVICTRL1)	32	R/W	0000_0000h	32.6.14/ 1975
21C_80E0	Digital Video Interface Control2 Register (LCDIF_DVICTRL2)	32	R/W	0000_0000h	32.6.15/ 1976
21C_80F0	Digital Video Interface Control3 Register (LCDIF_DVICTRL3)	32	R/W	0000_0000h	32.6.16/ 1977
21C_8100	Digital Video Interface Control4 Register (LCDIF_DVICTRL4)	32	R/W	0000_0000h	32.6.17/ 1978
21C_8110	RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF_CSC_COEFF0)	32	R/W	0000_0000h	32.6.18/ 1979
21C_8120	RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF_CSC_COEFF1)	32	R/W	0000_0000h	32.6.19/ 1980
21C_8130	RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF_CSC_COEFF2)	32	R/W	0000_0000h	32.6.20/ 1981
21C_8140	RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF_CSC_COEFF3)	32	R/W	0000_0000h	32.6.21/ 1981
21C_8150	RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF_CSC_COEFF4)	32	R/W	0000_0000h	32.6.22/ 1982
21C_8160	RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF_CSC_OFFSET)	32	R/W	0080_0010h	32.6.23/ 1983
21C_8170	RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF_CSC_LIMIT)	32	R/W	00FF_00FFh	32.6.24/ 1983
21C_8180	LCD Interface Data Register (LCDIF_DATA)	32	R/W	0000_0000h	32.6.25/ 1984
21C_8190	Bus Master Error Status Register (LCDIF_BM_ERROR_STAT)	32	R/W	0000_0000h	32.6.26/ 1985
21C_81A0	CRC Status Register (LCDIF_CRC_STAT)	32	R/W	0000_0000h	32.6.27/ 1985

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_81B0	LCD Interface Status Register (LCDIF_STAT)	32	R	9500_0000h	32.6.28/ 1986
21C_81C0	LCD Interface Version Register (LCDIF_VERSION)	32	R	0400_0000h	32.6.29/ 1988
21C_81D0	LCD Interface Debug0 Register (LCDIF_DEBUG0)	32	R	0E81_0000h	32.6.30/ 1988
21C_81E0	LCD Interface Debug1 Register (LCDIF_DEBUG1)	32	R	0000_0000h	32.6.31/ 1991
21C_81F0	LCD Interface Debug2 Register (LCDIF_DEBUG2)	32	R	0000_0000h	32.6.32/ 1992
21C_8200	eLCDIF Threshold Register (LCDIF_THRES)	32	R/W	0100_000Fh	32.6.33/ 1992
21C_8210	eLCDIF AS Buffer Control Register (LCDIF_AS_CTRL)	32	R/W	0000_0000h	32.6.34/ 1994
21C_8220	Alpha Surface Buffer Pointer (LCDIF_AS_BUF)	32	R/W	0000_0000h	32.6.35/ 1996
21C_8230	LCDIF_AS_NEXT_BUF	32	R/W	0000_0000h	32.6.36/ 1997
21C_8240	eLCDIF Overlay Color Key Low (LCDIF_AS_CLRKEYLOW)	32	R/W	00FF_FFFFh	32.6.37/ 1997
21C_8250	eLCDIF Overlay Color Key High (LCDIF_AS_CLRKEYHIGH)	32	R/W	0000_0000h	32.6.38/ 1998
21C_8260	LCD working insync mode with CSI for VSYNC delay (LCDIF_SYNC_DELAY)	32	R/W	0000_0000h	32.6.39/ 1998
21C_8270	eLCDIF Interface Debug3 Register (LCDIF_DEBUG3)	32	R/W	0000_0000h	32.6.40/ 1999
21C_8280	LCD Interface Debug4 (LCDIF_DEBUG4)	32	R/W	0000_0000h	32.6.41/ 2000
21C_8290	LCD Interface Debug5 (LCDIF_DEBUG5)	32	R/W	0000_0000h	32.6.42/ 2001

32.6.1 eLCDIF General Control Register (LCDIF_RL*n*)

The LCD Interface Control Register provides overall control of the eLCDIF block. The eLCDIF Control Register provides a variety of control functions to the programmer. These functions allow the interface to be very flexible to work with a variety of LCD controllers, and to minimize overhead and increase performance of LCD programming. The register has been organized such that switching between the different LCD modes can be done with minimum PIO writes.

Address: 21C_8000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFTRST	CLKGATE	YCBCR422_INPUT	READ_WRITEB	WAIT_FOR_VSYNC_EDGE	DATA_SHIFT_DIR	SHIFT_NUM_BITS					DVI_MODE	BYPASS_COUNT	VSYNC_MODE	DOTCLK_MODE	DATA_SELECT
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_DATA_SWIZZLE		CSC_DATA_SWIZZLE		LCD_DATABUS_WIDTH		WORD_LENGTH		RGB_TO_YCBCR422_CSC	ENABLE_PXP_HANDSHAKE	MASTER	Reserved	DATA_FORMAT_16_BIT	DATA_FORMAT_18_BIT	DATA_FORMAT_24_BIT	RUN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_RL*n* field descriptions

Field	Description
31 SFTRST	This bit must be set to zero to enable normal operation of the eLCDIF. When set to one, it forces a block level reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 YCBCR422_INPUT	Zero implies input data is in RGB color space. One implies input data is in YCbCr 4:2:2 format, such that YCbYCr are packed in a 32-bit word. It also means that there are 2 pixels in 4 bytes. If this bit is set, software should program the H_COUNT field in the TRANSFER_COUNT register to the total number of pixels that will have to be fetched by the eLCDIF block per line and the BYTE_PACKING_FORMAT should be 0xF. The WORD_LENGTH does not matter in this case.

Table continues on the next page...

LCDIF_RL*n* field descriptions (continued)

Field	Description
28 READ_WRITEB	By default, eLCDIF is in the write mode. Setting this bit to 1 will make the hardware go into 6800/8080 MPU read mode. The LCDIF_MASTER bit must be 0, since bus master mode can only be used for writing the display.
27 WAIT_FOR_ VSYNC_EDGE	Setting this bit to 1 will make the hardware wait for the triggering VSYNC edge before starting write transfers to the LCD. Used only in the VSYNC mode of operation.
26 DATA_SHIFT_ DIR	Use this bit to determine the direction of shift of transmit data. In the DVI mode, it works only on the active data, not on the timing codes and ancillary data. 0x0 **TXDATA_SHIFT_LEFT** — Data to be transmitted is shifted LEFT by SHIFT_NUM_BITS bits. 0x1 **TXDATA_SHIFT_RIGHT** — Data to be transmitted is shifted RIGHT by SHIFT_NUM_BITS bits.
25–21 SHIFT_NUM_ BITS	The data to be transmitted is shifted left or right by this number of bits.
20 DVI_MODE	Set this bit to 1 to get into the ITU-R BT.656 digital video interface mode. Toggle this bit from 1 to 0 to make the hardware go out of DVI mode after completing all data transfer and after the RUN bit has been deasserted.
19 BYPASS_ COUNT	When this bit is 0, it means that eLCDIF will stop the block operation and turn off the RUN bit after the amount of data indicated by the LCDIF_TRANSFER_COUNT register has been transferred out. When this bit is set to 1, the block will continue normal operation indefinitely until it is told to stop. This bit must be 0 in MPU and VSYNC modes, and must be 1 in DOTCLK and DVI modes of operation.
18 VSYNC_MODE	Setting this bit to 1 will make the eLCDIF hardware go into VSYNC mode. WAIT_FOR_VSYNC_EDGE can be used only if this bit is set. If VSYNC signal is required to be an output from the block, SYNC_SIGNALS_ON bit in LCDIF_VDCTRL4 register must be set.
17 DOTCLK_MODE	Set this bit to 1 to make the hardware go into the DOTCLK mode, i.e. VSYNC/HSYNC/DOTCLK/ENABLE interface mode. ENABLE is optional, selected by the ENABLE_PRESENT bit. Toggle this bit from 1 to 0 to make the hardware go out of DOTCLK mode after completing all data transfer and deasserting the RUN bit.
16 DATA_SELECT	Command Mode polarity bit. This bit should only be changed when RUN is 0. 0x0 **CMD_MODE** — Command Mode. DCn signal is Low. 0x1 **DATA_MODE** — Data Mode. DCn signal is High.
15–14 INPUT_DATA_ SWIZZLE	This field specifies how to swap the bytes fetched by the bus master interface. The swizzle function is independent of the WORD_LENGTH bit. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
13–12 CSC_DATA_ SWIZZLE	This field specifies how to swap the bytes after the data has been converted into an internal representation of 24 bits per pixel and before it is transmitted over the LCD interface bus. The data is always transmitted with the least significant byte/hword (half word) first after the swizzle takes place. So, INPUT_DATA_SWIZZLE takes place first on the incoming data, and then CSC_DATA_SWIZZLE is applied. The swizzle function is independent of the WORD_LENGTH or the LCD_DATABUS_WIDTH fields. If RGB_TO_YCRCB422_CSC bit is set, the swizzle occurs on the Y, Cb, Cr values. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian)

Table continues on the next page...

LCDIF_RL*n* field descriptions (continued)

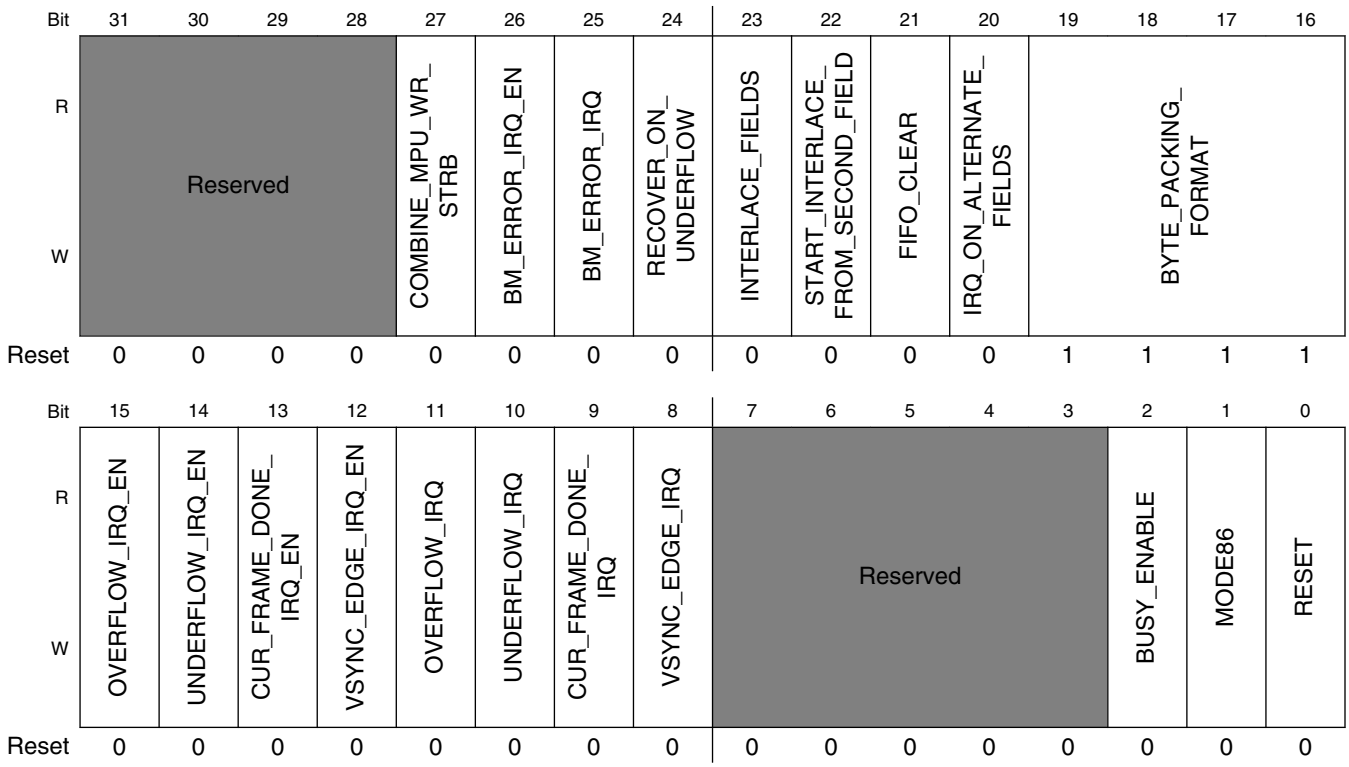
Field	Description
	0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
11–10 LCD_DATABUS_WIDTH	LCD Data bus transfer width. 0x0 **16_BIT** — 16-bit data bus mode. 0x1 **8_BIT** — 8-bit data bus mode. 0x2 **18_BIT** — 18-bit data bus mode. 0x3 **24_BIT** — 24-bit data bus mode.
9–8 WORD_LENGTH	Input data format. 0x0 **16_BIT** — Input data is 16 bits per pixel. 0x1 **8_BIT** — Input data is 8 bits wide. 0x2 **18_BIT** — Input data is 18 bits per pixel. 0x3 **24_BIT** — Input data is 24 bits per pixel.
7 RGB_TO_YCBCR422_CSC	Set this bit to 1 to enable conversion from RGB to YCbCr colorspace. See the LCDIF_CSC_ registers for further details.
6 ENABLE_PXP_HANDSHAKE	If this bit is set and LCDIF_MASTER bit is set, the eLCDIF will act as bus master and the handshake mechanism between eLCDIF and PXP will be turned on. If LCDIF_MASTER bit is not set, this bit becomes a don't care.
5 MASTER	Set this bit to make the eLCDIF act as a bus master. If this bit is reset, the eLCDIF will support or PIO mode.
4 RSRVD0	This field is reserved. Reserved bits. Write as 0.
3 DATA_FORMAT_16_BIT	When this bit is 1 and WORD_LENGTH = 0, it implies that the 16-bit data is in ARGB555 format. When this bit is 0 and WORD_LENGTH = 0, it implies that the 16-bit data is in RGB565 format. When WORD_LENGTH is not 0, this bit does not care.
2 DATA_FORMAT_18_BIT	Used only when WORD_LENGTH = 2, i.e. 18-bit. 0x0 **LOWER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that lower 18 bits contain RGB 666 and upper 14 bits do not contain any useful data. 0x1 **UPPER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that upper 18 bits contain RGB 666 and lower 14 bits do not contain any useful data.
1 DATA_FORMAT_24_BIT	Used only when WORD_LENGTH = 3, i.e. 24-bit. Note that this applies to both packed and unpacked 24-bit data. 0x0 **ALL_24_BITS_VALID** — Data input to the block is in 24 bpp format, such that all RGB 888 data is contained in 24 bits. 0x1 **DROP_UPPER_2_BITS_PER_BYTE** — Data input to the block is actually RGB 18 bpp, but there is 1 color per byte, hence the upper 2 bits in each byte do not contain any useful data, and should be dropped.
0 RUN	When this bit is set by software, the eLCDIF will begin transferring data between the SoC and the display. This bit must remain set until the operation is complete.

32.6.2 eLCDIF General Control1 Register (LCDIF_CTRL1*n*)

The eLCDIF Control Register provides overall control of the eLCDIF block.

The eLCDIF Control1 Register provides additional programming to the eLCDIF. It implements some bits which are unlikely to change often in a particular application. It also carries interrupt-related bits which are common across more than one mode of operation.

Address: 21C_8000h base + 10h offset + (4d × i), where i=0d to 3d



LCDIF_CTRL1*n* field descriptions

Field	Description
31–28 RSRVD1	This field is reserved. Reserved bits. Write as 0.
27 COMBINE_MPU_WR_STRB	If this bit is not set, the write strobe will be driven on LCD_WR_RWn pin in the 8080 mode and on the LCD_RD_E pin in the 6800 mode. If it is set, the write strobe of both the 6800 and 8080 modes will be driven only on the LCD_WR_RWn pin. Note that this does not work for read strobe.
26 BM_ERROR_IRQ_EN	This bit is set to enable bus master error interrupt in the eLCDIF master mode.
25 BM_ERROR_IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. This bit will be set when the eLCDIF is in master mode and an error response was returned by the slave.

Table continues on the next page...

LCDIF_CTRL1*n* field descriptions (continued)

Field	Description
	0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
24 RECOVER_ON_ UNDERFLOW	Set this bit to enable the eLCDIF block to recover in the next field/frame if there was an underflow in the current field/frame.
23 INTERLACE_ FIELDS	Set this bit if it is required that the eLCDIF block fetches odd lines in one field and even lines in the other field. It will work only in LCDIF_MASTER is set to 1.
22 START_ INTERLACE_ FROM_ SECOND_FIELD	The default is to grab the odd lines first and then the even lines. Set this bit if it is required to grab the even lines first and then the odd lines. (Line numbers start from 1, so odd lines are 1,3,5,etc. and even lines are 2,4,6, etc.)
21 FIFO_CLEAR	Set this bit to clear all the data in the latency FIFO (LFIFO), TXFIFO and the RXFIFO.
20 IRQ_ON_ ALTERNATE_ FIELDS	If this bit is set, the eLCDIF block will assert the cur_frame_done interrupt only on alternate fields, otherwise it will issue the interrupt on both odd and even field. This bit is mostly relevant if INTERLACE_FIELDS is set. This feature is only available in DOTCLK and DVI modes.
19–16 BYTE_ PACKING_ FORMAT	This bitfield is used to show which data bytes in a 32-bit word are valid. Default value 0xf indicates that all bytes are valid. For 8-bit transfers, any combination in this bitfield will mean valid data is present in the corresponding bytes. In the 16-bit mode, a 16-bit half-word is valid only if adjacent bits [1:0] or [3:2] or both are 1. A value of 0x0 will mean that none of the bytes are valid and should not be used. For example, set the bit field value to 0x7 if the display data is arranged in the 24-bit unpacked format (A-R-G-B where A value does not have be transmitted). When input data is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of pixels that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF.(Note - YCBCR422_INPUT = 1 implies 2 pixels per 32 bits).
15 OVERFLOW_ IRQ_EN	This bit is set to enable an overflow interrupt in the TXFIFO in the write mode.
14 UNDERFLOW_ IRQ_EN	This bit is set to enable an underflow interrupt in the TXFIFO in the write mode.
13 CUR_FRAME_ DONE_IRQ_EN	This bit is set to 1 enable an interrupt every time the hardware enters in the vertical blanking state.
12 VSYNC_EDGE_ IRQ_EN	This bit is set to enable an interrupt every time the hardware encounters the leading VSYNC edge in the VSYNC and DOTCLK modes, or the beginning of every field in DVI mode.
11 OVERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A latency FIFO (LFIFO) overflow in the write mode (MPU/VSYNC/DOTCLK/DVI mode) was detected, data samples have been lost. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
10 UNDERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A TXFIFO underflow in the write mode (MPU/VSYNC/DOTCLK/DVI mode) was detected. Could produce an error in the DOTCLK / DVI modes.

Table continues on the next page...

LCDIF_CTRL1*n* field descriptions (continued)

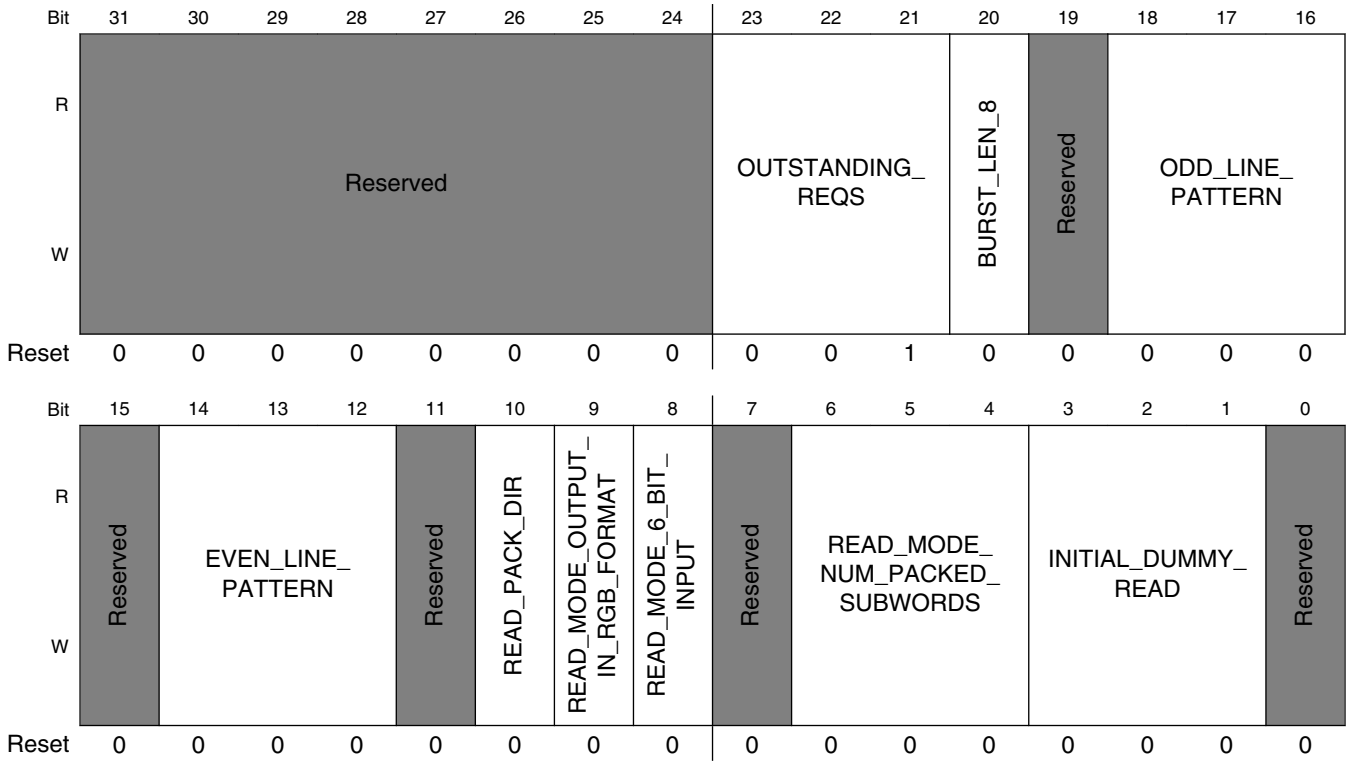
Field	Description
	0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
9 CUR_FRAME_DONE_IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It indicates that the hardware has completed transmitting the current frame and is in the vertical blanking period in the DOTCLK/DVI modes. In the MPU and VSYNC modes, this IRQ is asserted at the end of the data transfer indicated by LCDIF_TRANSFER_COUNT register. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
8 VSYNC_EDGE_IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It is set whenever the leading VSYNC edge is detected in the VSYNC and DOTCLK modes. In the DVI mode, it is asserted every time the block enters a new field. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
7–3 RSRVD0	This field is reserved. Reserved bits. Write as 0.
2 BUSY_ENABLE	This bit enables the use of the interface's busy signal input. This should be enabled for LCD controllers that implement a busy line (to stall the eLCDIF from sending more data until ready). Otherwise this bit should be cleared. 0x0 **BUSY_DISABLED** — The busy signal from the LCD controller will be ignored. 0x1 **BUSY_ENABLED** — Enable the use of the busy signal from the LCD controller.
1 MODE86	This bit is used to select between the 8080 and 6800 series of microprocessor modes. This bit should only be changed when RUN is 0. 0x0 **8080_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as active low WR and active low RD signals respectively. 0x1 **6800_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as Read/Write and active high Enable signals respectively.
0 RESET	Reset bit for the external LCD controller. This bit can be changed at any time. It CANNOT be reset by SFTRST. 0x0 **LCDRESET_LOW** — LCD_RESET output signal is low. 0x1 **LCDRESET_HIGH** — LCD_RESET output signal is high.

32.6.3 eLCDIF General Control2 Register (LCDIF_CTRL2*n*)

The eLCDIF Control Register provides overall control of the eLCDIF block.

The eLCDIF Control2 Register provides additional programming to the eLCDIF. It implements some bits which are unlikely to change often in a particular application.

Address: 21C_8000h base + 20h offset + (4d × i), where i=0d to 3d



LCDIF_CTRL2*n* field descriptions

Field	Description
31–24 RSRVD5	This field is reserved. Reserved bits. Write as 0.
23–21 OUTSTANDING_ REQS	This bitfield indicates the maximum number of outstanding transactions that eLCDIF should request when it is acting as a bus master. Default is 2 outstanding transactions. 0x0 **REQ_1** — 0x1 **REQ_2** — 0x2 **REQ_4** — 0x3 **REQ_8** — 0x4 **REQ_16** —
20 BURST_LEN_8	By default, when the eLCDIF is in the bus master mode, it will issue AXI bursts of length 16 (except when in packed 24 bpp mode, it will issue bursts of length 15). When this bit is set to 1, the block will issue bursts of length 8 (except when in packed 24 bpp mode, it will issue bursts of length 9). Note that this bitfield is only applicable when LCDIF_MASTER is set to 1.
19 RSRVD4	This field is reserved. Reserved bits. Write as 0.
18–16 ODD_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in ODD lines (line numbers 1,3,5,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** —

Table continues on the next page...

LCDIF_CTRL2*n* field descriptions (continued)

Field	Description
	0x4 **BRG** — 0x5 **BGR** —
15 RSRVD3	This field is reserved. Reserved bits. Write as 0.
14–12 EVEN_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in EVEN lines (line numbers 2,4,6,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —
11 RSRVD2	This field is reserved. Reserved bits. Write as 0.
10 READ_PACK_DIR	The default value of 0 indicates data is stored in the little endian format. When LCD_DATABUS_WIDTH is 8-bit, this bit provides the option of rearranging the data byte-wise in the big endian format. For example, if READ_MODE_NUM_PACKED_SUBWORDS = 3 and the order of incoming data is 0x11, 0x22 and 0x33, then setting this bit to 1 will cause the data to be stored as 0x00112233 as opposed to the default 0x00332211. This operation occurs after the shifting operation done by SHIFT_NUM_BITS bitfield.
9 READ_MODE_ OUTPUT_IN_ RGB_FORMAT	Setting this bit will enable the eLCDIF to convert the incoming data to the RGB format given by WORD_LENGTH bitfield. This feature is not available when WORD_LENGTH is set to 8 bits. eLCDIF performs this operation of converting to RGB format after the endianness has been determined by the READ_PACK_DIR bitfield.
8 READ_MODE_6_ BIT_INPUT	Setting this bit to 1 indicates to eLCDIF that even though LCD_DATABUS_WIDTH is set to 8 bits, the input data is actually only 6 bits wide and exists on D5-D0.
7 RSRVD1	This field is reserved. Reserved bits. Write as 0.
6–4 READ_MODE_ NUM_PACKED_ SUBWORDS	Indicates the number of valid 8/16/18/24-bit subwords that will be packed into the 32-bit word in read mode. The subword size (8,16, 18 or 24 bits) is determined by the LCD_DATABUS_WIDTH field. The swizzle operation is performed after READ_MODE_NUM_PACKED_SUBWORDS number of subwords has been received and stored in little-endian format. For example, if LCD_DATABUS_WIDTH is set to 8-bit and data to be read back has to be stored in memory in 24-bit unpacked RGB format, set READ_MODE_NUM_PACKED_SUBWORDS to 0x3 so that each 32-bit word will contain only 3 valid bytes (RGB). Maximum value of READ_MODE_NUM_PACKED_SUBWORDS is 4 for 8-bit databus, 2 for 16-bit databus and 1 for 18/24-bit databus.
3–1 INITIAL_DUMMY_ READ	The value in this field determines the number of dummy 8/16/18/24-bit subwords that have to be read back from the LCD panel/controller. They will then not be stored in the read FIFO.
0 RSRVD0	This field is reserved. Reserved bits. Write as 0.

32.6.4 eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT)

This register tells the eLCDIF how much data will be sent for this frame, or transaction. The total number of words is a product of the V_COUNT and H_COUNT fields. The word size is specified by the WORD_LENGTH field.

This register gives the dimensions of the input frame. For normal operation, but V_COUNT and H_COUNT should be non-zero.

Address: 21C_8000h base + 30h offset = 21C_8030h

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

LCDIF_TRANSFER_COUNT field descriptions

Field	Description
31–16 V_COUNT	Number of horizontal lines per frame which contain valid data. In DOTCLK mode, V_COUNT should be the same as the number of active horizontal lines in a progressive frame. In DVI mode, V_COUNT should be the number of active horizontal lines per frame, and not per field.
H_COUNT	Total valid data (pixels) in each horizontal line. The data size is given by the WORD_LENGTH. When input data is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of 32-bit words that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF. In 24-bit packed format (WORD_LENGTH=0x3, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 4 pixels. In 16-bit packed format (WORD_LENGTH=0x0, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 2 pixels.

32.6.5 LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF)

This register indicates the address of the current frame being transmitted by eLCDIF.

When the eLCDIF is behaving as a master, this address points to the address of the current frame of data being sent out via the LCDIF. When the current frame is done, the LCDIF block will assert the cur_frame_done interrupt for software to take action. The block will also copy the LCDIF_NEXT_BUF_ADDR into this bitfield so that the software can program the next frame address into the LCDIF_NEXT_BUF_ADDR bitfield. This address must always be double-word aligned.

Address: 21C_8000h base + 40h offset = 21C_8040h

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

LCDIF_CUR_BUF field descriptions

Field	Description
ADDR	Address of the current frame being transmitted by eLCDIF.

32.6.6 LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF)

This register indicates the address of next frame that will be transmitted by eLCDIF.

When the eLCDIF is behaving as a master, this address points to the address of the next frame of data that will be sent out via the eLCDIF. It is up to the software to make sure that this register is programmed before the end of the current frame, otherwise it might result in old data going out the eLCDIF. This address must always be double-word aligned.

Address: 21C_8000h base + 50h offset = 21C_8050h

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

LCDIF_NEXT_BUF field descriptions

Field	Description
ADDR	Address of the next frame that will be transmitted by eLCDIF.

32.6.7 LCD Interface Timing Register (LCDIF_TIMING)

The LCD interface timing register controls the various setup and hold times enforced by the LCD interface in the 6800/8080 MPU and VSYNC modes of operation.

The values used in this register are dependent on the particular LCD controller used, consult the users manual for the particular controller for required timings. Each field of the register must be non-zero, therefore the minimum value is: 0x01010101. NOTE: the timings are not automatically adjusted if the CLK_DIS_LCDIFn frequency changes—it

may be necessary to adjust the timings if CLK_DIS_LCDIFn changes. NOTE: Each field in this register must be non-zero for the MPU and VSYNC modes to function. The settings in this register do not affect the DOTCLK and DVI modes.

Address: 21C_8000h base + 60h offset = 21C_8060h

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

LCDIF_TIMING field descriptions

Field	Description
31–24 CMD_HOLD	Number of CLK_DIS_LCDIFn cycles that the DCn signal is active after CEn is deasserted.
23–16 CMD_SETUP	Number of CLK_DIS_LCDIFn cycles that the DCn signal is active before CEn is asserted.
15–8 DATA_HOLD	Data bus hold time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is deasserted in a cycle
DATA_SETUP	Data bus setup time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is asserted in a cycle.

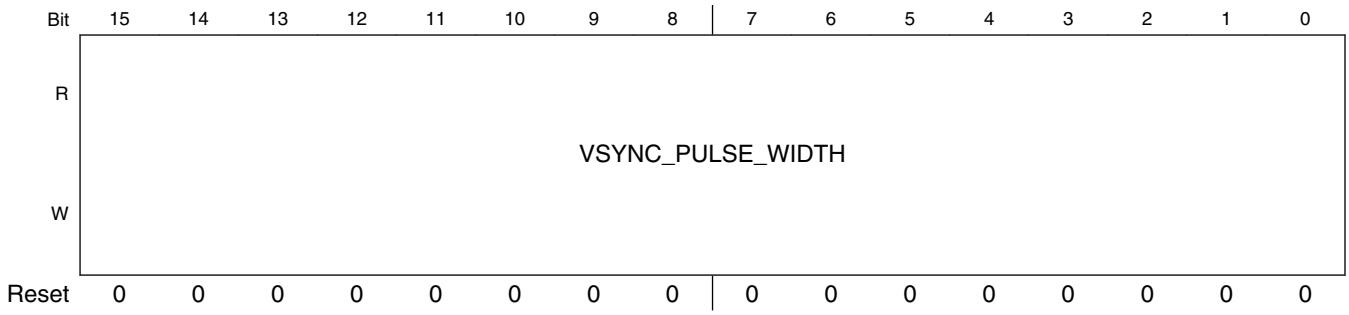
32.6.8 eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0*n*)

This register is used to control the VSYNC and DOTCLK modes of the LCDIF so as to work with different types of LCDs like moving picture displays and delta pixel displays.

This register gives general programmability to the VSYNC signal including polarity, direction, pulse width, etc.

Address: 21C_8000h base + 70h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		VSYNC_OEB	ENABLE_PRESENT	VSYNC_POL	HSYNC_POL	DOTCLK_POL	ENABLE_POL	Reserved		VSYNC_PERIOD_UNIT	VSYNC_PULSE_WIDTH_UNIT	HALF_LINE	HALF_LINE_MODE	VSYNC_PULSE_WIDTH	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



LCDIF_VDCTRL0*n* field descriptions

Field	Description
31–30 RSRVD2	This field is reserved. Reserved bits. Write as 0.
29 VSYNC_OEB	0 means the VSYNC signal is an output, 1 means it is an input. Should be set to 0 in the DOTCLK mode. 0x0 **VSYNC_OUTPUT** — The VSYNC pin is in the output mode and the VSYNC signal has to be generated by the eLCDIF block. 0x1 **VSYNC_INPUT** — The VSYNC pin is in the input mode and the LCD controller sends the VSYNC signal to the block.
28 ENABLE_ PRESENT	Setting this bit to 1 will make the hardware generate the ENABLE signal in the DOTCLK mode, thereby making it the true RGB interface along with the remaining three signals VSYNC, HSYNC and DOTCLK.
27 VSYNC_POL	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.
26 HSYNC_POL	Default 0 active low during HSYNC_PULSE_WIDTH time and will be high during the rest of the HSYNC period. Set it to 1 to invert the polarity.
25 DOTCLK_POL	Default is data launched at negative edge of DOTCLK and captured at positive edge. Set it to 1 to invert the polarity. Set it to 0 in DVI mode.
24 ENABLE_POL	Default 0 active low during valid data transfer on each horizontal line.
23–22 RSRVD1	This field is reserved. Reserved bits. Write as 0.
21 VSYNC_ PERIOD_UNIT	Default 0 for counting VSYNC_PERIOD in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines. CLK_DIS_LCDIFn cycles should be used in the VSYNC mode, while horizontal line should be used in the DOTCLK mode.
20 VSYNC_PULSE_ WIDTH_UNIT	Default 0 for counting VSYNC_PULSE_WIDTH in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines.
19 HALF_LINE	Setting this bit to 1 will make the total VSYNC period equal to the VSYNC_PERIOD field plus half the HORIZONTAL_PERIOD field (i.e. VSYNC_PERIOD field plus half horizontal line), otherwise it is just VSYNC_PERIOD. Should be only used in the DOTCLK mode, not in the VSYNC interface mode.
18 HALF_LINE_ MODE	When this bit is 0, the first field (VSYNC period) will end in half a horizontal line and the second field will begin with half a horizontal line. When this bit is 1, all fields will end with half a horizontal line, and none will begin with half a horizontal line.
VSYNC_PULSE_ WIDTH	Number of units for which VSYNC signal is active. For the DOTCLK mode, the unit is determined by the VSYNC_PULSE_WIDTH_UNIT. If the VSYNC_PULSE_WIDTH_UNIT is 0 for DOTCLK mode, VSYNC_PULSE_WIDTH must be less than HSYNC_PERIOD. For the VSYNC interface mode, it should be in terms of number of CLK_DIS_LCDIFn cycles only.

32.6.9 eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1)

This register is used to control the VSYNC signal in the VSYNC and DOTCLK modes of the block.

This register determines the period and duty cycle of the VSYNC signal when it is generated in the block.

Address: 21C_8000h base + 80h offset = 21C_8080h

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

LCDIF_VDCTRL1 field descriptions

Field	Description
VSYNC_PERIOD	Total number of units between two positive or two negative edges of the VSYNC signal. If HALF_LINE is set, it is implicitly calculated to be VSYNC_PERIOD plus half HSYNC_PERIOD.

32.6.10 LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2)

This register is used to control the HSYNC signal in the DOTCLK mode of the block.

This register determines the period and duty cycle of the HSYNC signal when it is generated in the block.

Address: 21C_8000h base + 90h offset = 21C_8090h

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

LCDIF_VDCTRL2 field descriptions

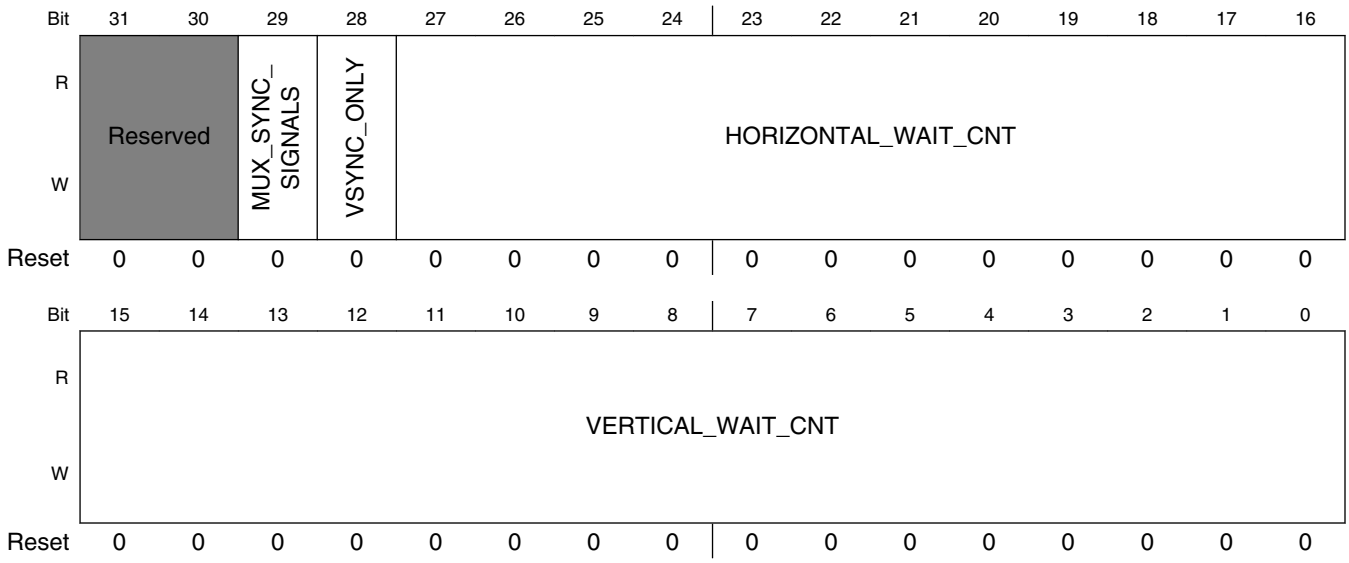
Field	Description
31–18 HSYNC_PULSE_WIDTH	Number of CLK_DIS_LCDIFn cycles for which HSYNC signal is active.
HSYNC_PERIOD	Total number of CLK_DIS_LCDIFn cycles between two positive or two negative edges of the HSYNC signal.

32.6.11 eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3)

This register is used to determine the vertical and horizontal wait counts.

This register determines the back porches of HSYNC and VSYNC signals when they are generated by the block.

Address: 21C_8000h base + A0h offset = 21C_80A0h



LCDIF_VDCTRL3 field descriptions

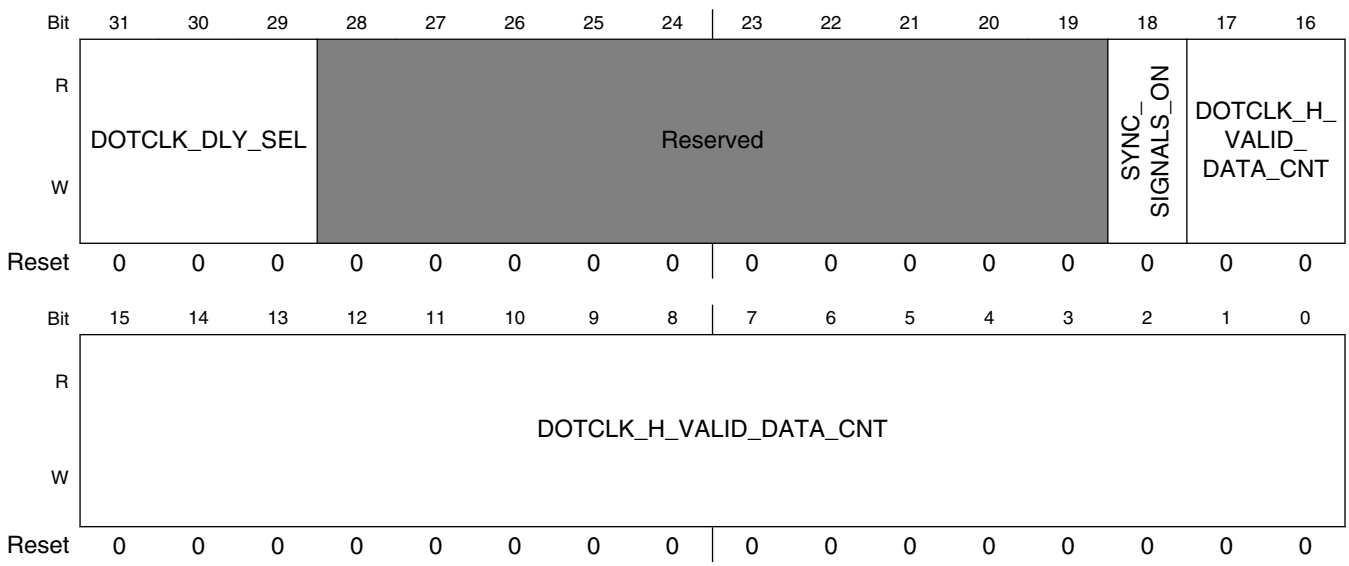
Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29 MUX_SYNC_SIGNALS	When this bit is set, the eLCDIF block will internally mux HSYNC with LCD_D14, DOTCLK with LCD_D13 and ENABLE with LCD_D12, otherwise these signals will go out on separate pins. This feature can be used to maintain backward compatible with 37xx.
28 VSYNC_ONLY	This bit must be set to 1 in the VSYNC mode of operation, and 0 in the DOTCLK mode of operation.
27–16 HORIZONTAL_WAIT_CNT	In the DOTCLK mode, wait for this number of clocks from falling edge (or rising if HSYNC_POL is 1) of HSYNC signal to account for horizontal back porch plus the number of DOTCLKs before the moving picture information begins.
VERTICAL_WAIT_CNT	In the VSYNC interface mode, wait for this number of CLK_DIS_LCDIFn cycles from the falling VSYNC edge (or rising if VSYNC_POL is 1) before starting LCD transactions and is applicable only if WAIT_FOR_VSYNC_EDGE is set. Minimum is CMD_SETUP+5. In the DOTCLK mode, it accounts for the vertical back porch lines plus the number of horizontal lines before the moving picture begins. The unit for this parameter is inherently the same as the VSYNC_PERIOD_UNIT.

32.6.12 eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4)

This register is used to control the DOTCLK mode of the block.

This register determines the active data in each horizontal line in the DOTCLK mode. Note that the total number of active horizontal lines in the DOTCLK mode is the same as the V_COUNT bitfield in the LCDIF_TRANSFER_COUNT register.

Address: 21C_8000h base + B0h offset = 21C_80B0h



LCDIF_VDCTRL4 field descriptions

Field	Description
31–29 DOTCLK_DLY_ SEL	This bitfield selects the amount of time by which the DOTCLK signal should be delayed before coming out of the LCD_DOTCK pin. 0 = 2ns; 1=4ns;2=6ns;3=8ns. Remaining values are reserved.
28–19 RSRVD0	This field is reserved. Reserved bits, write as 0.
18 SYNC_ SIGNALS_ON	Set this field to 1 if the LCD controller requires that the VSYNC or VSYNC/HSYNC/DOTCLK control signals should be active at least one frame before the data transfers actually start and remain active at least one frame after the data transfers end. The hardware does not count the number of frames automatically. Rather, the VSYNC edge interrupt can be monitored by software to count the number of frames that have occurred after this bit is set and then the RUN bit can be set to start the data transactions. This bit must always be set in the DOTCLK mode of operation, and it must be set in the VSYNC mode of operation when VSYNC signal is an output.
DOTCLK_H_ VALID_DATA_ CNT	Total number of CLK_DIS_LCDIFn cycles on each horizontal line that carry valid data in DOTCLK mode.

32.6.13 Digital Video Interface Control0 Register (LCDIF_DVICTRL0)

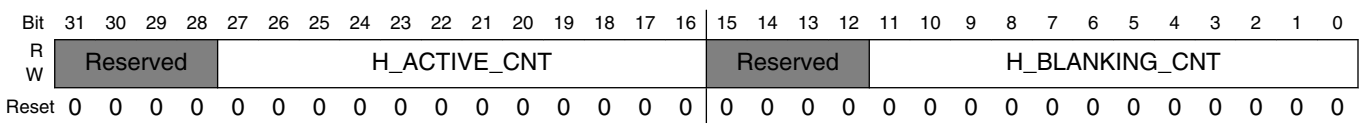
The Digital Video interface Control0 register provides the overall control of the Digital Video interface.

This register gives information about the horizontal active, horizontal blanking and total number of lines in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x106);//262
//625/50 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x112);//274
```

Address: 21C_8000h base + C0h offset = 21C_80C0h



LCDIF_DVICTRL0 field descriptions

Field	Description
31–28 RSRVD1	This field is reserved. Reserved bits, write as 0.
27–16 H_ACTIVE_CNT	Number of active video samples to be transmitted. (Mostly will be 1440 for both PAL and NTSC). Must always be a multiple of 4.
15–12 RSRVD0	This field is reserved. Reserved bits, write as 0.
H_BLANKING_ CNT	Number of blanking samples to be inserted between EAV and SAV during horizontal blanking interval.

32.6.14 Digital Video Interface Control1 Register (LCDIF_DVICTRL1)

The Digital Video interface Control1 register provides the overall control of the Digital Video interface.

This register contains information about the Field1 start and end, and the Field2 start in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x4);//4
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x109);//265
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x10A);//266
//625/50 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x138);//312
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x139);//313
```

Address: 21C_8000h base + D0h offset = 21C_80D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		F1_START_LINE										F1_END_LINE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F1_END_LINE								F2_START_LINE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DVICTRL1 field descriptions

Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29–20 F1_START_LINE	Vertical line number from which Field 1 begins.
19–10 F1_END_LINE	Vertical line number at which Field1 ends.
F2_START_LINE	Vertical line number from which Field 2 begins.

32.6.15 Digital Video Interface Control2 Register (LCDIF_DVICTRL2)

The Digital Video interface Control2 register provides the overall control of the Digital Video interface.

This register contains information about the Field2 end, and the Vertical Blanking1 interval in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x3);//3
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x108);//264
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x11A);//282
//625/50 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x271);//625
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x137);//311
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x14F);//335
```

Address: 21C_8000h base + E0h offset = 21C_80E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		F2_END_LINE										V1_BLANK_START_LINE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V1_BLANK_START_LINE						V1_BLANK_END_LINE									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DVICTRL2 field descriptions

Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29–20 F2_END_LINE	Vertical line number at which Field 2 ends.
19–10 V1_BLANK_ START_LINE	Vertical line number towards the end of Field1 where first Vertical Blanking interval starts.
V1_BLANK_ END_LINE	Vertical line number in the beginning part of Field2 where first Vertical Blanking interval ends.

32.6.16 Digital Video Interface Control3 Register (LCDIF_DVICTRL3)

The Digital Video interface Control3 register provides the overall control of the Digital Video interface.

This register contains information about the Vertical Blanking2 interval in the ITU-R BT. 656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x13);//19
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x20D);//525
//625/50 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x270);//624
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x16);//22
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x271);//625
```

Address: 21C_8000h base + F0h offset = 21C_80F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		V2_BLANK_START_LINE										V2_BLANK_END_LINE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V2_BLANK_END_LINE								V_LINES_CNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DVICTRL3 field descriptions

Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29–20 V2_BLANK_ START_LINE	Vertical line number towards the end of Field2 where second Vertical Blanking interval starts.
19–10 V2_BLANK_ END_LINE	Vertical line number in the beginning part of Field1 where second Vertical Blanking interval ends.
V_LINES_CNT	Total number of vertical lines per frame (generally 525 or 625)

32.6.17 Digital Video Interface Control4 Register (LCDIF_DVICTRL4)

The Digital Video interface Control4 register provides the overall control of the Digital Video interface.

This register is used to add side borders to the output if the input frame width is less than 720 pixels.

EXAMPLE

```
                //If input frame has only 640 pixels per line, but output is supposed to have
720 pixels per line.
                HW_LCDIF_DVICTRL4_H_FILL_CNT_WR(0x50);//80
                HW_LCDIF_DVICTRL4_Y_FILL_VALUE_WR(0x10);//16
                HW_LCDIF_DVICTRL4_CB_FILL_VALUE_WR(0x80);//128
                HW_LCDIF_DVICTRL4_CR_FILL_VALUE_WR(0x80);//128
```

Address: 21C_8000h base + 100h offset = 21C_8100h

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

LCDIF_DVICTRL4 field descriptions

Field	Description
31–24 Y_FILL_VALUE	Value of Y component of filler data

Table continues on the next page...

LCDIF_DVICTRL4 field descriptions (continued)

Field	Description
23–16 CB_FILL_VALUE	Value of CB component of filler data
15–8 CR_FILL_VALUE	Value of CR component of filler data.
H_FILL_CNT	Number of active video samples that have to be filled with the filler data in the front and back portions of the active horizontal interval. Must be a multiple of 4. This field will have to be programmed if the input frame has less than 720 pixels per line.

32.6.18 RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF_CSC_COEFF0)

LCDIF_CSC_COEFF0 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF0_C0_WR(0x41);//0.257x256=65
HW_LCDIF_CSC_COEFF0_CSC_SUBSAMPLE_FILTER_WR(0x3);
```

Address: 21C_8000h base + 110h offset = 21C_8110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								C0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

LCDIF_CSC_COEFF0 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C0	Two's complement red multiplier coefficient for Y
15–2 RSRVD0	This field is reserved. Reserved bits, write as 0.

Table continues on the next page...

LCDIF_CSC_COEFF0 field descriptions (continued)

Field	Description
CSC_ SUBSAMPLE_ FILTER	This register describes the filtering and subsampling scheme to be performed on the chroma components in order to convert from YCbCr 4:4:4 to YCbCr 4:2:2 space. Note that the following descriptions apply individually to Cb and Cr. 0x0 **SAMPLE_AND_HOLD** — No filtering, simply keep every chroma value for samples numbered 2n and discard chroma values associated with all samples numbered 2n+1. 0x1 **RSRVD** — Reserved 0x2 **INTERSTITIAL** — Chroma samples numbered 2n and 2n+1 are averaged (weights 1/2, 1/2) and that chroma value replaces the two chroma values at 2n and 2n+1. This chroma now exists horizontally halfway between the two luma samples. 0x3 **COSITED** — Chroma samples numbered 2n-1, 2n, and 2n+1 are averaged (weights 1/4, 1/2, 1/4) and that chroma value exists at the same site as the luma sample numbered 2n and the chroma samples at 2n+1 are discarded.

32.6.19 RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF_CSC_COEFF1)

LCDIF_CSC_COEFF1 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF1_C1_WR(0x81);//0.504x256=129
HW_LCDIF_CSC_COEFF1_C2_WR(0x19);//0.098x256=25
```

Address: 21C_8000h base + 120h offset = 21C_8120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						C2										Reserved						C1									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CSC_COEFF1 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C2	Two's complement blue multiplier coefficient for Y
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C1	Two's complement green multiplier coefficient for Y

32.6.20 RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF_CSC_COEFF2)

LCDIF_CSC_COEFF2 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF2_C3_WR(0x3DB);//-0.148x256=-37
HW_LCDIF_CSC_COEFF2_C4_WR(0x3B6);//-0.291x256=-74
```

Address: 21C_8000h base + 130h offset = 21C_8130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						C4										Reserved						C3									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CSC_COEFF2 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C4	Two's complement green multiplier coefficient for Cb
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C3	Two's complement red multiplier coefficient for Cb

32.6.21 RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF_CSC_COEFF3)

LCDIF_CSC_COEFF3 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF3_C5_WR(0x70);//0.439x256=112
HW_LCDIF_CSC_COEFF3_C6_WR(0x70);//0.439x256=112
```

Address: 21C_8000h base + 140h offset = 21C_8140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						C6										Reserved						C5									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CSC_COEFF3 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C6	Two's complement red multiplier coefficient for Cr
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C5	Two's complement blue multiplier coefficient for Cb

32.6.22 RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF_CSC_COEFF4)

LCDIF_CSC_COEFF4 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF4_C7_WR(0x3A2);//-0.368x256=-94
HW_LCDIF_CSC_COEFF4_C8_WR(0x3EE);//-0.071x256=-18
```

Address: 21C_8000h base + 150h offset = 21C_8150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						C8										Reserved						C7									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CSC_COEFF4 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.

Table continues on the next page...

LCDIF_CSC_COEFF4 field descriptions (continued)

Field	Description
25–16 C8	Two's complement blue multiplier coefficient for Cr
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C7	Two's complement green multiplier coefficient for Cr

32.6.23 RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF_CSC_OFFSET)

LCDIF_CSC_ register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

Address: 21C_8000h base + 160h offset = 21C_8160h

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

LCDIF_CSC_OFFSET field descriptions

Field	Description
31–25 RSRVD1	This field is reserved. Reserved bits, write as 0.
24–16 CBCR_OFFSET	Two's complement offset for the Cb and Cr components
15–9 RSRVD0	This field is reserved. Reserved bits, write as 0.
Y_OFFSET	Two's complement offset for the Y component

32.6.24 RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF_CSC_LIMIT)

LCDIF_CSC_CTRL0 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC. Note that the values in this register are unsigned.

EXAMPLE

```
HW_LCDIF_CSC_LIMIT_CBCR_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_CBCR_MAX_WR(0xF0);//240
HW_LCDIF_CSC_LIMIT_Y_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_Y_MAX_WR(0xEB);//235
```

Address: 21C_8000h base + 170h offset = 21C_8170h

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

LCDIF_CSC_LIMIT field descriptions

Field	Description
31–24 CBCR_MIN	Lower limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
23–16 CBCR_MAX	Upper limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
15–8 Y_MIN	Lower limit of Y after RGB to 4:2:2 YCbCr conversion
Y_MAX	Upper limit of Y after RGB to 4:2:2 YCbCr conversion

32.6.25 LCD Interface Data Register (LCDIF_DATA)

This register is used to transfer data using the PIO interface mode of operation. In MPU mode, data written to this register will be transferred out to the display device. When receiving data from the display, data is read from this register using PIO operations. During write operations, data can be written to this register (from the processor's perspective) as bytes, half-words (16 bits), or words (32 bits) as desired.

This register holds the 32-bit word written by the ARM platform into LCDIF. This data then gets sent out by the block across the interface.

Address: 21C_8000h base + 180h offset = 21C_8180h

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

LCDIF_DATA field descriptions

Field	Description
31–24 DATA_THREE	Byte 3 (most significant byte) of data written to LCDIF.
23–16 DATA_TWO	Byte 2 of data written to eLCDIF.
15–8 DATA_ONE	Byte 1 of data written to eLCDIF.
DATA_ZERO	Byte 0 (least significant byte) of data written to eLCDIF.

32.6.26 Bus Master Error Status Register (LCDIF_BM_ERROR_STAT)

This register reflects the virtual address at which the AXI master received an error response from the slave.

When the BM_ERROR_IRQ is asserted, the address of the bus error is updated in the register.

Address: 21C_8000h base + 190h offset = 21C_8190h

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

LCDIF_BM_ERROR_STAT field descriptions

Field	Description
ADDR	Virtual address at which bus master error occurred.

32.6.27 CRC Status Register (LCDIF_CRC_STAT)

This register reflects the CRC value of each frame sent out by eLCDIF. The CRC is done on the final output bus, so the value will be dependent on the LCD_DATABUS_WIDTH bitfield even if the input data is the same.

This register will be updated when the CUR_FRAME_DONE_IRQ is asserted. In the case of DVI mode, the CRC is calculated for the entire frame, not separately for each field in the frame.

Address: 21C_8000h base + 1A0h offset = 21C_81A0h

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

LCDIF_CRC_STAT field descriptions

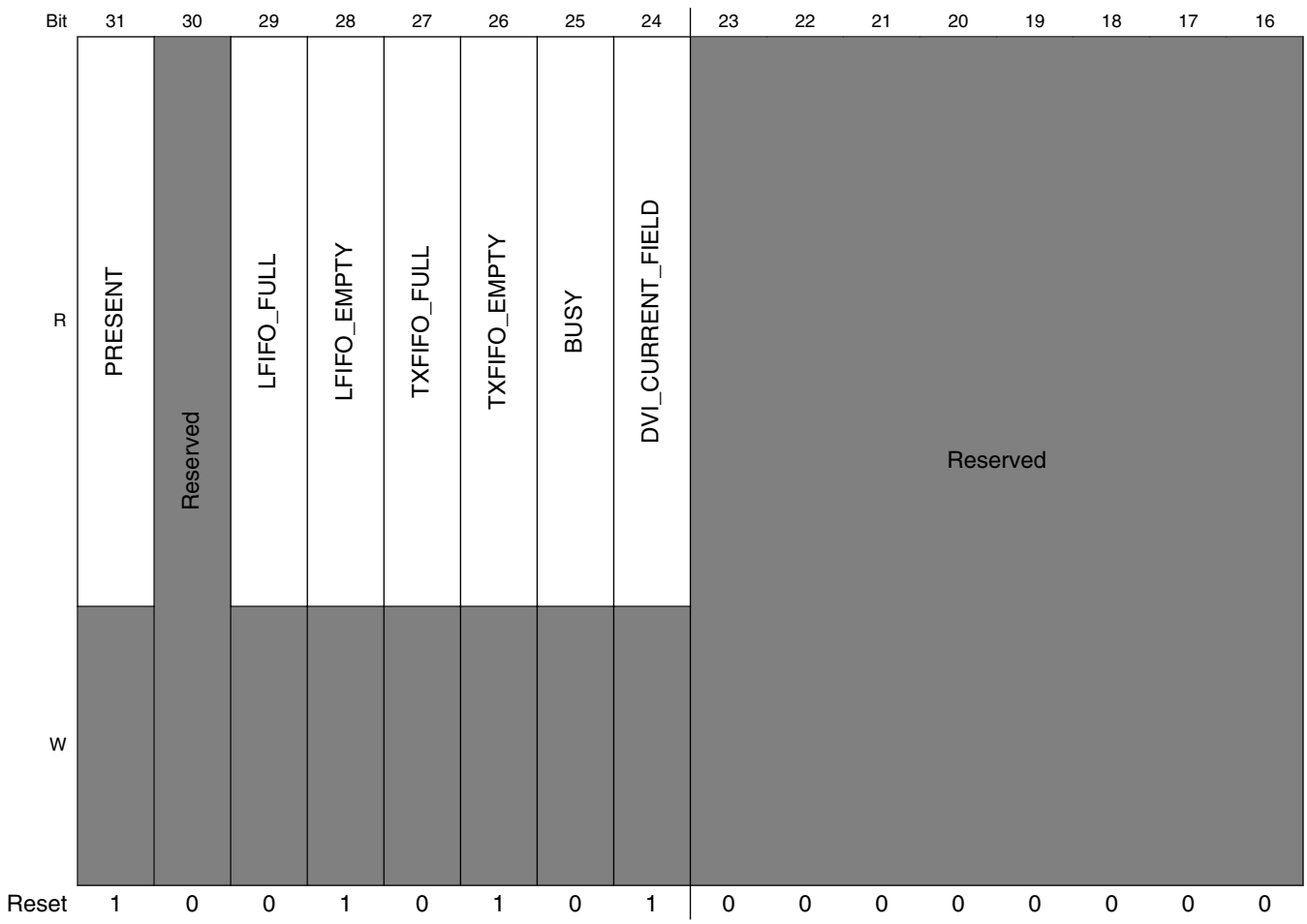
Field	Description
CRC_VALUE	Calculated CRC value.

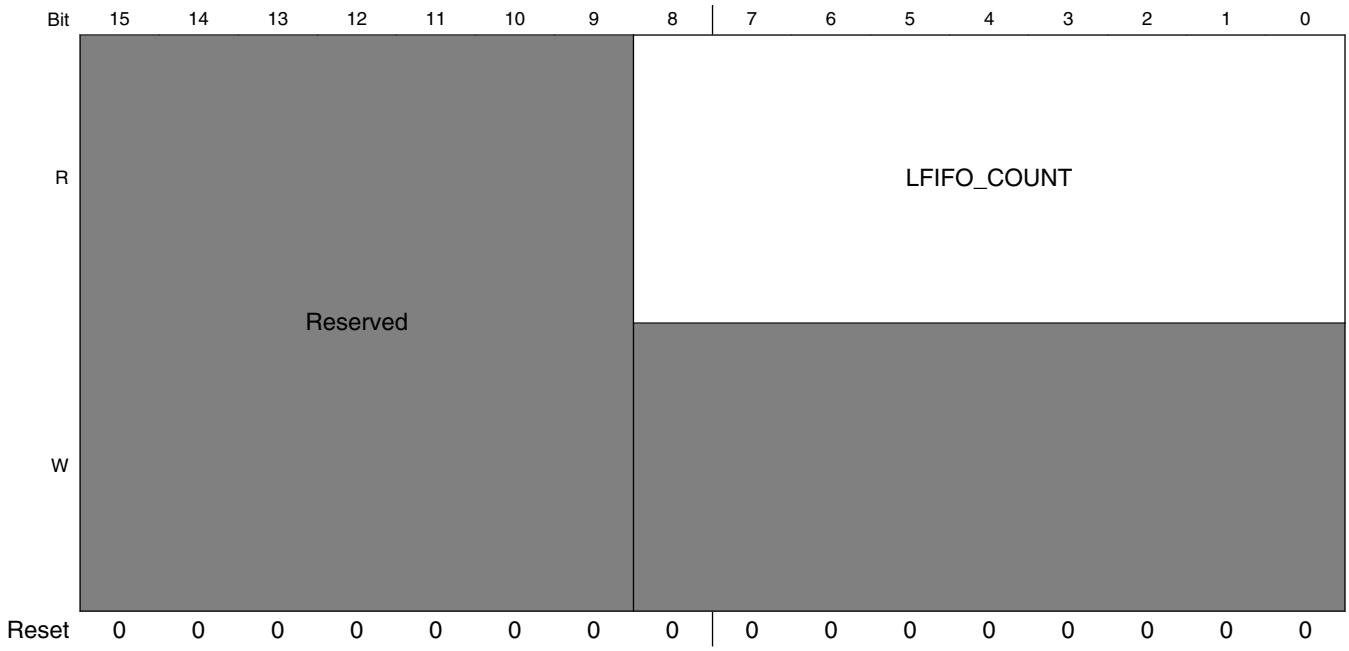
32.6.28 LCD Interface Status Register (LCDIF_STAT)

The LCD interface status register can be used to check the current status of the eLCDIF block.

The LCD interface status register that contains read only views of some parameters or current state of the block.

Address: 21C_8000h base + 1B0h offset = 21C_81B0h





LCDIF_STAT field descriptions

Field	Description
31 PRESENT	0: eLCDIF not present on this product 1: eLCDIF is present.
30 -	This field is reserved. Reserved.
29 LFIFO_FULL	Read only view of the signal that indicates that LCD read datapath FIFO is full, will be generally used in the write mode of the LCD interface.
28 LFIFO_EMPTY	Read only view of the signal that indicates that LCD read dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
27 TXFIFO_FULL	Read only view of the signal that indicates that LCD write datapath FIFO is full, will be generally used in the write mode of the LCD interface.
26 TXFIFO_EMPTY	Read only view of the signal that indicates that LCD write dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
25 BUSY	Read only view of the input busy signal from the external LCD controller.
24 DVI_CURRENT_FIELD	Read only view of the current field being transmitted. DVI_CURRENT_FIELD = 0 means field 1. DVI_CURRENT_FIELD = 1 means field 2.
23–9 RSRVD0	This field is reserved. Reserved bits. Write as 0.
LFIFO_COUNT	Read only view of the current count in Latency buffer (LFIFO).

32.6.29 LCD Interface Version Register (LCDIF_VERSION)

The LCD interface version register can be used to read the version of the eLCDIF IP being used in this SoC.

The LCD interface debug register is for diagnostic use only.

Address: 21C_8000h base + 1C0h offset = 21C_81C0h

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

LCDIF_VERSION field descriptions

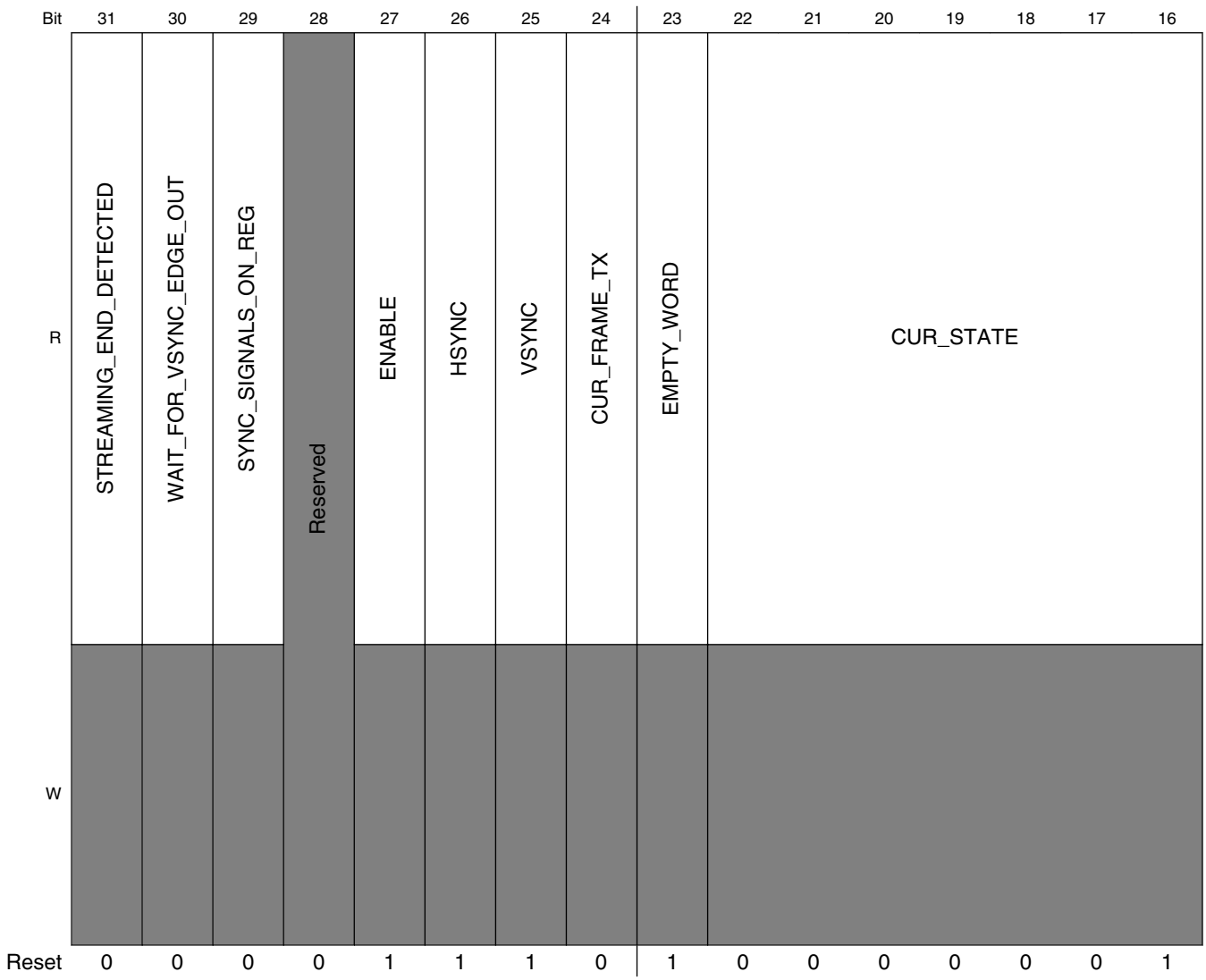
Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of RTL version.
STEP	Fixed read-only value reflecting the stepping of RTL version.

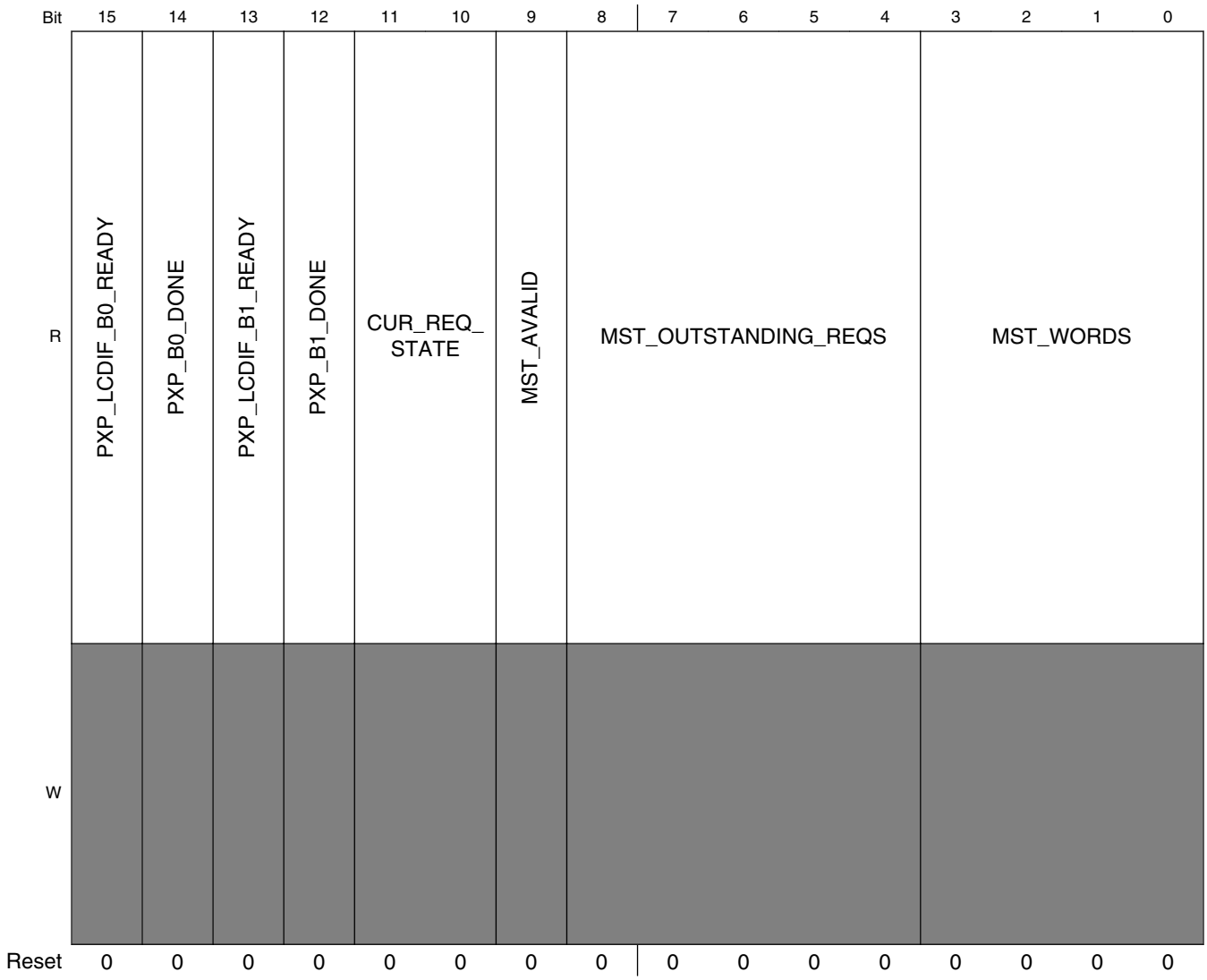
32.6.30 LCD Interface Debug0 Register (LCDIF_DEBUG0)

The LCD interface debug0 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: 21C_8000h base + 1D0h offset = 21C_81D0h





LCDIF_DEBUG0 field descriptions

Field	Description
31 STREAMING_ END_DETECTED	Read only view of the DOTCLK_MODE or DVI_MODE bit going from 1 to 0.
30 WAIT_FOR_ VSYNC_EDGE_ OUT	Read only view of WAIT_FOR_VSYNC_EDGE bit in the VSYNC mode after it comes out of the TXFIFO.
29 SYNC_SIGNALS_ ON_REG	Read only view of internal sync_signals_on_reg signal.
28 -	This field is reserved. Reserved.
27 ENABLE	Read only view of ENABLE signal.

Table continues on the next page...

LCDIF_DEBUG0 field descriptions (continued)

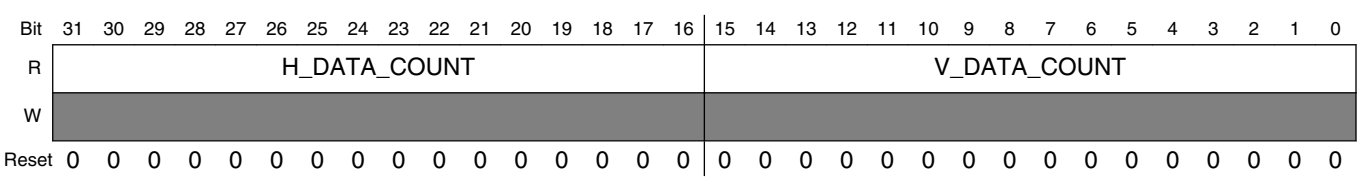
Field	Description
26 HSYNC	Read only view of HSYNC signal.
25 VSYNC	Read only view of VSYNC signal.
24 CUR_FRAME_TX	This bit is 1 for the time the current frame is being transmitted in the VSYNC mode. Useful for VSYNC mode debug.
23 EMPTY_WORD	Indicates that the current word is empty.
22–16 CUR_STATE	Read only view of the current state machine state in the current mode of operation.
15 PXP_LCDIF_B0_READY	Buffer0 ready signal issued by PXP.
14 PXP_B0_DONE	Buffer0 done signal issued by eLCDIF.
13 PXP_LCDIF_B1_READY	Buffer1 ready signal issued by PXP.
12 PXP_B1_DONE	Buffer1 done signal issued by eLCDIF.
11–10 CUR_REQ_STATE	Read only view of the request state machine.
9 MST_AVALID	Read only view of the mst_avalid signal issued by the AXI bus master.
8–4 MST_OUTSTANDING_REQS	Read only view of the current outstanding requests issued by the AXI bus master.
MST_WORDS	Read only view of the current bursts issued by the AXI bus master.

32.6.31 LCD Interface Debug1 Register (LCDIF_DEBUG1)

The LCD interface debug1 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: 21C_8000h base + 1E0h offset = 21C_81E0h



LCDIF_DEBUG1 field descriptions

Field	Description
31–16 H_DATA_ COUNT	Read only view of the current state of the horizontal data counter.
V_DATA_ COUNT	Read only view of the current state of the vertical data counter.

32.6.32 LCD Interface Debug2 Register (LCDIF_DEBUG2)

The LCD interface debug2 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: 21C_8000h base + 1F0h offset = 21C_81F0h

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

LCDIF_DEBUG2 field descriptions

Field	Description
MST_ADDRESS	Read only view of the current address issued by the AXI bus master.

32.6.33 eLCDIF Threshold Register (LCDIF_THRES)

This register is used to activate control signals when the number of pixels reaches the programmed threshold. These control signals, in turn, can be used to manipulate access priority or dynamically change the input clock frequency to meet the required pixel throughput.

Memory request priority threshold register.

Address: 21C_8000h base + 200h offset = 21C_8200h

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

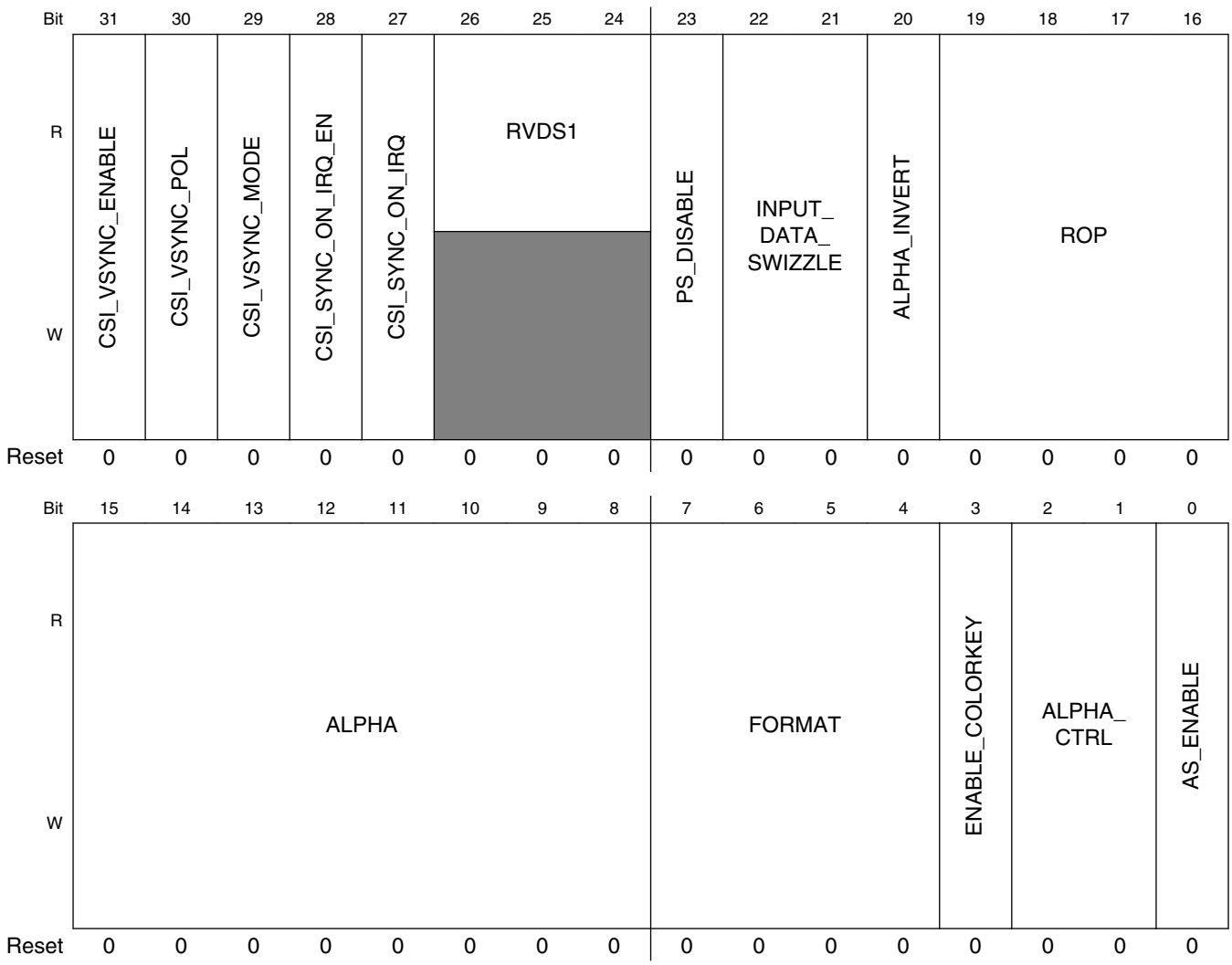
LCDIF_THRES field descriptions

Field	Description
31–25 RSRVD2	This field is reserved. Reserved bits. Write as 0.
24–16 FASTCLOCK	This value should be set to a value of pixels, from 0 to 511. When the number of pixels in the input pixel FIFO is LESS than this value, the fast clock control output will be raised. This signal can be used to reduce the system bus clock frequency to save power during horizontal or vertical blanking intervals. This value should also be programmed to a value that is greater than the "PANIC" threshold value. This will allow a faster clock to recover the number of pixels in the FIFO before a "panic" level is encountered.
15–9 RSRVD1	This field is reserved. Reserved bits. Write as 0.
PANIC	This value should be set to a value of pixels from 0 to 511. When the number of pixels in the input pixel FIFO is less than this value, the internal panic control output will be raised. This signal can be used to raise the access eLCDIF's access priority.

32.6.34 eLCDIF AS Buffer Control Register (LCDIF_AS_CTRL)

The Alpha Surface Parameter register provides additional controls for AS.

Address: 21C_8000h base + 210h offset = 21C_8210h



LCDIF_AS_CTRL field descriptions

Field	Description
31 CSI_VSYNC_ ENABLE	When this bit is set by software, the LCDIF work as sync mode with CSI input.
30 CSI_VSYNC_ POL	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.

Table continues on the next page...

LCDIF_AS_CTRL field descriptions (continued)

Field	Description
29 CSI_VSYNC_MODE	this bit is set by software to decide which vsync generate mode. LCDIF vsync generate by internal counter when set to 0, LCDIF vsync delayed by each csi_vsync_in when set to 1; INT_SYNC_MODE = 0x0 LCDIF vsync generate by internal counter. EXT_SYNC_MODE = 0x1 LCDIF vsync delayed by each csi_vsync_in.
28 CSI_SYNC_ON_IRQ_EN	This bit is set to enable an interrupt when LCDIF lock with CSI vsync input.
27 CSI_SYNC_ON_IRQ	this bit is set by software to decide which vsync generate mode. LCDIF vsync generate by internal counter when set to 0, LCDIF vsync delayed by each csi_vsync_in when set to 1; INT_SYNC_MODE = 0x0 LCDIF vsync generate by internal counter. EXT_SYNC_MODE = 0x1 LCDIF vsync delayed by each csi_vsync_in.
26–24 RVDS1	Reserved, always set to zero.
23 PS_DISABLE	When this bit is set by software, the LCDIF will disable PS buffer data.
22–21 INPUT_DATA_SWIZZLE	This field specifies how to swap the bytes either in the HW_LCDIF_DATA register or those fetched by the AXI master part of LCDIF. The swizzle function is independent of the WORD_LENGTH bit. See the explanation of the HW_LCDIF_DATA below for names and definitions of data register fields. The supported swizzle configurations are: NO_SWAP = 0x0 No byte swapping.(Little endian) LITTLE_ENDIAN = 0x0 Little Endian byte ordering (same as NO_SWAP). BIG_ENDIAN_SWAP = 0x1 Big Endian swap (swap bytes 0, 3 and 1, 2). SWAP_ALL_BYTES = 0x1 Swizzle all bytes, swap bytes 0, 3 and 1, 2 (aka Big Endian). HWD_SWAP = 0x2 Swap half-words. HWD_BYTE_SWAP = 0x3 Swap bytes within each half-word.
20 ALPHA_INVERT	Setting this bit to logic 0 will not alter the alpha value. A logic 1 will invert the alpha value and apply (1-alpha) for image composition.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CTRL field. MASKAS = 0x0 AS AND PS MASKNOTAS = 0x1 nAS AND PS MASKASNOT = 0x2 AS AND nPS MERGEAS = 0x3 AS OR PS MERGENOTAS = 0x4 nAS OR PS MERGEASNOT = 0x5 AS OR nPS NOTCOPYAS = 0x6 nAS NOT = 0x7 nPS NOTMASKAS = 0x8 AS NAND PS NOTMERGEAS = 0x9 AS NOR PS XORAS = 0xA AS XOR PS NOTXORAS = 0xB AS XNOR PS

Table continues on the next page...

LCDIF_AS_CTRL field descriptions (continued)

Field	Description
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE values are programmed in REG_AS_CTRL[ALPHA_CTRL]. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when selected.
7–4 FORMAT	Indicates the input buffer format for AS. ARGB8888 = 0x0 32-bit pixels with alpha RGB888 = 0x4 32-bit pixels without alpha (unpacked 24-bit format) ARGB1555 = 0x8 16-bit pixels with alpha ARGB4444 = 0x9 16-bit pixels with alpha RGB555 = 0xC 16-bit pixels without alpha RGB444 = 0xD 16-bit pixels without alpha RGB565 = 0xE 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this alpha surface. Pixels found in the alpha surface colorkey range will be displayed as transparent (the PS pixel will be used).
2–1 ALPHA_CTRL	Determines how the alpha value is constructed for this alpha surface. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. Embedded = 0x0 Indicates that the AS pixel alpha value will be used to blend the AS with PS. The ALPHA field is ignored. Override = 0x1 Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. Multiply = 0x2 Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. ROPs = 0x3 Enable ROPs. The ROP field indicates an operation to be performed on the alpha surface and PS pixels.
0 AS_ENABLE	When this bit is set by software, the LCDIF will start fetching AS buffer data in bus master mode and combine it with another buffer.

32.6.35 Alpha Surface Buffer Pointer (LCDIF_AS_BUF)

This register is used to indicate the base address of the AS buffer.

Address: 21C_8000h base + 220h offset = 21C_8220h

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

LCDIF_AS_BUF field descriptions

Field	Description
ADDR	Address pointer for the alpha surface 0 buffer.

32.6.36 LCDIF_AS_NEXT_BUF

When the LCDIF is behaving as a master, this address points to the address of the next frame of data that will be sent out via the LCDIF. It is upto the software to make sure that this register is programmed before the end of the current frame, otherwise it might result in old data going out the LCDIF. This address must always be double-word aligned.

Address: 21C_8000h base + 230h offset = 21C_8230h

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

LCDIF_AS_NEXT_BUF field descriptions

Field	Description
ADDR	Address of the next frame that will be transmitted by eLCDIF.

32.6.37 eLCDIF Overlay Color Key Low (LCDIF_AS_CLRKEYLOW)

If a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. Colorkey operations are higher priority than alpha or ROP operations.

Address: 21C_8000h base + 240h offset = 21C_8240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1								PIXEL																							
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LCDIF_AS_CLRKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of RGB color key applied to AS buffer

32.6.38 eLCDIF Overlay Color Key High (LCDIF_AS_CLRKEYHIGH)

If a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. Colorkey operations are higher priority than alpha or ROP operations.

Address: 21C_8000h base + 250h offset = 21C_8250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1								PIXEL																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_AS_CLRKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of RGB color key applied to AS buffer

32.6.39 LCD working insync mode with CSI for VSYNC delay (LCDIF_SYNC_DELAY)

The LCDIF DOTCLK mode VSYNC will delay from CSI_VSYNC as
(V_COUNT_DELAY * HSYNC_PERIOD + H_COUNT_DELAY) PIXCLK cycles

Address: 21C_8000h base + 260h offset = 21C_8260h

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

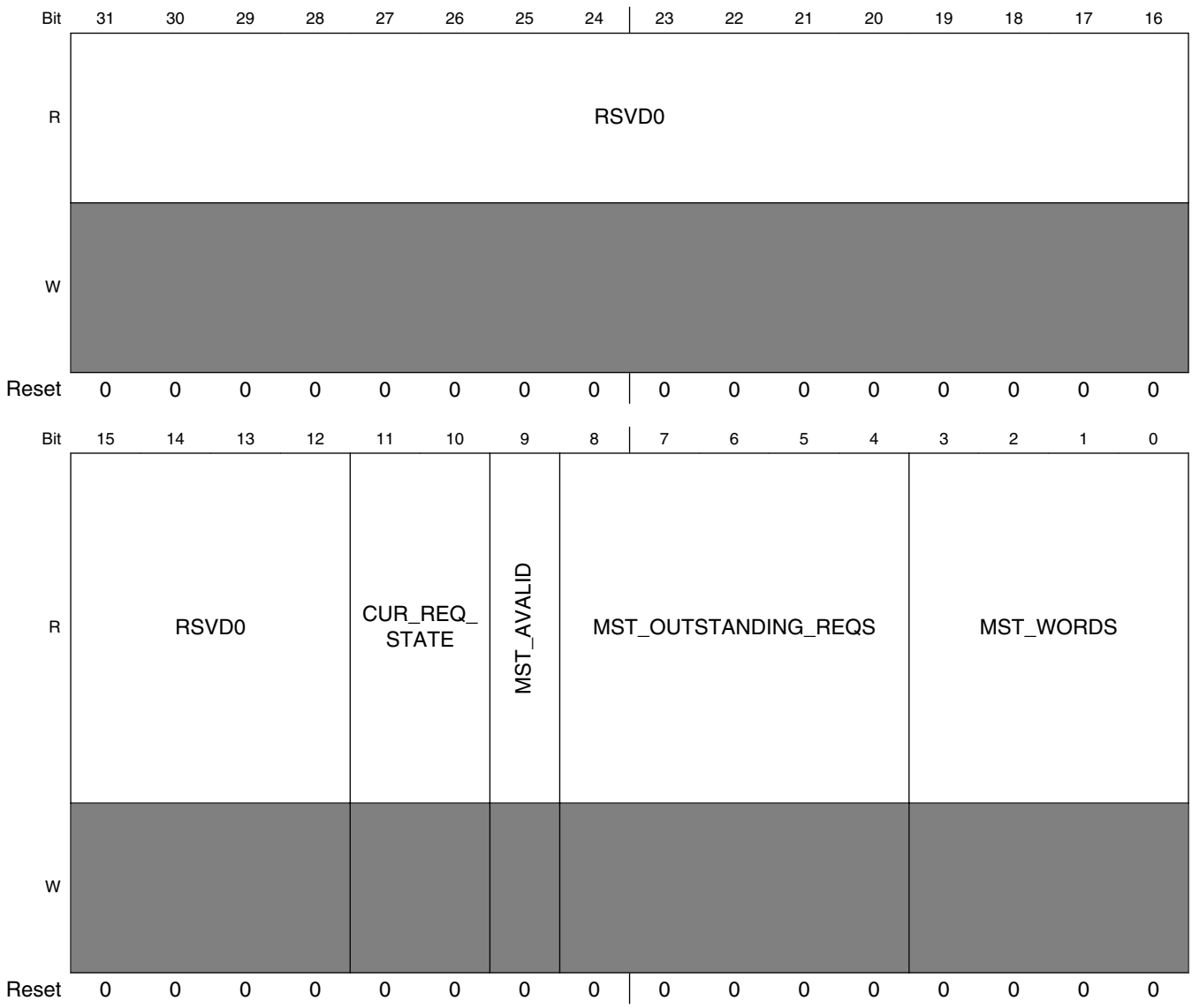
LCDIF_SYNC_DELAY field descriptions

Field	Description
31–16 V_COUNT_ DELAY	LCDIF VSYNC delayed counter for CSI_VSYNC.
H_COUNT_ DELAY	LCDIF VSYNC delayed counter for CSI_VSYNC.

32.6.40 eLCDIF Interface Debug3 Register (LCDIF_DEBUG3)

The LCD interface debug register is for diagnostic use only.

Address: 21C_8000h base + 270h offset = 21C_8270h



LCDIF_DEBUG3 field descriptions

Field	Description
31–12 RSVD0	Reserved bits, write as 0.

Table continues on the next page...

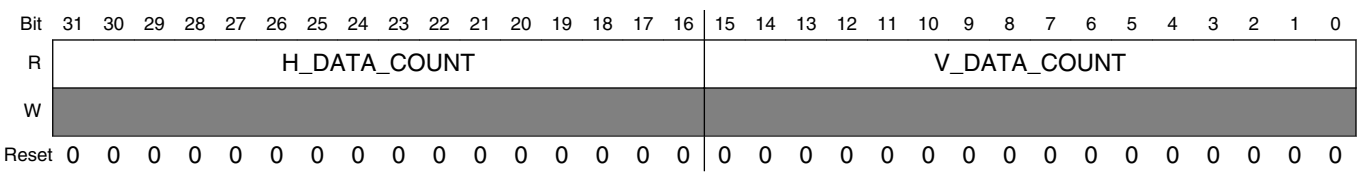
LCDIF_DEBUG3 field descriptions (continued)

Field	Description
11–10 CUR_REQ_ STATE	Read only view of the request state machine
9 MST_AVALID	Read only view of the mst_avalid signal issued by the AXI bus master
8–4 MST_ OUTSTANDING_ REQS	Read only view of the current outstanding requests issued by the AXI bus master
MST_WORDS	Read only view of the current bursts issued by the AXI bus master.

32.6.41 LCD Interface Debug4 (LCDIF_DEBUG4)

The LCD interface debug register is for diagnostic use only.

Address: 21C_8000h base + 280h offset = 21C_8280h



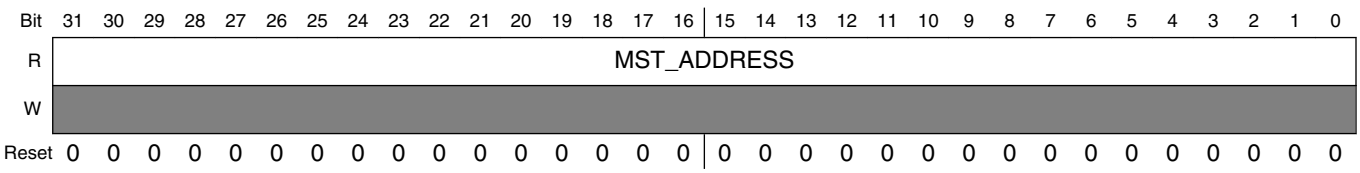
LCDIF_DEBUG4 field descriptions

Field	Description
31–16 H_DATA_ COUNT	Read only view of the current AS state of the horizontal data counter.
V_DATA_ COUNT	Read only view of the current AS state of the vertical data counter.

32.6.42 LCD Interface Debug5 (LCDIF_DEBUG5)

The LCD interface debug register is for diagnostic use only.

Address: 21C_8000h base + 290h offset = 21C_8290h



LCDIF_DEBUG5 field descriptions

Field	Description
MST_ADDRESS	Read only view of the AS channel address issued by the AXI bus master.

Chapter 33
Multi Mode DDR Controller (MMDC)

33.1 Overview

MMDC is a multi-mode DDR controller that supports DDR3/DDR3L x16 and LPDDR2 x16 memory types. MMDC is configurable, high performance, and optimized.

The following figure shows the MMDC block diagram.

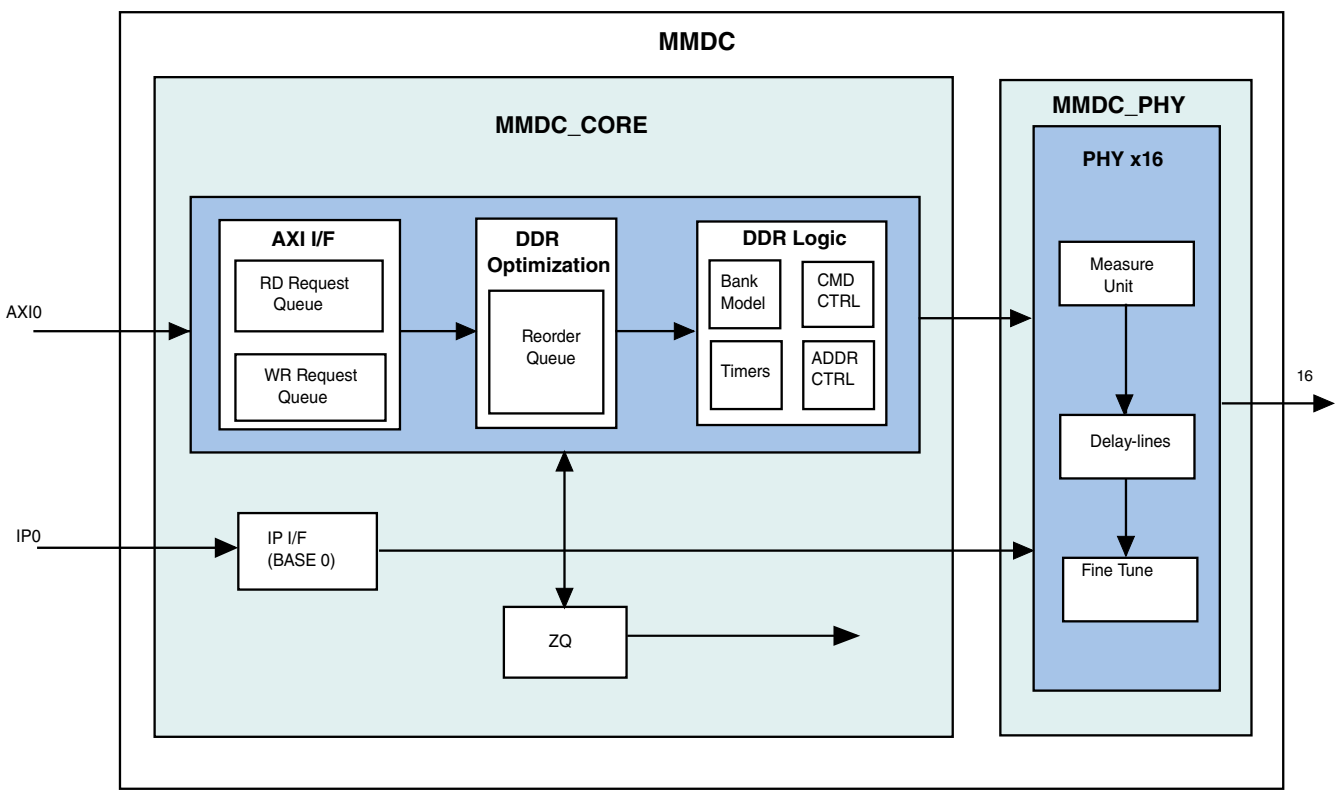


Figure 33-1. MMDC block diagram

MMDC consists of a core (MMDC_CORE) and PHY (MMDC_PHY).

- The core is responsible for communication with the system through an AXI interface, DDR command generation, DDR command optimizations, and a read/write data path.
- The PHY is responsible for the timing adjustment; it uses special calibration mechanisms to ensure data capture margin at a clock rate of up to 400 MHz.

The internal memory map(configuration registers) of the MMDC can be configured through an IP channel (IP0). .

33.1.1 MMDC feature summary

The table found here summarizes the MMDC features.

Table 33-1. MMDC feature summary

Feature	Details
DDR standards	• DDR3L, DDR3 x16 • LPDDR2 x16 • Does not support LPDDR1MDDR or DDR2

DDR interface	• x16 data bus width
	• Density per DDR device of 256 Mbits–8 Gbits with the following column and row combinations:
	  • Column size of 8–12 bits
	  • Row size of 11–16 bits
	• Two chip selects (one chip select for DDR3/DDR3L, two chip selects for LPDDR2)
	• Up to 4 Gbytes of address space with configurable partitioning between CS0 and CS1 (for LPDDR2 2ch x32 up to 2 Gbytes per channel)
	• Supports burst length of 8 (aligned) for DDR3
	• Supports burst length of 4 for LPDDR2

DDR performance	• MMDC running at up to 400 MHz (800 MT/s), see CCM block for actual clock frequencies supported. • Supports Real-Time priority by means of QoS sideband priority signals from the chip to enable various priority levels in the re-ordering mechanism: real-time, latency sensitive, normal priority. • Page hit/page miss optimizations • Consecutive read/write access optimizations • Supports deep read and write request queues to enable bank prediction. • Drives back the critical word in a read transaction as soon as it is received by the DDR device (does not wait until the whole data phase has been completed). • Keeps tracking of open memory pages • Supports bank interleaving • Special optimization in case of non-aligned wrap accesses in DDR3 mode (burst length 8) **NOTE:** Due to reordering and optimization mechanisms (per different AXI Identifier (ID)), the transactions towards the DDR device may be driven in a different ID order than was received by the AXI master. In a similar fashion, the write response, read response or read data may be driven to the AXI master in a different ID order.
AXI interface	• AXI bus compliant • Supports bus transfers of 8, 16 , 64 bits (single accesses and bursts) running at 400 MHz.

Table continues on the next page...

Table 33-1. MMDC feature summary (continued)

Feature	Details

	• Supports AXI bursts length of up to 16
	• Supports burst types of WRAP, INCR and FIXED
	• Supports 16 bits AXI ID
	• Write data interleave depth is 1 (no support for Write Data Interleave)
	• Supports write data before address
	• Supports buffered/non-buffered accesses (AWCACHE[0] = 0b means a non-bufferable access and AWCACHE[0] = 1b means a bufferable access). The rest of the CACHE options are not supported
	  • To keep data access coherency between write and read access of the same master, the response signal is sent as follows:
	  • Bufferable write access—BRESP will be sent when last data of the access has entered the MMDC.
	  • Non-bufferable write access—BRESP will be sent when the data was physically written into the external memory device.
	• Supports four exclusive monitors per configurable ID for only a single access with a size of up to 64 bits
	• Supports AXI responses as follows:
	  • Okay in case the access has been successful or exclusive access failure
	  • Slave error in case of security violation
	  • Exclusive okay in case the read or the write portion of an exclusive access has been successful

DDR calibration and delay-lines.	• Supports various calibration processes which can be performed either automatically (hardware) or manually (software) towards either CS0 or CS1. (At the end of the process the delay-lines will work with one set of results.) The following calibration processes are supported:
	  • ZQ calibration for external DDR device (in DDR3 through ZQ calibration command and in LPDDR2 through MRW command)
	  • Can be handled automatically for ZQ Short (periodically) and ZQ Long (at exit from self-refresh)
	  • Can be handled manually at ZQ INIT
	  • ZQ calibration for DDR I/O pads for calibrating the DDR driving strength
	  • The sequence can be handled automatically by hardware
	  • The sequence can be handled step by step manually by software
	  • Read data calibration. Adjustment of read DQS with read data byte.
	  • Read DQS gating calibration for DDR3 only. Adjustment of DQS gate with read preamble window.
	  • Write data calibration. Adjustment of write DQS with write data byte.
	  • Write leveling calibration. Adjustment of write DQS with CK (DDR differential clock).
	  • Read fine tuning. Adjustment of up to 7 delay-line units for each read data bit.
	  • Write fine tuning. Adjustment of up to 3 delay-line units for each read data bit.
	  • Periodic delay-line measurement for keeping its accuracy during refresh interval.
	  • Additional fine tuning delay lines to adjust DDR clock delay, DDR clock duty cycle, DQS duty cycle.

Power saving	• Support of dynamic voltage, frequency change and self-refresh mode entry through hardware and software negotiation with the system (request/acknowledge handshake)
	  • Upon hardware or software self-refresh request assertion, further AXI requests are blocked (even before the assertion of the acknowledge).
	  • During self-refresh mode the system may deassert the operating clock of the MMDC for power saving.
	  • During self-refresh mode the clock (CK) that is driven to the DDR device will be gated for power saving.
	• Supports automatic self-refresh and power down entry and exit
	  • In automatic self-refresh, the internal operating clock will be gated for power saving.

Table continues on the next page...

Table 33-1. MMDC feature summary (continued)

Feature	Details
	• Supports fast and slow precharge power down in DDR3 • Automatic active and precharge power down timer per chip select (one chip select can enter power down while the other is still working) • While CS (chip-select) is inactive (high) the command and address buses are not toggling for power saving.

DDR general	• Configurable timing parameters
	• Configurable refresh scheme
	• Page boundary crossing support
	  • Automatically generates precharge command and activates the next row
	• Supports various ODT control schemes
	  • Assertion or deassertion of ODT control per read or write accesses and for active or passive CS (chip-select)
	• Supports MRW and MRR commands for LPDDR2
	• Software control in LPDDR2 mode for switching to derated timing parameters and/or update the refresh rate according to temperature sensor
	• Debug and profiling capabilities

33.2 External Signals

The table found here describes the external signals of MMDC.

Table 33-2. MMDC External Signals

Signal	Description	Pad	Mode	Direction
DRAM_ADDR[15:0]	Address Bus Signals	DRAM_A[15:0]	No Muxing	O
DRAM_CAS	Column Address Strobe Signal	DRAM_CAS	No Muxing	O
DRAM_CS[1:0]	Chip Selects	DRAM_CS[1:0]	No Muxing	O
DRAM_DATA[31:0]	Data Bus Signals	DRAM_D[31:0]	No Muxing	I/O
DRAM_DQM[1:0]	Data Mask Signals	DRAM_DQM[1:0]	No Muxing	O
DRAM_ODT[1:0]	On-Die Termination Signals	DRAM_SDODT[1:0]	No Muxing	O
DRAM_RAS	Row Address Strobe Signal	DRAM_RAS	No Muxing	O
DRAM_RESET	Reset Signal	DRAM_RESET	No Muxing	O
DRAM_SDBA[2:0]	Bank Select Signals	DRAM_SDBA[2:0]	No Muxing	O
DRAM_SDCKE[1:0]	Clock Enable Signals	DRAM_SDCKE[1:0]	No Muxing	O
DRAM_SDCLK0_N	Negative Clock Signals	DRAM_SDCLK_[1:0]	No Muxing	O
DRAM_SDCLK0_P	Positive Clock Signals	DRAM_SDCLK_[1:0]	No Muxing	O
DRAM_SDQS[1:0]_N	Negative DQS Signals	DRAM_SDQS[1:0]_N	No Muxing	I/O
DRAM_SDQS[1:0]_P	Positive DQS Signals	DRAM_SDQS[1:0]_P	No Muxing	I/O
DRAM_SDWE	WE signal	DRAM_SDWE	No Muxing	O
DRAM_ZQPAD	ZQ signal	DRAM_ZQPAD	No Muxing	O

33.3 Clocks

The table found here describes the clock sources for MMDC. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 33-3. MMDC Clocks

Clock name	Clock Root	Description
aclk_fast_core_p0	mmdc_axi_clk_root	Fast clock (channel 1)
ipg_clk_p0	ipg_clk_root	Peripheral clock (channel 1)
aclk_fast_phy_p0	mmdc_axi_clk_root	Fast clock (channel 1 - PHY)

33.4 Functional Description

This section provides a complete functional description of the block.

33.4.1 Write/Read data flow

33.4.1.1 Write data flow

1. Write requests are received into an 8 entries request FIFO. Access is received only when there are at least two available entries. Each entry holds all of the AXI attributes.
   - If the burst length is greater than 8, the access splits into two accesses: one with burst length 8 and the other with the remainder.
   - The access can be performed as soon as the entire data phase of the associated write request is completed (all data beats were received).
2. A simple round-robin arbitration between the pending read and write accesses is performed, and the pointer to this stage's winner access is sent to the re-ordering buffer.
3. The reordering mechanism is activated to find the winner access, which is the access that best utilizes the DDR bus, based on its dynamic score. For further information see Dynamic scoring mode (Arbitration Winning Conditions).
4. The winner write access at the previous stage is received and is held for dispatch to the DDR logic.

5. When the DDR command control unit is ready to accept the write request, it issues (if needed) a precharge/active command to the DDR device according to the status of the bank model and the parameters of the timers.
6. The DDR logic drives the associated data to the DDR device through the DDR PHY.

33.4.1.2 Read data flow

1. Read requests are received into a 16 entry request FIFO in MMDC if there are at least two available entries. Each entry holds all of the AXI attributes.

NOTE

If the burst length is greater than 8, the access splits into 2 accesses (one with burst length 8 and the other with the remainder).

2. A simple round-robin arbitration between the pending read and write accesses is performed and the pointer to this phase's winner access is sent to the re-ordering buffer.
3. The reordering mechanism is activated to find the winner access, which is the access that best utilizes the DDR bus, based on its dynamic score. For further information see Dynamic scoring mode (Arbitration Winning Conditions).
4. The winner read access at the previous stage is sampled and is held for dispatch to the DDR logic. This read access will be dispatched when there is at least one free slot in the read data buffer to store the data.
5. When the DDR command control unit is ready to accept the read request, it issues (if needed) a precharge/active command to the DDR device according to the status of the bank model and the parameters of the timers.
6. The MMDC PHY samples the read data, and the DDR logic transfers the data to the associated slot in the read data buffer.
7. MMDC transfers the data back to the master.

33.4.2 MMDC initialization

Because the MMDC is disabled when the chip exits reset, no clock is driven to the DDR device and the whole interface towards the DDR device is inactive. The following steps are required to activate the MMDC properly.

NOTE

To guarantee that the DRAM_RESET and DRAM_SDCKE signals are kept low during the power-up and reset sequences of

the chip (as defined by JEDEC), you must connect those signals to pull-down resistors.

1. Set MDSCR[CON_REQ], which sets the configuration request; note that because the MMDC is disabled, there is no need to poll the configuration acknowledge bit at MDSCR[CON_ACK].
2. Configure the desired timing parameters at the MDCFG0, MDCFG1, MDCFG2, and MDOTC registers.
3. Configure the DDR type and other miscellaneous parameters at the MDMISC register.
4. Configure the required delay while leaving reset, at the MDOR register.
5. Configure the DDR physical parameters (density and burst length) at the MDCTL register.
6. Perform a ZQ calibration of the MMDC module to correctly initialize drive strengths.
7. Enable MMDC with the desired chip select at MDCTL[SDE_0] (for chip select 0) and MDCTL[SDE_1] (for chip select 1). At this point, MMDC starts the reset and initialization sequence related to DRAM_RESET/DRAM_SDCKE as defined by JEDEC.
8. Complete the initialization sequence as defined by JEDEC by issuing MRS/MRW commands for (ZQ, ODT, PRE, and so on). To issue those commands, configure the appropriate command and address at the MDSCR register.
9. Program the DDR mode registers by configuring the appropriate command and address at the MDSCR register.
10. Configure the power down and self-refresh entry and exit parameters at the MDPDC and MAPSR registers.
11. Configure the ZQ scheme at the MPZQHWCTRL and MPZQLP2CTRL registers.
12. Configure and activate the periodic refresh scheme at the MDREF register.
13. Deassert the configuration request by clearing MDSCR[CON_REQ].

NOTE

Steps 1 through 5 are non-blocking and can be done in any order.

Upon completion of these steps, MMDC is ready for work and to process AXI accesses.

NOTE

To achieve better timing and better precision, it is recommended that users configure the MMDC PHY delay parameters by operating either the automatic or manual calibration process. Before starting any calibration process, you must disable the periodic refresh scheme (MDREF[REF_SEL] = 11) and then issue a manual refresh command by configuring

MDSCR[CMD] to 2h. For further information, see Calibration Process.

33.4.3 Configuring the MMDC registers

To safely modify MMDC's internal configuration registers, MMDC must be placed into configuration mode.

Use the following steps to enter configuration mode.

1. Issue a configuration request by setting MDSCR[CON_REQ].
2. Poll on configuration acknowledge until it is set at MDSCR[CON_ACK].

At this point, MMDC enters configuration mode and accessing the MMDC registers is permitted.

NOTE

During configuration mode, MMDC prevents further AXI accesses from being acknowledged.

Upon deassertion of MDSCR[CON_REQ], MMDC leaves configuration mode and AXI accesses are processed.

33.4.4 MMDC Address Space

33.4.4.1 Address decoding

MMDC supports up to two consecutive chip selects, each with the same density.

It is optional to configure the partition between the chip selects through MDASP[CS0_END].

The incoming AXI address bus is 32 bits. MMDC decodes each access as follows:

1. chip select
2. bank number
3. row number
4. column number

The following registers in the MMDC define the DDR address space:

- MDMISC[DDR_4_BANK]—Defines either 4 or 8 banks in the DDR device
- MDCTL[DSIZ]—Defines the DDR data bus width of x16

- MDMISC[BI]—Defines whether bank interleaving is on or off
- MDCTL[COL]—Defines the column size of the DDR device
- MDCTL[ROW]—Defines the row size of the DDR device

The following tables show address decoding examples for x16 bit DDR devices when bank interleaving is both on and off. It is assumed that the configration is as follows: 8 banks (3 bits), 15 bit assignment for the row, and 10 bit sassignment for the column. The total density is 256 MWords (512 Mbytes for x16).

NOTE

Chip selection is done by comparing the 7 most significant address bits (ARADDR[31:25]/AWADDR[31:25]) with MDASP[CS0_END].

Table 33-4. Address decoding—bank interleaving off

AXI ADDRESS	x16 DDR	x32 DDR
A29	—	BANK[2]
A28	BANK[2]	BANK[1]
A27	BANK[1]	BANK[0]
A26	BANK[0]	ROW[14]
A25	ROW[14]	ROW[13]
A24	ROW[13]	ROW[12]
A23	ROW[12]	ROW[11]
A22	ROW[11]	ROW[10]
A21	ROW[10]	ROW[9]
A20	ROW[9]	ROW[8]
A19	ROW[8]	ROW[7]
A18	ROW[7]	ROW[6]
A17	ROW[6]	ROW[5]
A16	ROW[5]	ROW[4]
A15	ROW[4]	ROW[3]
A14	ROW[3]	ROW[2]
A13	ROW[2]	ROW[1]
A12	ROW[1]	ROW[0]
A11	ROW[0]	COL[9]
A10	COL[9]	COL[8]
A9	COL[8]	COL[7]
A8	COL[7]	COL[6]
A7	COL[6]	COL[5]
A6	COL[5]	COL[4]
A5	COL[4]	COL[3]
A4	COL[3]	COL[2]

Table continues on the next page...

Table 33-4. Address decoding—bank interleaving off (continued)

AXI ADDRESS	x16 DDR	x32 DDR
A3	COL[2]	COL[1]
A2	COL[1]	COL[0]
A1	COL[0]	—
A0	—	—

Table 33-5. Address decoding—bank interleaving on

AXI ADDRESS	x16 DDR	x32 DDR
A29	—	ROW[14]
A28	ROW[14]	ROW[13]
A27	ROW[13]	ROW[12]
A26	ROW[12]	ROW[11]
A25	ROW[11]	ROW[10]
A24	ROW[10]	ROW[9]
A23	ROW[9]	ROW[8]
A22	ROW[8]	ROW[7]
A21	ROW[7]	ROW[6]
A20	ROW[6]	ROW[5]
A19	ROW[5]	ROW[4]
A18	ROW[4]	ROW[3]
A17	ROW[3]	ROW[2]
A16	ROW[2]	ROW[1]
A15	ROW[1]	ROW[0]
A14	ROW[0]	BANK[2]
A13	BANK[2]	BANK[1]
A12	BANK[1]	BANK[0]
A11	BANK[0]	COL[9]
A10	COL[9]	COL[8]
A9	COL[8]	COL[7]
A8	COL[7]	COL[6]
A7	COL[6]	COL[5]
A6	COL[5]	COL[4]
A5	COL[4]	COL[3]
A4	COL[3]	COL[2]
A3	COL[2]	COL[1]
A2	COL[1]	COL[0]
A1	COL[0]	—
A0	—	—

NOTE

In cases where this is an access to a non-initialized or disconnected chip select, behavior may be unexpected.

33.4.4.2 Chip select settings

MMDC drives the incoming access to either CS0 or CS1 by comparing the 7 most significant address bits (ARADDR[31:25]/AWADDR[31:25]) with MDASP[CS0_END].

Generally, the total density per chip-select must be the same, and the total density per chip-select must be a power of two.

Creating 4 Gbyte address space with 2 Gbyte CS density and Creating 2 Gbyte address spaces with 1 Gbyte CS density show how to create a continous address space and configure the MMDC accordingly.

33.4.4.2.1 Creating 4 Gbyte address space with 2 Gbyte CS density

If the DDR memory space allocation is 4 Gbytes, only one configuration of chip select partition is allowed. The register MDASP[CS0_END] should be set to 2 Gbytes partition (16Gb).

The figure below shows the associated memory space.

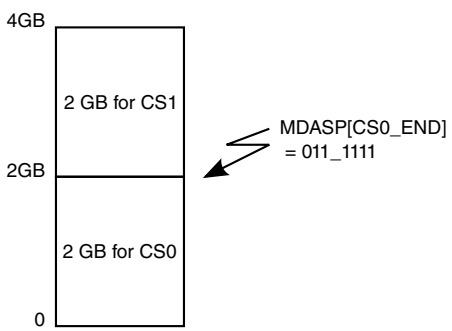


Figure 33-2. Chip select partion—2 Gbytes per chip select

33.4.4.2.2 Creating 2 Gbyte address spaces with 1 Gbyte CS density

If the DDR memory space allocation is 2 Gbytes, there are three options for configuring the chip select partition:MDASP[CS0_END] to 001_1111 (1 Gbyte), MDASP[CS0_END] to 011_1111 (2 Gbytes), and MDASP[CS0_END] to 101_1111 (3 Gbytes).

If DDR memory space allocation is 2 Gbytes, there are three options for configuring the chip select partition:

- MDASP[CS0_END] to 001_1111 (1 Gbyte)
- MDASP[CS0_END] to 011_1111 (2 Gbytes)
- MDASP[CS0_END] to 101_1111 (3 Gbytes)

The figure below shows the associated memory space:

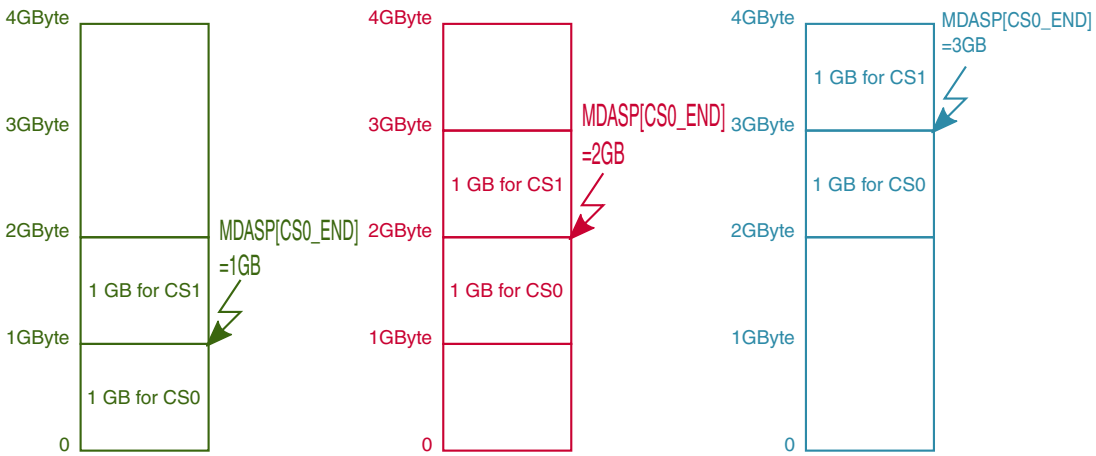


Figure 33-3. Chip select partion—1 Gbyte per chip select

33.4.4.3 Translation of AXI accesses to DDR accessess

33.4.4.3.1 Example 1

Assume the AXI read access has the following attributes:

- Wrap (arburst[1:0] = 10b)
- AXI size of 128 bits (arsize[2:0] = 100b)
- AXI length of 8 (arlen[3:0] = 0111b)
- AXI address with suffix B0h (non-aligned to AXI wrap boundary which is 16Bx8 = 128B = 0x80)

Toward DDR3(MDMISC[DDR_TYPE] = 00b) with the following attributes:

- x32 (MDCTL[DSIZ] = 01b)
- burst length of 8 (MDCTL[BL] = 1b)

In this case, the AXI wrap boundary is every 16Bx8=128B (0x80) and the DDR wrap boundary is every 4Bx8=32B(0x20).

The master expects to fetch the data that is associated with the following addresses: 0xB0, 0xC0, 0xD0, 0xE0, 0xF0, 0x80, 0x90, 0xA0.

The first aligned AXI address that is associated with suffix 0xB0 is with suffix 0x80, and the last wrap AXI address is with suffux 0xFF. Because the AXI master expects to get the first data from AXI address with suffix 0xB0, the MMDC issues the following accesses toward the DDR:

- Read access toward logic address with suffix 0xA0 (DDR boundary is 0x20 and 0xA0 is the closest to 0xB0)

NOTE

Logic address is the address of the column normalized to 1 byte.

- Read access toward logic address with suffix 0xC0
- Read access toward logic address with suffix 0xE0
- Read access toward logic address with suffix 0x80

The MMDC breaks the AXI access into four DDR accesses and returns the read data associated with address 0xB0 to the master first. The read data fetched from address 0xA0 is stored in the internal buffers and is driven back to the master at the end.

33.4.4.3.2 Example 2

Assume the AXI write access has the following attributes:

- Increment (awburst[1:0]=2'b01)
- AXI size of 64bits (awsize[2:0]=3'b011)
- AXI length of 8 (awlen[3:0]=4'b0111)
- AXI address with suffix 0xB0 (aligned as the size of the increment is 8B)

Toward DDR3(MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x64 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case, the AXI alignment is every 8B and the DDR boundary is every 8Bx8=64B(0x40).

The master expects to write the data to the following addresses: 0xB0, 0xB8, 0xC0, 0xC8, 0xD0, 0xD8, 0xE0, 0xE8.

The MMDC will issue the following accesses toward the DDR:

- Write access toward logic address with suffix 0x80 (DDR boundary is 0x40 and 0x80 is the closest to 0xB0) while address 0x80 to 0xAF are masked by DM (data masking signal).

NOTE

Logic address is the address of the column normalized to 1 byte.

- Write access toward logic address with suffix 0xC0 while addresses 0xF0 through 0xFF are masked by DM (data masking signal)

The MMDC will break the AXI access into two DDR accesses.

33.4.4.3.3 Example 3

Assume the AXI write access has the following attributes:

- Wrap (awburst[1:0]=2'b10)
- AXI size of 128bits (awsize[2:0]=3'b100)
- AXI length of 4 (awlen[3:0]=4'b0011)
- AXI address with suffix 0x80 (aligned)

Toward DDR3(MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x64 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case, the AXI wrap boundary is every 16Bx4=64B (0x40) and the DDR wrap boundary is every 8x8=64B(0x40).

The master expects to write the data to the following addresses: 0x80, 0x90, 0xA0, 0xB0.

Because the AXI wrap boundary and DDR wrap boundary are similar and the starting AXI address is aligned, the MMDC will issue only one access toward the DDR as follows:

- Write access towards logic address with suffix 0x80

NOTE

Logic address is the address of the column normalized to 1 byte.

33.4.4.3.4 Example 4

Assume the AXI write access has the following attributes:

- Increment (awburst[1:0]=2'b01)

- AXI size of 64bits (awsize[2:0]=3'b011)
- AXI length of 2 (awlen[3:0]=4'b0001)
- AXI address with suffix 0x5 (non aligned)

Toward DDR3(MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x32 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case the AXI alignment is every 8B (0x8) and the DDR boundary is every 4Bx8=32B(0x20).

The master expects to write the data to the following addresses: 0x5 (with WSTRB=0xE0), 0x8 (till 0xF).

The MMDC will issue one access toward the DDR as follows:

Write access toward logic address with suffix 0x0 (DDR boundary is 0x20 and 0x0 is the closest to 0x0) while address 0x0 till 0x4 are masked by DM (data masking signal) and address 0x10 till 0x1F are also masked by DM.

33.4.4.3.5 Example 5

Assume AXI write access has the following attributes:

- Increment (awburst[1:0]=2'b01)
- AXI size of 64bits (awsize[2:0]=3'b011)
- AXI length of 7 (awlen[3:0]=4'b0001)
- AXI address with suffix 0x10 (aligned)

Toward DDR3 (MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x64 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case the AXI alignment is every 8B (0x8) and the DDR boundary is every 8Bx8=64B(0x40).

The master expects to write the data to the following addresses: 0x10, 0x18, 0x20, 0x28, 0x30, 0x38, 0x40, 0x48.

Because the AXI access is not aligned to DDR boundary, which is every 0x40, the MMDC will issue two accesses toward the DDR as follows:

- Write access toward logic address with suffix 0x0 while address 0x0 till 0xF are masked by DM (data masking signal).

NOTE

Logic address is the address of the column normalized to 1 byte.

- Write access towards logic address with suffix 0x40 while addresses 0x50 till 0x7F are masked by DM (data masking signal).

33.4.4.4 Address mirroring

When enabling this feature, address bits DRAM_A3, DRAM_A4, DRAM_A5, DRAM_A6, DRAM_A7, DRAM_A8, DRAM_SDBA0, and DRAM_SDBA1 behave differently according to the associated chip select.

This feature facilitates PCB board routing for devices on chip select 1, which are typically populated on the opposite side of the PCB from the devices on chip select 0.

NOTE

This feature is only supported for DDR3 memories. It is not supported for LPDDR2 memories.

NOTE

This feature will not be supported for DDR3 since only a single chip select is supported for DDR3

The following table specifies the address mirroring options:

Table 33-6. Address mirroring options

MMDC pin	Chip select 0 pin	Chip select 1 pin
DRAM_A3	DRAM_A3	DRAM_A4
DRAM_A4	DRAM_A4	DRAM_A3
DRAM_A5	DRAM_A5	DRAM_A6
DRAM_A6	DRAM_A6	DRAM_A5
DRAM_A7	DRAM_A7	DRAM_A8
DRAM_A8	DRAM_A8	DRAM_A7
DRAM_SDBA0	DRAM_SDBA0	DRAM_SDBA1
DRAM_SDBA1	DRAM_SDBA1	DRAM_SDBA0

33.4.5 LPDDR2 and DDR3 pin mux mapping

The following table shows the pin mux mapping between LPDDR2 and DDR3. The i.MX DDR I/O pads corresponds with the DDR3 standard.

- In DDR3, all DRAM_DATA, DRAM_SDQS, and DRAM_DQM data lines work with channel 0.
- In LPDDR2, DRAM_DDQS[3:0], DRAM_DATA[31:0] and DRAM_DQM[3:0] work with channel 0. DRAM_SDQS[7:4], DRAM_DATA[63:32], and DRAM_DQM[7:4] work with channel 1.

Table 33-7. LPDDR2 and DRAM pin mux mapping

DRAM I/O pad	LPDDR2 I/O pad
DRAM_ADDR00	LPDDR2_CA0
DRAM_ADDR01	LPDDR2_CA1
DRAM_ADDR02	LPDDR2_CA2
DRAM_ADDR03	LPDDR2_CA3
DRAM_ADDR04	LPDDR2_CA4
DRAM_ADDR05	LPDDR2_CA5
DRAM_ADDR06	LPDDR2_CA6
DRAM_ADDR07	LPDDR2_CA7
DRAM_ADDR08	LPDDR2_CA8
DRAM_ADDR09	LPDDR2_CA9
DRAM_ADDR10	—
DRAM_ADDR11	—
DRAM_ADDR12	—
DRAM_ADDR13	—
DRAM_ADDR14	—
DRAM_ADDR15	—
DRAM_CAS_B	—
DRAM_RAS_B	—
DRAM_WE_B	—
DRAM_SDCKE0	LPDDR2_CKE0
DRAM_SDCKE1	LPDDR2_CKE1
DRAM_CS_B0	LPDDR2_CS_B0
DRAM_CS_B1	LPDDR2_CS_B1
DRAM_ODT0	LPDDR2_ODT0
DRAM_ODT1	LPDDR2_ODT1
DRAM_SDCLK0_P	LPDDR2_CK0
DRAM_SDCLK1	LPDDR2_CK1
DRAM_BA0	—
DRAM_BA1	—
DRAM_BA2	—

33.4.6 Power Saving and Clock Frequency Change modes

33.4.6.1 Power saving general

MMDC supports multiple DDR power saving modes.

NOTE

At default, the power saving modes are disabled. These modes may dramatically decrease the power consumption of DDR memories.

1. Self-refresh entry to the entire DDR device (for both chip select 0 and 1) can be activated through two mechanisms:
   - LPMD (Low Power Mode)
     - Hardware handshaking (LPMD/LPACK) with the clock module in the system
     - Software handshaking by setting the field MAPSR[LPMD] and polling MAPSR[LPACK]
     - Automatic entry by configuring the amount of idle cycle for triggering self-refresh entry through MAPSR[PST] and by clearing MAPSR[PSD]
   - DVFS (Dynamic Voltage and Frequency Change)
     - Hardware handshaking (DVFS/DVACK) with the clock module in the system
     - Software handshaking by setting the field MAPSR[DVFS] and polling MAPSR[DVACK]

NOTE

If hardware or software requests for self-refresh entry were detected by the MMDC (even before the assertion of the LPACK), no write or read accesses will be acknowledged utnil the deassertion of those requests.

2. Automatic active/precharge power down entry to a specific chip select can be activated by configuring the ESDPDC register:
   - PWDT_0/PWDT_1 - define the number of idle cycles before entering power down, can be different value per chip select.
   - SLOW_PD - In case of DDR3 memory is configured to use slow precharge power down then this bit should be set as well.
   - BOTH_CS_PS - The MMDC can either set each chip select independently to power down, according to its idle state, or set both chip selects to power down only if both in idle state for the configured period.
   - Few paramters must be configured in addition:

- Timing parameters at MDCFG0[tXP and tXPDLL].
- ODT timing at MDOTC[tAOFPD, tAONPD, tANPD and tAXPD]

NOTE

It is possible to enter certain chip selects to low power consumption while the second chip select is activated.

3. Automatic precharge of all DDR banks to a specific chip select. Can be activated by configuring ESDPDC fields: PRCT_0 and PRCT_1. Each field determines a value loaded to a different chip select.

33.4.6.2 Self refresh and Frequency change entry/exit

As described in Power saving general, the MMDC supports two mechanisms that will cause the DDR device to enter self-refresh mode:

- LPMD (Low Power Mode) - For power saving purposes
- DVFS (Dynamic Voltage and Frequency Change) - For clock frequency changes

While the DDR device is in self-refresh mode, there is no need to provide periodic refresh commands.

The MMDC treats hardware/software handshaking of LPMD/DVFS in the same manner:

- Upon the assertion of LPMD/DVFS request, the following is done:
  - The MMDC blocks any further AXI accesses even before the acknowledge is asserted
  - Completes all opened AXI accesses
  - Closes (precharge) all banks in the appropriate timing
  - Drives self-refresh command by deasserting clock enable signal (DRAM_SDCKE is driven to "0") together with a refresh command. This occurs after satisfying tRP/tRPA from the precharge all command.
  - Deasserts the clock (CK) that is driven to the DDR device
  - Asserts LPMD/DVFS acknowledge (LPACK/DVACK)
  - Allows deassertion of the operating clock of the MMDC (AXI clock)
- Upon the deassertion of LPMD/DVFS request, the following is done:
  - Operating clock of the MMDC must be turned on before LPMD/DVFS is deasserted
  - Starts driving the clock (CK) to the DDR device
  - After satisfying tCKSRX from clock renewal the clock enable signal (DRAM_SDCKE) is asserted
  - LPMD/DVFS acknowledge (LPACK/DVACK) is deasserted

- After satisfying tXS from the assertion of DRAM_SDCKE, a refresh command is driven to the DDR device.
- If ZQ calibration is enabled then tRFC is satisfied from the refresh command and a long ZQ command is driven.
- tZQoper idle cycles are counted after the ZQ command.
- After satisfying tDLLK from the assertion DRAM_SDCKE, the MMDC returns to normal operation.

The figure below shows the timing diagram of the hardware/software handshaking of LPMD/DVFS:

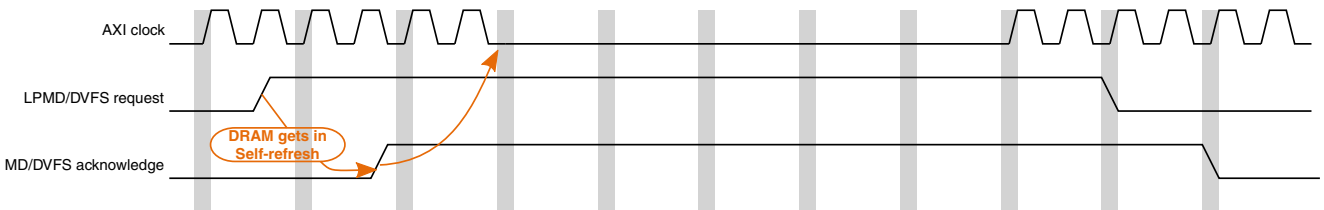


Figure 33-4. LPMD/DVFS Hardware/Software Handshaking

Note for self-refresh:

- As soon as LPMD or DVFS requests are detected by either hardware or software handshaking, the MMDC will deassert the AXI ARREADY/AWREADY signals immediately to block further requests from the system.
- In case of automatic self-refresh, the internal operating clock will be negated to save power.

33.4.7 Reset

33.4.7.1 Hard reset

When hard reset is asserted (aresetn is driven to "0") while warm reset is deasserted (warm_reset is driven to "0"), the entire MMDC will be initialized, including configuration/status registers and state machines.

In order to access the DDR device, the MMDC will then have to be reconfigured.

33.4.7.2 Warm reset

The MMDC supports warm reset signal. The warm reset signal must envelop the hard reset signal and then the MMDC will reset all the internal registers. The only registers that are not reset are those that are essential for returning it to normal operation without repeating the initialization sequence and without losing data stored in the memory (configuration/status registers won't be initialized).

For the successful operation of warm reset, the following steps must be performed:

- The MMDC must enter self-refresh mode. This can be achieved by either LPMD or DFVS requests
- Wait for LPMD or DVFS acknowledge
- Assert warm reset signal (i.e. drive warm_reset to "1")
- Assert hard reset signal (i.e. drive aresetn to "0")
- Deassert hard reset signal
- Deassert warm reset
- Get out of the LPMD/DVFS mode

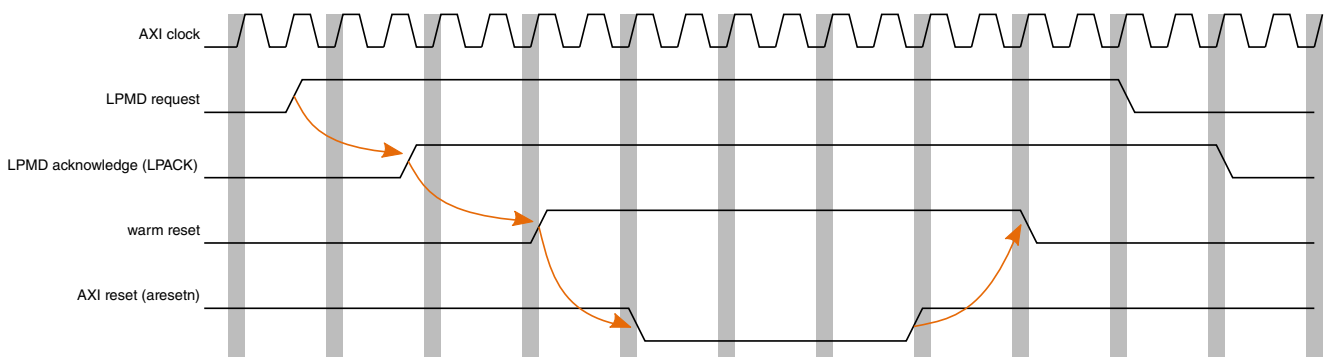


Figure 33-5. Warm Reset Diagram

33.4.7.3 Software reset

The MMDC supports software reset. When software reset is configured then the MMDC will reset all the internal registers except those that are essential for returning to normal operation without repeating the initialization sequence or without losing data stored in the memory (configuration/status registers won't be initialized).

The following steps should be performed for successful operation of software reset:

- The MMDC should enter self-refresh mode. This can be achieved by either LPMD or DFVS request.
- Wait for LPMD or DVFS acknowledge

- Assert software reset, by setting MDMISC[RST]
- Get out of the LPMD/DVFS mode

Normal operation can be resumed.

33.4.8 Refresh Scheme

The MMDC supports various automatic refresh options which can be configured via the MDREF register.

The periodic auto refresh can be triggered by the following clocks:

- 32 kHz clock
- 64 kHz clock
- MMDC operating clock

The refresh scheme of the MMDC is flexible and allows the system to configure the desired AXI accesses delay/latency in each refresh cycle.

The table below shows an example of four configurations of the refresh cycles that will be handled by the MMDC. Each configuration meets a refresh rate of 3.9us (tREFI, refresh command every 3.9µs).

Table 33-8. MMDC Refresh Scheme

Option number	Description	REFR	REF_SEL	REF_CNT	DDR hang time
1	Issue 8 refresh commands every 31,250 ns	0x7 (8 refreshes)	0x0 (64 kHz)	not needed	tRFC * 8
2	Issue 4 refresh commands every 15,625ns	0x3 (4 refreshes)	0x1(32 kHz)	not needed	tRFC * 4
3	Issue 2 refresh commands every 7800ns	0x1(2 refreshes)	0x2 (REF_CNT)	7800/2.5 = 3120 (0xC30)	tRFC * 2
4	Issue 1 refresh command every 3900 ns	0x0 (1 refresh)	0x2 (REF_CNT)	3900/2.5 = 1560(0x618)	tRFC

33.4.9 Burst Length options towards DDR

The MMDC supports burst lengths which can be configured through MD+CTL[BL] as follows:

- In DDR3 mode, only burst length 8 can be used.
- In LPDDR2 mode, only burst length 4 can be used.

In DDR3 mode read/write accesses to the DDR are always 8 words (x16) and aligned in according to JEDEC standards.

In case of AXI INCREMENT, accesses that are not aligned the irrelevant data is masked in write accesses and ignored in read accesses. In case of AXI WRAP accesses, even if the access is not aligned, then the MMDC provides an internal optimization mechanism for better efficiency of the DDR data bus.

33.4.10 Exclusive accesses handling

The MMDC contains four exclusive monitors, each for dedicated ID as configured in MAEXIDR0 and MAEXIDR1.

- If legal read exclusive is received by the MMDC, the associated monitor is turned on.
- While the monitor is turned on upon legal write exclusive, the monitor will be turned off and the write will be completed successfully with EXOKAY.
- The following rules must be met for successful exclusive access:
  - Aligned access (the AXI address is aligned to the AXI size)
  - AXI single access (AXI burst length isn't greater than 1)
  - AXI size of up to 64 bits
  - AXI non-cachable access (i.e. ARCACHE[1]/AWCACHE[1] is equal "0" or ARCACHE[1]/AWCACHE[1] is equal "1" while ARCACHE[3:2]/ AWCACHE[3:2] are equal "00")
  - AXI ID that matches one of the four exclusive IDs

Exclusive read behavior (first bullet also correct for non-exclusive accesses):

- In case of security violation, the read is blocked and is not sent to DDR. There are two options for response:
  - If ARCR_SEC_ERR_EN (MAARCR[30]) is high, SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.
- If AXI exclusive rules violation occurs (as described above), the read access is not blocked and is sent to DDR. The data will be fetched and be driven to the master, but the type of response may be unpredicted.
- If none of the above occurs, the read is sent to the DDR. The exclusive monitor will be turned on and the response is ExOKAY
- If additional legal AXI read exclusive is received with the same ID before the AXI exclusive write, the monitor will be updated with the latest attributes.

Exclusive write behavior (first bullet also correct for non-exclusive accesses):

- In case of security violation, the write is blocked and is not sent to DDR, but the monitor will be kept on. There are two options for response:
  - If ARCR_SEC_ERR_EN (MAARCR[30]) is high then SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.
- In case of AXI exclusive rules violation (as described above), the write is blocked and is not sent to DDR. In that case the type of response may be unpredicted.
- In case the exclusive write access has different AXI attributes, but the same ID as the read exclusive access, the write is blocked and is not sent to DDR and the monitor will be turned off. There are two options for response:
  - If ARCR_EXC_ERR_EN (MAARCR[28]) is high then SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.
- In case of regular (non exclusive) write access is received to the same address or overlapping addresses then the write will be sent to the DDR and the monitor will be turned off.
- In case of legal write exclusive access is received with the same attributes as the read exclusive access while the monitor is on (no write accesses occured to the same address between the read exclusive and write exclusive), then the write is sent to DDR and the response is EXOKAY. But, if the legal write exclusive is received while the monitor is off, the write is blocked and there are two options for response.
  - If ARCR_EXC_ERR_EN (MAARCR[28]) is high then SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.

33.4.11 AXI Error Handling

The MMDC supports the AXI responses listed here.

- In case of AXI exclusive violation there are two options for response:
  - If MAARCR[28] is high then SLVError is issued towards the Master, Otherwise OKAY response is sent to the Master

NOTE

In case of read error MMDC drives zeros on the read data bus

33.5 Performance

33.5.1 Arbitration and reordering mechanism

33.5.1.1 Arbitration General

The following specifies arbitration and reordering flow in MMDC towards the DDR.

- AXI read and write accesses are sampled in the associated queue.
- Read/write arbitration is handled to select the winning access.
- Winning access is sampled in the reordering queue
- Reordering mechanism is handled between valid requests that reside in the reordering queue to select the access that will be dispatched to the DDR.
  - The reordering is held in order to optimize the accesses and to maximize the utilization of the DDR bus
  - As soon as the reordered access is completed (indicated by end of response or data phase) then it is erased from the associated queue and the MMDC is ready to receive the next available access from the master

In general, the reordering/arbitration mechanism is based on dynamic priority mechanism, which compares dynamic priorities between valid entries in the reordering queue and issues the entry with highest dynamic priority towards the DDR Logic.

The selection of the winning access is based on two modes, which can be activated together, as following:

- Real time channel mode:
  - Accesses with QoS='f' (i.e. awqos[3:0]/arqos[3:0] = "f") will bypass all other requests towards the DDR
- Dynamic scoring mode:
  - The arbitration mechanism is based on dynamic priority. Relevant for the accesses with QoS smaller than 'f' or when real time channel mode is disabled.

NOTE

Due to re-ordering and optimization mechanism (per different AXI ID), the transactions towards the DDR may be driven in a different ID order they were received by the AXI master. In similar way, the write response, read response or read data may be driven to the AXI master in a different ID order.

33.5.1.2 Real time channel mode

When real time mode is enabled (i.e MAARCR[ARCR_RCH_EN] = "1") , all requests with QoS='f' (i.e. awqos[3:0]/aqqos[3:0] = "f") will bypass all other pending accesses towards the DDR. This mode is enabled by default.

33.5.1.3 Dynamic scoring mode (Arbitration Winning Conditions)

The arbitration between pending accesses in the MMDC is handled according to a dynamic priority of each access.

The dynamic priority (may be also called score) is calculated according to a sum of some factors (final_score[3:0]), where part of them may be updated dynamically. The following will specify each scoring factor:

- MAARCR[ARCR_PAG_HIT] (Page hit score) - A static score which is taken into account in case the pending access has a page hit
- MAARCR[ARCR_ACC_HIT] (Access hit score) - A static score, which is taken into account in case the current access type (read/write) is the same as the access that has been dispatched to the DDR previously
- MAARCR[ARCR_DYN_JMP] (Dynamic jump score) - A dynamic score which is given to any pending access in case it was not chosen in the arbitration. The dynamic jump counter is limited by maximum value which is set in MAARCR[ARCR_DYN_MAX] .
- QoS score which is indicated through a sideband 4bits AXI signals (awqos[3:0]/ aqqos[3:0]) and is driven by the AXI master per access

Note: In order to prevent an overflow in the total sum of scores, a clipping is held and selects the maximum score value of 'f' once a total scores sum is greater than 'f'.

The figure below shows the dynamic score calculations

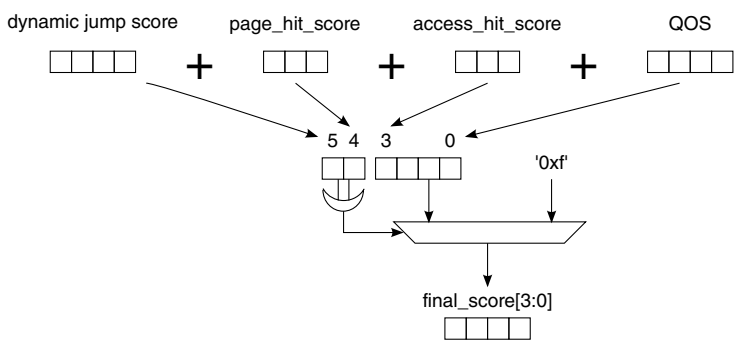


Figure 33-6. Dynamic score/priority calculation

33.5.1.4 Guarding (aging) mechanism

The guarding mechanism (may be also called aging) is used to prevent a starvation of accesses.

As soon as the dynamic jump score reaches its maximum value (MAARCR[ARCR_DYN_MAX]) then each time a pending request was not chosen in the arbitration, the "guarding" counter is incremented by 1. When the "guarding" counter reaches its predefined value, set in MAARCR[ARCR_GUARD], the associated request gets the highest priority and will be chosen in the next arbitration cycle towards the DDR unless a real time channel (i.e access with QoS ="f") is arrived.

Note: In case real time channel has arrived then the dynamic score of the non real time channels won't increment in order to prevent a case where the "guarding" counter of more than one access has reached its limit.

33.5.2 Prediction mechanism

When prediction mechanism is enabled (i.e by configuring MDMISC[MIF3_MODE]) then the MMDC predicts the chip-select, bank address and row address that is going to be issued towards the DDR before the access is physically dispatched towards DDR device.

That mechanism enables to prepare the DDR device with future accesses and improves the overall DDR performance.

This prediction mechanism operates in parallel to the reordering mechanism and may yield a prediction based on 3 levels of pending accesses:

1. Access in first stage of pipeline.
2. Valid access on AXI bus either read channel or write channel.
3. Valid access on special bus from arbitration - this access is chosen by the arbitration as the next miss access in its buffers

33.5.3 Special Optimization for accesses towards DDR3

In case an AXI read/write wrap non-aligned access is acknowledged in DDR3 mode with the same wrap boundary as the DDR wrap boundary then the MMDC will make an optimization and issue only one access towards the DDR, although all the accesses towards the DDR3 must be aligned.

For example: AXI write access with size of 128bits (awsize[2:0]=3'b100), length of 4 (awlen[3:0]=4'b0011) towards DDR3 x64 (burst length 8). In that case the AXI wrap boundary is 16Bx4=64B (0x40) and the DDR3 wrap boundary is 8Bx8=64B (0x40). If, for example, the AXI access is towards AXI address with suffix of 0x10 (non-aligned to 64B boundary) then the MMDC will get from the AXI master the data that is associated with addresses 0x10, 0x20, 0x30, 0x0. The MMDC will rearrange internally the data so it

will match DDR3 alignment as following: 0x0, 0x10, 0x20, 0x30 and drive it in one access towards the DDR to address 0x0. The alternative was to issue two accesses towards the DDR with address 0x0 with different data masking

NOTE

In read wrap access the same optimization is handled, while as soon as the critical AXI word is fetched from the DDR then it is driven immediately to the AXI master without buffering. Based on the example above, the master expects to fetch first the data that is associated with address 0x10. Therefore the MMDC will issue read access from address 0x0 of the DDR and as soon as the data that is associated with address 0x10 is received then it will be driven back immediately to the master even before fetching the data of the furhter addresses.

33.6 MMDC Debug

33.6.1 Hardware debug monitor

The MMDC has a hardware debugging mechanism that monitors each access that is driven to the MMDC. Every time this mechanism is enabled (setting of MADPCR0[DBG_EN] to "1") then each access that will be dispatched to the DDR will be also observed in the I/O pads (i.e. over ipp_do_ddr_debug[50:0]). The content of this bus is described in the table below.

Table 33-9. Hardware monitor debugging

Signal Name	Number of Bits	Description
acc_addr	[31:0]	AXI ADDRESS of the selected access
acc_type	1	access type of the selected access. "0 indicates write. "1" indicates read.
acc_id	[15:0]	AXI transaction ID of the selected access
valid_strobe	1	indication for a valid request . This signal will be asserted for 1 clock cycle

The fields above are organized as following:

MMDC_DEBUG[50:0] = {1'b0,valid_strobe,acc_id,access_type,addr}

These signals are sent to IOMUX, in IOMUX user can configure it to be output from the chip for debug usage.

33.6.2 Step By Step (SBS) software monitor

The MMDC has a Step By Step (SBS) software debugging mechanism that monitors each access that is driven to the MMDC. Every time this mechanism is trigerred then one AXI access will be dispatched to the DDR and in parallel its attributes will be observed in a status register.

Once the "step by step" is enabled (i.e. MADPCR0[SBS_EN] is "1") then all accesses to the DDR device will be halted. This SBS feature is used during DDR SDCLK frequency scaling to prevent any accesses to the DDR device during the transition. Setting MADPCR0[SBS] to "1" will dispatch the access that is pending in the head of the MMDC queue (read or write). Upon every setting of MADPCR0[SBS]:

- The AXI attributes of the access will be sampled in the associated MASBS0 and MASBS1 fields
- MADPCR0[SBS] will be cleared automatically.

Setting again MADPCR0[SBS] to "1" will dispatch the next pending access in the MMDC queue.

NOTE

Setting the ARCR_ARB_REO_DIS bit in the MMDC0_MAARCR Register disables the SBS function

33.7 MMDC Profiling

The profiling mechanism provides the ability to calculate the DDR utilization together with read and write accesses statistics towards DDR per given period of time.

MMDC supports the following profiling counters:

- MADPSR0 (Total cycles count) - Indicates the total amount of cycles of the profiling period (up to 2^32 cycles)
- MADPSR1 (Busy cycles count) - Indicates the total busy cycles during the profiling period. Busy cycles are any MMDC clock cycles where the internal state machine is not idle. If any read or write requests are pending in the FIFOs, the MMDC is not idle.
- MADPSR2 (Total read accesses count) - Indicates the total read accesses towards MMDC during the profiling period
- MADPSR3 (Total write accesses count) - Indicates the total write accesses towards MMDC during the profiling period

- MADPSR4 (Total read bytes count) - Indicates total bytes that were read from MMDC during the profiling period
- MADPSR5 (Total write bytes count) - Indicates total bytes that were written to MMDC during the profiling period

All profiling items described above are disabled by default. The following describes how to control the profiling mechanism:

- MADPCR0[DBG_EN] enables profiling.
- MADPCR0[PRF_FRZ] stops/freezes the profiling for example in case user wishes to perform DDR profiling per specific task. In order to resume profiling then MADPCR0[PRF_FRZ] should be cleared.
- MADPCR0[DBG_RST] clears all profiling counters
- MADPCR0[CYC_OVF] indicates whether an overflow occured in the total cycles counter (i.e. total amount of cycles are greater than 2^32) . This field can only cleared by writing '0'.

Read/Write statistics can be collected per specific AXI ID (16bits). The following fields in MADPCR1 register determines which AXI-ID or AXI-ID's to monitor:

- PRF_AXI_ID defines which AXI IDs are taken for profiling. Default values is 16'h0.
- PRF_AXI_ID_MASK defines which bits from PRF_AXI_ID will be compared with AXI ID of read/write access. "1" means to monitor the associated bit and "0" means don't care. Default value is 16'h0000, meaning all IDs are not monitored

So the AXI-IDs to be monitored are calculated according to the following equation:

(AXI-ID & PRF_AXI_ID_MASK) Xnor (PRF_AXI_ID & PRF_AXI_ID_MASK)

For example if AXI ID's between A100 till A1FF are wished to be monitored then the following should be configured:

- PRF_AXI_ID = A100
- PRF_AXI_ID_MASK = FF00

33.8 LPDDR2 Refresh Rate Update and Timing Derating

LPDDR2 devices may have a temperature sensor that is used to determine an appropriate refresh rate and whether AC timing derating is required. The status of the temprature sensor can be read through MRR command from LPDDR2 MR4 register.

The MMDC supports refresh update and timing derating mechanism on the fly. The following steps specify how to use that mechanism:

- Perform periodic polling on MR4 LPDDR2 register using MRR command

- Read MDMRR register and analyze the MR4 indication
- In case refresh rate update and/or AC timing derating is required then it is needed to update MDREF and/or MDMR4[tRCD_DE, tRC_DE, tRAS_DE, tRP_DE, tRRD_DE] parameters

NOTE

MDMR4[tRCD_DE, tRC_DE, tRAS_DE, tRP_DE, tRRD_DE] are referred to the associated values configured at MDCFG3LP[tRC_LP, tRP_LP, tRCD_LP], MDCFG1[tRAS], MDCFG2[tRRD]

- Assert MDMR4[UPDATE_DE_REQ]
- When the MMDC switch to the new values then an acknowledge will be indicated at MDMR4[UPDATE_DE_ACK]

The enabled power saving features may be interfering with the MR4 reads. When these features are enabled the MMDC will automatically enter self-refresh and hence prevent the MR4 reads. Hence the Power Saving and refresh functionality should be disabled why SW performs the MR4 reads.

To disable the “Power Down” mode, set bit 0 of MAPSR (0x021B0404) to 1’b.
To disable the “Self-Refresh Power Down Timer” mode, set bits [15:8] of MDPDC (0x021B0004) to 0x00.

Overall Sequence to Perform MR4 reads

1. Set MAPSR to 0x00010107
2. Set MDPDC to 0x00020076
3. MRR command to read MR4
4. Write MDPDC to 0x00025576 to re-enable this feature
5. Write MAPSR to 0x00010106 to re-enable this feature

33.9 Changing MMDC SDCLK Frequency

Changing the MMDC SDCLK frequency when using either DDR3 or LPDDR2 can be done dynamically in software using the DVFS. Several important factors must be considered for each type of memory and whether transitioning between high and low frequency. The following sections highlight these factors and provide a step-by-step process when making frequency transitions on each type of memory.

33.9.1 Changing MMDC SDCLK Frequency Using DDR3/DDR3L Memories

This section provides the sequence of steps which should be followed to ensure a successful frequency transition when using DDR3/DDR3L memories.

NOTE

There are additional steps required to ensure that the ARM core and associated caches are in the correct state prior to any frequency transition or low power entry which are not detailed here. The user must assure that the ARM core or other bus masters are not accessing DDR during these transitions.

There are two general categories of frequency transitions – transitioning from high frequency to low frequency, and vice versa. High frequency is defined as all settable frequencies between 125 MHz and the maximum bus frequency, and low frequency is defined as all settable frequencies between 125 MHz and the low-power mode frequency of 24 MHz. Setting to a low frequency (below 125MHz) requires that the DDR3 device's delay line (DLL) be switched off, or be in DLL Off mode. Setting to a high frequency (above 125 MHz) requires that the DLL be switched on, or be in DLL On mode.

For further details refer to the DLL-off Mode chapter in the JEDEC standard.

A special consideration in changing the MMDC frequency on i.MX 6DualPlus/ 6QuadPlus is that all transitions are performed using the MMDC's Reorder Pass-through mode. Options for reordering and arbitration are selected by the ARCR_REO_DIS (bit 26) and ARCR_ARB_REO_DIS (bit 25) bits in the MMDC Core AXI Reordering Control Register (MMDC0_MAARCR). Two modes have been added in MMDC to disable the reorder function. The following table shows the selection of settings for reordering based on these bits (bit[26:25]) of the MMDC0_MAARCR register.

Table 33-10. Arbitration and Reordering Control in MMDC0_MAARCR

Register Bits	Settings	Notes
ARCR_REO_DIS MMDC0_MAARCR[25]	0 - MMDC reordering controls enabled 1 - MMDC reordering controls disabled (bypassed)	**MMDC Reorder Pass-through mode**: reordering is disabled when this bit is set.
ARCR_ARB_REO_DIS MMDC0_MAARCR[26]	0 - MMDC arbitration and reordering controls enabled 1 - MMDC arbitration and reordering controls disabled (Recommended)	**Adopt_Bypass mode**: MMDC reorder and arbitration pipeline is totally bypassed when this bit is set.

When the ARCR_REO_DIS bit (MMDC0_MAARCR[25]) is set, the MMDC Reorder Pass-through mode is enabled. In this mode the DRAM access commands are still processed by the MMDC reorder and arbitration pipeline, but the commands are not

reordered. When the ARCR_ARB_REO_DIS bit (MMDC0_MAARCR[26]) is set the MMDC reorder and arbitration pipeline is totally bypassed. In this Adopt_Bypass mode the DRAM access latency is reduced and performance is improved.

The MMDC Reorder Pass-through mode is used for all frequency transitions, and ARCR_REO_DIS is set prior to the transition. In addition, this mode is used after transition for all frequency settings except for the maximum setting which, for the i.MX 6DualPlus/6QuadPlus, is 528 MHz. For this frequency setting, both arbitration and reordering are bypassed in the Adopt_Bypass mode.

The below table shows the relationship between the ARCR_REO_DIS and ARCR_ARB_REO_DIS bits.

Table 33-11. ARCR_ARB_REO_DIS and ARCR_REO_DIS bits of the MMDC0_MAARCR Register

ARCR_ARB_REO_DIS	ARCR_REO_DIS	Function	Notes
0	0	Reset Value – Both arbitration and reordering enabled	This is default value out of reset
0	1	Reordering only disabled	Reorder Pass-through mode enabled
1	x	Both arbitration and reordering disabled	Adopt_Bypass mode enabled. ARCR_REO_DIS has no impact since all AXI transactions bypass the reordering logic.

The recommended setting of these two bits based on frequency transition is shown in the following table.

Table 33-12. Required ARCR_ARB_REO_DIS and ARCR_REO_DIS Settings before and after Frequency Transitions

Transition	Before Transition MMDC0_MAARCR [26:25][1]	After Transition MMDC0_MAARCR [26:25]	Comments
Initialized to 528 MHz at system boot	-	10[2]	Set to Adopt_Bypass mode after transition to 528 MHz during system boot
to other than 528MHz (including 24MHz)	01	01	Set to Reorder pass-through mode after transition
to 528MHz	01	10	Set to Adopt_Bypass mode after transition

1. Set to Reorder pass-through mode before frequency transition.
2. Set during MMDC Initialization at boot up for optimal performance.

In addition to setting the ARCR_REO_DIS bit prior to the MMDC frequency transition, the Step-by-Step (SBS) monitor (see section 46.6.2) is also enabled to block DDR accesses during the frequency transition. Setting the ARCR_ARB_REO_DIS bit disables the SBS function, therefore the Adopt_Bypass mode is used during transitions to allow the SBS logic to be enabled. The SBS function is only used during actual frequency transitions. Once frequency has switched the SBS is disabled to unblock DDR accesses.

NOTE

For DDR3 there is no need to perform any configuration of the MMDC1 registers for channel 1.

For further information please refer to EB828: MMDC and NoC Configuration for Optimal DDR3 Performance on the i.MX 6DualPlus/6QuadPlus available on NXP.com.

33.9.1.1 High Frequency to Low Frequency Transition (DLL On to DLL Off)

When switching from high frequency, including the maximum frequency, to low frequency the following steps should be executed.

NOTE

Unless specified otherwise, all registers cited in the switching procedures in this section and the following sections are in the MMDC0 bank.

1. Disable power-down timers that can conflict with this sequence, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
2. Assert the MDSCR[CON_REQ] and wait for the MDSCR[CON_ACK] assertion.
3. Execute PRECHARGE ALL banks command (MDSCR[CMD] = 3b’101).
4. Execute MRW command to MR1 and disable RTT_Nom (A9, A6, A2 = 0) and DLL ON (A0 = 1, disabling the DLL)
5. Disable reordering (MAARCR[26:25] = 2b’01).
6. Execute MRW command to MR2 in order to update CWL to 6.
7. Execute MRW command to MR0 in order to update CL to 6.
8. Enter self-refresh mode. (See Self-refresh and Frequency change entry/exit.)
9. De-assert MDSCR[CON_REQ] signal.
10. Wait for self-refresh entry acknowledge (MAPSR[LPACK] = 1).
11. Bypass the Measurement Unit (set MPMUR0[MU_BYP_EN]) and follow steps outlined in ERR005778.
12. Perform a force measure by setting MPMUR0[FRC_MSR]. Wait for force measurement acknowledge MPMUR0[FRC_MSR] =0.

13. Change to low frequency. (See Self-refresh and Frequency change entry/exit.)
14. Enable Step-by-Step Mode (MADPCR0[SBS_EN] = 1).
15. Keep reordering disabled (MAARCR[26:25] = 2b'01)
16. Exit self-refresh mode.
17. Assert MDSCR[CON_REQ] and wait for MDSCR[CON_ACK] assertion.
18. Update tCWL =6 and tCL =6 to meet the values configured in the DDR device. (MDCGFG0, MDCFG1)
19. Set read latency (MDMISC[RALAT] = 2).
20. Enable pull-down resistors on DQS (through the I/O-MUX).
21. Disable ODT resistor (MPODTCTRL = 0).
22. Disable DQS gating (MPDGCTRL0[DG_DIS] = 1).
23. Disable Step-by_Step Mode (MADPCR0[SBS_EN] = 0).
24. De-assert MDSCR[CON_REQ].
25. Enable required power-down timers that were disabled, such as: MAPSR[PSD], MDPDC[PWDT_1] , MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].

33.9.1.2 Low Frequency to High Frequency Transition (DLL Off to DLL On)

When switching from low frequency, including 24MHz low power mode, to high frequency the following steps should be executed.

1. Disable power-down timers that can conflict with this sequence, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
2. Assert MDSCR[CON_REQ] and wait for MDSCR[CON_ACK] assertion.
3. Execute PRECHARGE ALL banks command (MDSCR[CMD] = 3b'101).
4. Enter self-refresh mode. (See Self-refresh and Frequency change entry/exit.)
5. De-assert MDSCR[CON_REQ].
6. Wait for self-refresh entry acknowledge (MMDCx_MAPSR[LPACK] = 1).
7. Disable reordering (MAARCR[26:25] = 2b'01).
8. Change to high frequency. (See Self-refresh and Frequency change entry/exit.)
9. Enable Step-by-Step Mode (MADPCR0[SBS_EN] = 1).
10. Exit self-refresh mode.
11. Optionally, force ZQ automatic calibration process by setting MPZQHWCTRL[ZQ_HW_FOR] = 1 and setting MPZQHWCTRL[ZQ_MODE] = 2b'11.)
12. Enable DQS gating (MPDGCTRL0[DG_DIS] = 0)
13. Restore ODT setting in MPODTCTRL to original value.
14. Enable Measurement Unit by clearing MPMUR0[MU_BYP_EN]) per ERR005778.

15. Perform a force measure by setting MPMUR0[FRC_MSR]. Wait for force measurement acknowledge MPMUR0[FRC_MSR] = 0.
16. Disable pull-down resistors on DQS (through the I/O-MUX).
17. Restore read latency (MISC[RALAT]) to original value.
18. Restore calibration registers of MMDC BLOCK (if they were changed).
19. Assert MDSCR[CON_REQ] and wait for MDSCR[CON_ACK] assertion.
20. Execute MRW command to MR1 and enable RTT Nom (A9, A6, A2) and DLL ON (A0 = 0).
21. Execute MRW command to MR0 to reset the DLL (A8) and revert to CL values prior to entry.
22. Execute MRW command to MR2 to update CWL value.
23. Update tCWL and tCL to meet the values configured in the memory. (MDCGFG0, MDCFG1)
24. Execute a ZQ command to the DDR3 device (MDSCR[CMD] = 3b'100).
25. Disable Step-by_Step Mode (MADPCR0[SBS_EN] = 0 .
26. Transition to Adopt_ByPass mode to disable both arbitration and reordering (MAARCR[26:25] = 2b'10) if transitioning to highest frequency; otherwise switch to Reorder Pass-through mode to disable only reordering (MAARCR[26:25] = 2b'01).
27. De-assert MDSCR[CON_REQ].
28. Enable required power-down timers that were disabled, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].

33.9.2 Changing MMDC Frequency Using LPDDR2

This section provides the sequence of steps which should be followed to ensure a successful frequency transition when using LPDDR2 memories.

NOTE

In the procedure sequence provided in the following sections, the user should take caution in adjusting the parameters MMDC MDCFG0[tCL] and MDCFG1[tCWL] to ensure that these parameters match the RL/WL parameter of the LPDDR2 memory devices mode register 2 (MR2). If the MDDC parameters do not match those programmed in the LPDDR2 memory device, the memory interface will become unstable.

33.9.2.1 High Frequency to Low Frequency Transition

When switching from high frequency, including the maximum frequency, to low frequency the following steps should be executed.

1. Disable power-down timers that can conflict with this sequence, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
2. Enter self-refresh mode. (See Self-refresh and Frequency change entry/exit.)
3. Wait for self-refresh entry acknowledge (MMDCx_MAPSR[LPACK] = 1).
4. Disable reordering (MAARCR[26:25] = 2b'01).
5. Enable Step-by-Step Mode (MADPCR0[SBS_EN] = 1).
6. Set read latency (MDMISC[RALAT] = 2).
7. Adjust LPDDR2 timings for low frequency operation: set MDCFG0, MDCFG1, MDCFG2, MDCFG33LP.
8. Read and store the measure count value MU_VAL=MPMUR0[MU_BYP_VAL].
9. Bypass the Measurement Unit by setting MPMUR0[MU_BYP_EN] and follow steps outlined in ERR005778.
10. Double the measure count value for the reduced frequency operation MPMUR0[MU_BYP_VAL] = 2*MU_VAL.
11. Perform a force measurement by setting MPMUR0[FRC_MSR] =1. Wait for force measurement acknowledge MPMUR0[FRC_MSR] =0.
12. Change to low frequency (See 27.4.2. DVFS Change Request Sequence Diagrams)
13. Exit self-refresh mode.
14. Enable required power down timers that were disabled, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
15. Disable Step-by-Step Mode (MADPCR0[SBS_EN] = 0).
16. Keep reordering disabled (MAARCR[26:25] = 2b'01); at max frequency disable both arbitration and reordering (MAARCR[26:25] = 2b'10).

33.9.2.2 Low Frequency to High Frequency Transition

When switching from low frequency to high frequency the following steps should be executed.

1. Disable power-down timers that can conflict with this sequence, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
2. Enter self-refresh mode. (See Self-refresh and Frequency change entry/exit.)
3. Wait for self-refresh entry acknowledge (MMDCx_MAPSR[LPACK] = 1).
4. Disable reordering (MAARCR[26:25] = 2b'01).
5. Enable SBS (MADPCR0[SBS_EN] = 1).
6. Change to high frequency (See 27.4.2. DVFS Change Request Sequence Diagrams)
7. Restore read latency (MISC[RALAT]) to original value.
8. Restore LPDDR2 timings for high frequency operation: set MDCFG0, MDCFG1, MDCFG2, MDCFG33LP.

9. Enable Measurement Unit by clearing MPMUR0[MU_BYP_EN]) per ERR005778.
10. Restore original MPMUR0[MU_BYP_VAL] =MU_VAL.
11. Perform a force measurement MPMUR0[FRC_MSR] =1. Wait for force measurement acknowledge MPMUR0[FRC_MSR] =0
12. Exit self-refresh mode.
13. Enable required power down timers that were disabled, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
14. Disable SBS (MADPCR0[SBS_EN] = 0).
15. Transition to Adopt_ByPass mode to disable both arbitration and reordering (MAARCR[26:25] = 2b'10) if transitioning to highest frequency; otherwise switch to Reorder Pass-through mode to disable only reordering (MAARCR[26:25] = 2b'01).

33.10 ODT Configuration

The MMDC supports one DRAM_ODT signal (DRAM_ODT for each DRAM_CS) in DDR3 mode in order to allow the DDR device to turn on/off its termination resistors. The MMDC suggests various configuration for the assertion of the ODT signal as well as configuration of several related timing.

MDOTC register controls the timing for the DRAM_ODT signals assertion.

NOTE

tODTLon determines the delay between DRAM_ODT signal and the associated RTT, where according to JEDEC standard it equals WL(write latency) - 2. Therefore, the value configured to MDOTC[tODTLon] field should correspond with the value configured to MDCGFG1[tCWL].

In precharge power down mode , when all banks are closed, the assertion of ODT corresponds with tAOFPD and tAONPD which are configured in MDOTC register.

The figure below shows timing diagram of DRAM_ODT and RTT signals while MPODTCTRL[0] is set to "1" (i.e. assertion of DRAM_ODT to the non active DRAM_CS in write access command) and MDOTC[tODTLon] is set to 4.

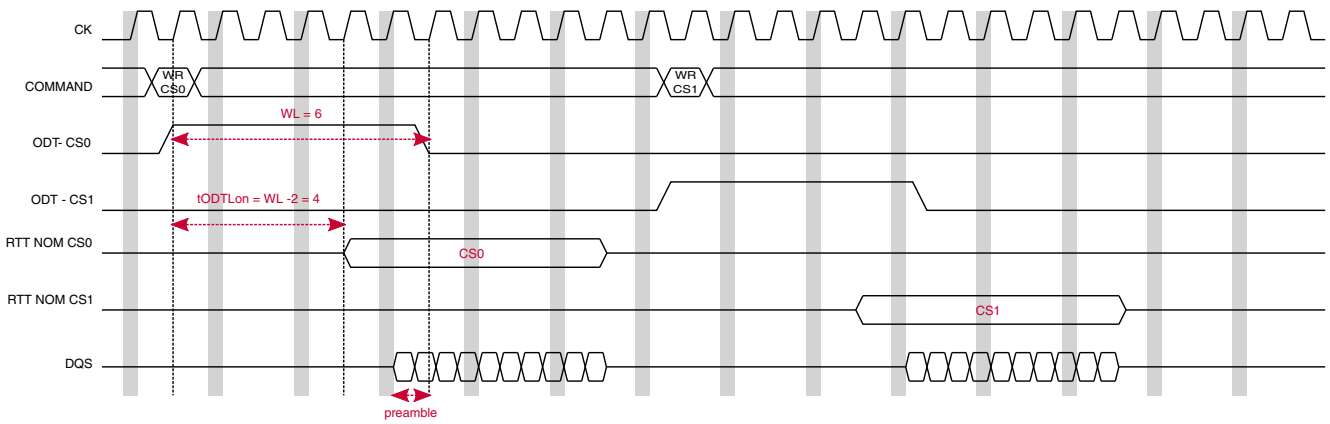


Figure 33-7. ODT - Timing Diagram DDR3 WL=6, BL=8

33.11 Calibration Process

The MMDC offers various calibration processes that are used to obtain better timing accuracy, board skew compensation and I/O pad driving strength adjustment. Each calibration process can be performed either automatically (hardware) or manually (software), though the manual method is typically reserved for debugging purposes. The following calibration processes are supported:

NOTE

Power saving features should be disabled before the calibration process begin. (Such as: MDPDC[PWDT#], MDPDC[PRCT#], MAPSR[PSD])

- ZQ calibration for external DDR device (in DDR3 through ZQ calibration command and in LPDDR2/LPDDR3 through MRW command)
  - Can be handled automatically for ZQ Short (periodically) and ZQ Long (at exit from self-refresh)
  - Can be handled manually at ZQ INIT
- ZQ calibration for i.MX DDR I/O pads for calibrating the DDR driving strength
  - The sequence can be handled automatically by hardware
  - The sequence can be handled step by step manually by software
- Read DQS gating calibration for DDR3 only. Adjustment of DQS gate with read preamble window. For further information refer to Read DQS Gating Calibration
- Read DQS gating calibration not supported for LPDDR2/LPDDR3
- Read data calibration. Adjustment of read DQS with read data byte. For further information refer to Read Calibration
- Write data calibration. Adjustment of write DQS with write data byte. For further information refer to Write Calibration

- Write leveling calibration. Adjustment of write DQS with CK (DDR differential clock). For further information refer to Write leveling Calibration
- Read fine tuning. Adjustment of up to 7 delay-line units for each read data bit.
- Write fine tuning. Adjustment of up to 3 delay-line units for each read data bit.

NOTE

Before starting any calibration process that involves the DDR3 device MPR mode or write leveling calibration, the following should be done:

- Disable the periodic refresh scheme (i.e. setting MDREF[REF_SEL] = "11") and then issue manual refresh command burst by configuring MDSCR[CMD]= 0x2. At the end of the calibration it is needed to enable the periodic refresh scheme.
- Disable the automatic power saving mode (i.e set MAPSR[PSD] = "1").

33.11.1 Delay-line

Each of the calibration processes controls several delay-lines for aligning data and strobes.

By default the delay-line is configured to generate 1/4 clock cycle of delay.

Moreover, when the operating clock is at the maximum allowed frequency, as appeared in the features list, then the delay-line is capable to issue a configurable delay of up to 1/2 clock cycle.

NOTE

At the beginning of the calibration process the initial value of the delay-line must be a valid value (i.e. the strobes must be somewhere among the associated data window) though it might not be the optimal value. The delay-line calibration should be done after Read DQS gating and write-leveling calibrations.

In order to generate an adequate delay during normal operation of the MMDC the delay-line is going through an automatic measurement process during the refresh period of the DDR device

33.11.2 ZQ calibration

The MMDC supports ZQ calibration process to calibrate the driving strength of the DDR I/O pads as well as driving ZQ commands to calibrate the external DDR device driving strength.

The first ZQ calibration (after booting the processor) is performed prior to turning on the MMDC. Subsequent ZQ calibrations may be executed in parallel to the DDR ZQ calibration. The MMDC supports 2 types of ZQ calibration commands: short and long.

The ZQ long calibration is executed during power up sequence, when existing self-refresh mode or when exiting slow precharge power down (DLL lock can be done in parallel). The ZQ short calibration is executed periodically according to a configurable timer defined by MPZQHWCTRL[ZQ_HW_PER].

The field MPZQHWCTRL[ZQ_MODE] determines whether the MMDC will execute ZQ calibration to DDR I/O pads and/or issue ZQ short/long command to the DDR device.

The MMDC supports both automatic (hardware) and manual (software) ZQ calibration process for the DDR I/O pads.

It is possible to perform automatic (hardware) ZQ calibration only once (i.e. non-periodical) by asserting MPZQHWCTRL[ZQ_HW_FOR].

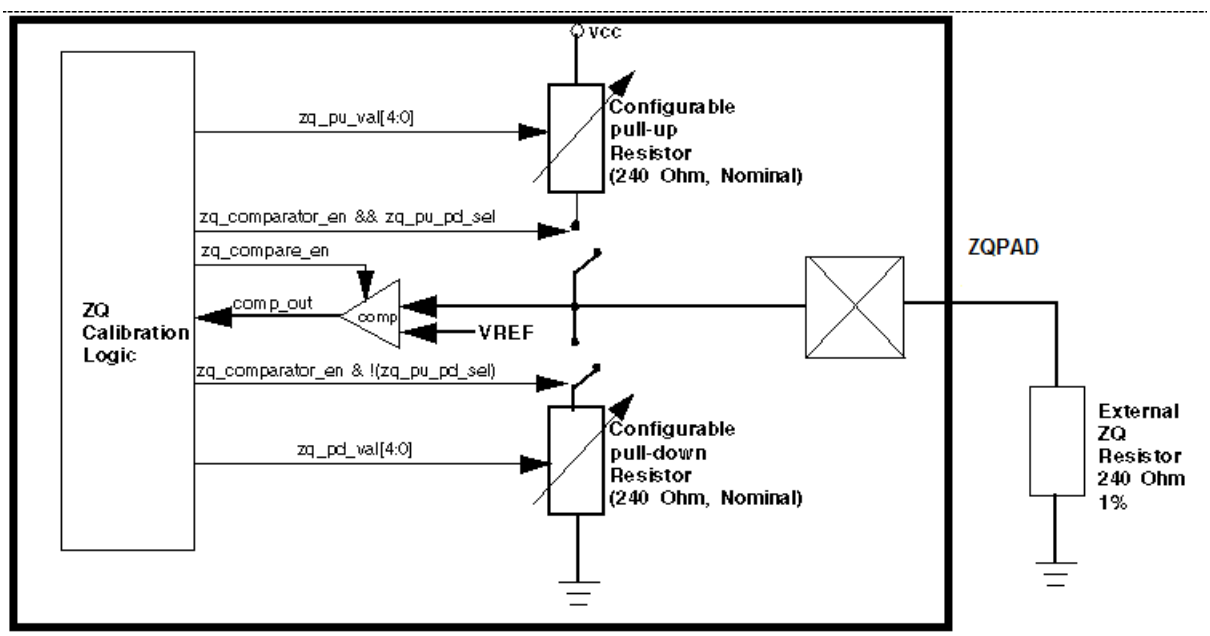


Figure 33-8. MMDC ZQ IF with PAD

33.11.2.1 ZQ automatic (hardware) calibration process

The ZQ automatic calibration lasts 11 steps. 5 steps for the pull up resistors calibration and 6 for the pull down resistors calibration.

The calibration control interface with ZQ pin is described below:

The calibration process is as follows:

33.11.2.1.1 ZQ automatic Pull-up calibration

The MMDC automatically performs a handshaking mechanism with the ZQ calibration pad as follows:

1. The MMDC drives zq_comparator_en to "1"
2. The MMDC waits few cycles according to MPZQHWCTRL[EARLY_COMPARATOR_EN_TIMER]
3. The MMDC drives zq_pu_pd_sel to "1" for indication of pull-up calibration and drives zq_pu_val[4:0] = 5'b00000
4. MMDC drives zq_pu_val[4] to "1"
5. MMDC asserts zq_compare_en
6. MMDC waits few cycles according to MPZQSWCTRL[ZQ_CMP_OUT_SMP] before sampling the comparator output (i.e zq_comp_out). If zq_comp_out is "1" then it means that the output voltage is greater than Vdd/2 (i.e. internal resistor is less than 240 ohm) and drives bit zq_pu_val[4] to "1" else it drives zq_pu_val[4] to "0"
7. MMDC deasserts zq_compare_en
8. MMDC repeats steps 4- 7for zq_pu_val bits 3 to 0
9. MMDC drives ZQ calibration result to MPZQHWCTRL[ZQ_HW_PU_RES]
10. MMDC advances to pull-down calibration

33.11.2.1.2 ZQ automatic Pull-down calibration

1. The MMDC drives zq_pu_pd_sel to "0" for indication of pull-down calibration and drives zq_pd_val[4:0] = 5'b00000
2. MMDC drives zq_pd_val[4] to "1"
3. MMDC asserts zq_compare_en
4. MMDC waits few cycles according to MPZQSWCTRL[ZQ_CMP_OUT_SMP] before sampling the comparator output (i.e zq_comp_out). If zq_comp_out is "1" then it means that the output voltage is greater than Vdd/2 (i.e. internal resistor is less than 240 ohm) and drives bit zq_pd_val[4] to "0" else it drives zq_pd_val[4] to "1"
5. MMDC deasserts zq_compare_en
6. MMDC repeats steps 12- 15 for zq_pd_val bits 3 to 0
7. MMDC drives ZQ calibration result to MPZQHWCTRL[ZQ_HW_PD_RES]
8. MMDC deassert zq_comparator_en to indicate the completion of the ZQ calibration

33.11.2.2 ZQ software calibration process

The ZQ calibration can be done also in software. However since software ZQ calibration is much slower than hardware calibration it should be used mainly for debugging.

Software should configure the ZQ calibration parameters (Pull-up or Pull-down and their value) then assert the MPZQSWCTRL[ZQ_SW_FOR] bit. Then software should wait till ZQ_SW_FOR is de-asserted and use ZQ_SW_RES status bit in order to calculate the next ZQ calibration parameters.

33.11.2.3 ZQ calibration commands

Before the MMDC can issue a ZQCL/ZQCS command to the memory it should precharge all memory banks and wait tRP period. A single ZQ command can be issued to all devices as long as the devices don't share the same ZQ resistor.

When the MMDC issues the ZQ command it should also drive A10 (long or short command) and CS (0, 1 or both).

The MMDC must keep the memory lines quiet (except for CK) for the ZQ calibration time as defined in the Jedec (512 cycles for ZQCL after reset, 256 for other ZQCL and 64 for ZQCS).

33.11.3 Read DQS Gating Calibration

The read DQS gating calibration is used to adjust the read DQS gating with the middle of the read DQS preamble. The DQS gating includes a delay of up to 7 cycles (The delay is chosen according to two fields MPDGCTRL#[DG_HC_DEL#] and MPDGCTRL#[DG_DL_ABS_OFFSET#]

Each DQS has its own delay-line. The DQS gating process can be done for all DQS in parallel.

NOTE

In LPDDR2/LPDDR3 mode hardware Read DQS gating should be disabled and Pull-up/pull-down resistors on DQS/DQS# should be enabled while ODT resistors must be disconnected.

33.11.3.1 Hardware DQS Gating Calibration

- There are two modes of operations:
  - Calibration with the MPR (Multi Purpose Register)
  - Calibration with MMDC pre-defined values

33.11.3.1.1 Hardware DQS Calibration with MPR

The following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as requried by the standard.
2. Enter the DDR device into MPR mode through MRS commands
3. Configure the MMDC to work with MPR mode by asserting MPPDCMPR2[MPR_CMP]
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window
5. Start the calibration process by asserting MPDGCTRL0[HW_DG_EN]

33.11.3.1.2 Hardware DQS Calibration with pre-defined value

In case pre-defined mode is used, (i.e. MPPDCMPR2[MPR_CMP]) is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calibration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1 (MMDC will generate internally write access without intervention of the system towards bank 0, row 0, column 0)
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window
5. Start the calibration process by asserting MPDGCTRL0[HW_DG_EN]

   The following steps will be executed automatically by the MMDC for both modes (MPR and Pre-defined value):

6. MMDC waits till the read DQS delay-line is updated with the absolute delay value for all bytes at MPDGCTRL#[DG_HC_DEL#] and MPDGCTRL#[DG_DL_ABS_OFFSET#] and also satisfying the Tmod + 4 requirement

7. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
8. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point. If the comparison passes then MMDC advances to step 14
9. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
10. MMDC increments the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1)
11. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
12. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8).
13. If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 9-12. If the comparison passes then MMDC stores the value of the temporary low boundary and advances to next step
14. MMDC increments the read DQS gating delay-line of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1) and issue measurement process of the read DQS gating delay-line to update itself with the new value
15. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
16. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
17. If the comparison passes then it indicates that the read DQS gating is asserted inside the read preamble window and it is needed to repeat steps 14-16. If the comparison fails then MMDC stores the value of the temporary upper boundary and starts searching the adequate low and high boundaries
18. MMDC returns to the temporary low boundary minus half cycle and issue measurement process of the read DQS gating delay-line to update itself with the new value
19. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
20. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)

21. If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 22-23. If the comparison passes then MMDC stores the value of the adequate low boundary and advances to step 24
22. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
23. MMDC increments the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and issue measurement process of the read DQS gating delay-line to update itself with the new value and advances to step 19
24. MMDC returns to the temporary upper boundary minus half cycle and issue measurement process of the read DQS gating delay-line to update itself with the new value
25. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
26. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
27. If the comparison passes then it indicates that the read DQS gating is asserted inside the read preamble window and it is needed to repeat steps 28-29. If the comparison fails then MMDC stores the value minus 1 of the adequate upper boundary and advances to step 30
28. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
29. MMDC increments the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and issue measurement process of the read DQS gating delay-line to update itself with the new value and advances to step 25
30. After the MMDC finds the window boundary (lower and upper) of each read data byte then it stores the average between lower and upper boundaries at the associated MPDGCTRL#[DG_DL_ABS_OFFSET#] and issue measurement process of the read DQS delay-line to update itself with the new value.
31. MMDC indicates that the read DQS gating calibration had finished by setting MPDGCTRL0[HW_DG_EN] = 0
32. Exit the DDR device from MPR mode through MRS command
33. Read the upper boundary that was found: MPDGHWST#[HW_DG_UP#]. This field is 11 bits, 7 LSB bits correspond to MPDGCTRL#[DG_DL_ABS_OFFSET#] upper limit value and 4 MSB bits correspond to MPDGCTRL#[DG_HC_DEL#] upper limit value.
34. Set MPDGHWST#[HW_DG_UP#][6:0] to MPDGCTRL#[DG_DL_ABS_OFFSET#].

35. Set (MPDGHWST#[HW_DG_UP#][10:7] - 1) to MPDGCTRL#[DG_HC_DEL#]. (We set the DQS gating value to be the upper limit value minus 1 half cycle)

33.11.3.2 SW read DQS gating Calibration

- There are two modes of operations:
- Calibration with the MPR (Multi Purpose Register)
- Calibration with MMDC pre-defined values

33.11.3.2.1 SW read Calibration with MPR

The following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Enter the DDR device into MPR mode through MRS commands
3. Configure the MMDC to work with MPR mode by asserting MPPDCMPR2[MPR_CMP]
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

33.11.3.2.2 SW read Calibration with pre-defined value

In case pre-defined mode is used, (i.e. MPPDCMPR2[MPR_CMP]) is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calicration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access (with any legal DDR address) to external DDR device
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

   The following steps should be executed automatically by the MMDC for both modes (MPR and Pre-defined value):

5. Configure the read DQS delay-line to issue zero delay by setting MPDGCTRL#[DG_DL_ABS_OFFSET#] = 0 and MPDGCTRL#[DG_HC_DEL#] = 0

6. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
7. Wait 16 DDR cycles till the read DQS delay-line is updated with the absolute delay value for all bytes
8. Issue read command (with the legal DDR address chosen in step 3) from the external DDR device
9. Waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC]) assuming that the data has arrived from the DDR device.
10. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point. If the comparison passes then advance to step 15
11. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
12. Increment the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1)
13. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC]) assuming that the data has arrived from the DDR device.
14. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 11 - 14. If the comparison passes then advance to step 15
15. Store the temporary lower boundary and start searching the temporary upper boundary
16. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
17. Increment the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1)
18. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
19. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8).
20. If the comparison passes then it indicates that the read DQS gating is asserted inside the read preamble window and it is needed to repeat steps 16-19. If the comparison fails then it is needed to store the value of the temporary upper boundary and starts searching the adequate low and high boundaries
21. Load the temporary low boundary minus half cycle into the associated MPDGCTRL#[DG_HC_DEL#]

22. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
23. Increment the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and force the delay line to measure itself and to issue the requested read DQS delay by configuring MPMUR[FRC_MSR] = 1
24. Issue read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC]) assuming that the data has arrived from the DDR device.
25. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
26. If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 22-26. If the comparisons passes then advance to the next step.
27. Store the adequate lower boundary
28. Load the temporary upper boundary minus half cycle into the associated MPDGCTRL#[DG_HC_DEL#]
29. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
30. Increment the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and force the delay line to measure itself and to issue the requested read DQS delay by configuring MPMUR[FRC_MSR] = 1
31. Issue read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
32. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
33. If the comparison passes then it is needed to repeat steps 29-32. If the comparisons fails then advance to the next step.
34. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
35. Store the adequate upper boundary.
36. Keep the MPDGCTRL#[DG_DL_ABS_OFFSET#] value of the upper limit.
37. Set MPDGCTRL#[DG_HC_DEL#] = (MPDGCTRL#[DG_HC_DEL#] - 1). (We set the DQS gating value to be the upper limit value minus 1 half cycle)
38. Issue the requested read DQS delay by configuring MPMUR[FRC_MSR] = 1
39. Exit the DDR device from MPR mode through MRS command

33.11.4 Read Calibration

The read calibration is used to adjust the read DQS with read data byte. It is assumed that the read DQS gating calibration process is completed prior to the read calibration.

NOTE

In DDR3 mode, the activation of the calibration is done by setting MPRDDLHWCTL[HW_RD_DL_EN].

NOTE

In LP2_x16 the activation of the calibration is done by setting MPRDDLHWCTL[HW_RD_DL_EN].

NOTE

In LP2/3_x16 the activation of the calibration is done by setting MPRDDLHWCTL[HW_RD_DL_EN].

33.11.4.1 Hardware (automatic) Read Calibration

- There are two modes of operations:
- Calibration with the MPR (Multi Purpose Register)/DQ calibration(LPDDR2)
- Calibration with MMDC pre-defined values

33.11.4.1.1 Hardware (automatic) Calibration with MPR (DDR3) /DQ Calibration (LPDDR2)

The following steps should be executed:

1. Precharge all active banks (can be done through MDSCR) as requried by the standard.
2. Enter the DDR device into MPR/DQ calibration mode through MRS/MRW commands.
3. Configure the MMDC to work with MPR/DQ calibration mode by asserting MPPDCMPR2[MPR_CMP].
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window.
5. Start the calibration process by asserting MPRDDLHWCTL[HW_RD_DL_EN].

33.11.4.1.2 Hardware (automatic) Calibration with pre-defined value

In case pre-defined mode is used, i.e. MPPDCMPR2[MPR_CMP] is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calibration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1 (MMDC will generate internally write access without intervention of the system towards bank 0, row 0, column 0)
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window
5. Start the calibration process by asserting MPRDDLHWCTL[HW_RD_DL_EN]

   The following steps will be executed automatically by the MMDC for both modes (MPR and Pre-defined value):

6. MMDC waits till the read delay-line is updated with the absolute delay value for all bytes at MPRDDLCTL[RD_DL_ABS_OFFSET#] and also satisfying the Tmod + 4 requirement
7. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device.
8. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the initial read DQS isn't inside the read DQ window and the MMDC generates an error for the associated byte at MPRDDLHWCTL[HW_RD_DL_ERR#] . If the comparison passes then MMDC advances to next step.
9. MMDC resets the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
10. MMDC decrements the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) and issue measurement process of the read delay-line to update itself with the new value.
11. MMDC drives read command to the DDR external devices and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device
12. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the low read boundary of the associated byte for each byte at MPRDDLHWST0/1[HW_RD_DL_LOW#] . If the comparison passes then MMDC repeats steps 9-11. If all read data comparisons fail then the MMDC advances to the next step

13. The MMDC start seeking the upper boundary and sets the read delay line absolute offset of each byte to the initial value + 1 as determined at step 4 and issue measurement process of the read delay-line to update itself with the new value
14. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
15. MMDC drives read command to the DDR external devices and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device
16. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the upper read boundary of the associated byte for each byte at MPRDDLHWST0/1[HW_RD_DL_UP#] . If the comparison passes then MMDC increments the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) and issue measurement process of the read delay-line to update itself with the new value.
17. If all read data comparisons fail then the MMDC advances to the next step. otherwise, MMDC repeats steps 14-16.
18. After the MMDC finds the window boundary (lower and upper) of each read data byte then it stores the average between lower and upper boundaries at the associated MPRDDLCTL[RD_DL_ABS_OFFSET#] and issue measurement process of the read delay-line to update itself with the new value.
19. MMDC indicates that the read data calibration had finished by setting MPRDDLHWCTL[HW_RD_DL_EN] = 0
20. Exit the DDR device from MPR/DQ calibration mode through MRS/MRW commands

33.11.4.2 SW Read Calibration

- There are two modes of operations:
- Calibration with the MPR (Multi Purpose Register)/DQ calibration (LPDDR2)
- Calibration with MMDC pre-defined values

33.11.4.2.1 Calibration with MPR(DDR3)/DQ calibration(LPDDR2)

The following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Enter the DDR device into MPR/DQ calibration mode through MRS/MRW commands

3. Configure the MMDC to work with MPR/DQ calibration mode by asserting MPPDCMPR2[MPR_CMP]
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

33.11.4.2.2 Calibration with pre-defined value

In case pre-defined mode is used, i.e. MPPDCMPR2[MPR_CMP] is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as requried by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calibration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access (with any legal DDR address) to external DDR device.
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

   The following steps will be executed manually by SW for both modes (MPR/DQ calibration and Pre-defined value):

5. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
6. Wait 16 DDR cycles till the read delay-line is updated with the absolute delay value for all bytes
7. Issue read command (with any legal DDR address) from the external DDR device
8. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the initial read DQS isn't inside the read DQ window. If the comparison passes then advance to next step.
9. Reset the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
10. Decrement the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#])
11. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
12. Issue read command (with the legal DDR address chosen in step 7) from the external DDR device and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device

13. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed to store the low read boundary of the associated byte at of each byte . If the comparison passes then repeat steps 9-12. If all read data comparisons fail then advance to the next step.
14. Start seeking the upper boundary and set the read delay line absolute offset of each byte to the initial value + 1 as determined at step 4
15. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
16. Resets the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
17. Issue read command (with the legal DDR address chosen in step 7) from the external DDR device
18. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed to store the upper read boundary of the associated byte at of each byte. If the comparison passes then increment the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#])
19. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
20. If all read data comparisons fail then advance to the next step, else repeat steps 16-19
21. After finding the window boundary (lower and upper) of each read data byte then calculate the average between lower and upper boundaries and store the associated average at MPRDDLCTL[RD_DL_ABS_OFFSET#]
22. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
23. Exit the DDR device from MPR/DQ calibration mode through MRS/MRW commands.

33.11.5 Write Calibration

The write calibration is used to adjust the write DQS with write data byte. It is assumed that the read calibration process is completed prior to the write calibration.

NOTE

In DDR3_x64 and LP2_1ch_x64 modes, the activation of the calibration is done by setting MPWRDLHWCTL0[HW_WR_DL_EN]

NOTE

In LP2/3_2ch_x16, LP2/3_2ch_x32 the activation of the calibration of each channel is done by setting MPWRDLHWCTL0[HW_WR_DL_EN] and MPWRDLHWCTL0[HW_WR_DL_EN]

NOTE

In LP2_2ch_x16, LP2_2ch_x32 the activation of the calibration of each channel is done by setting MPWRDLHWCTL0[HW_WR_DL_EN] and MPWRDLHWCTL0[HW_WR_DL_EN]

33.11.5.1 HW (automatic) Write Calibration

The following steps should be executed:

1. Make sure that the initial value that is configured in the write delay line absolute offset of each byte (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) will place the write DQS somewhere inside the write DQ window
2. Configure the pre-defined value, which reflects the value that will be written and compared through the write calibration, to MPPDCMPR1[PDV1, PDV2]
3. Assert MPWRDLHWCTL0[HW_WR_DL_EN]

   The following steps will be executed automatically:

4. MMDC waits till the write delay-line is updated with the absolute delay value for all bytes at MPWRDCTL[WR_DL_ABS_OFFSET#]
5. MMDC drives write command to the external DDR devices (to bank 0 address 0) and waits 16 or 32 cycles (according to MPWRDLHWCTL[HW_WR_DL_CMP_CYC]) assuming that the data has arrived to the DDR device.
6. MMDC drives read command to the same address from the external DDR
7. MMDC compares the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the initial write DQS isn't inside the write DQ window and the MMDC generates an error for the associated byte at MPWRDLHWCTL[HW_WR_DL_ERR#] . If the comparison passes then MMDC advances to next step.
8. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
9. MMDC decrements the write delay line absolute offset of each byte by 1 (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) and issue measurement process of the write delay-line to update itself with the new value.

10. MMDC drives write command to the external DDR devices (to bank 0 address 0) and waits 16 or 32 cycles (according to MPWRDLHWCTL[HW_WR_DL_CMP_CYC]) assuming that the data has arrived to the DDR device
11. MMDC drives read command to the same address from the external DDR
12. MMDC compares the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the low write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_LOW#] . If the comparison passes then MMDC repeats steps 8-11. If all data comparisons fail then the MMDC advances to the next step
13. The MMDC start seeking the upper boundary and sets the write delay line absolute offset of each byte to the initial value + 1 as determined at step 4 and issue measurement process of the write delay-line to update itself with the new value
14. MMDC resets the rd fifo (to the inverted pre-defined value) and its pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
15. MMDC drives write command to the external DDR devices (to bank 0 address 0) and waits 16 or 32 cycles (according to MPWRDLHWCTL[HW_WR_DL_CMP_CYC]) assuming that the data has arrived to the DDR device.
16. MMDC drives read command to the same address from the external DDR
17. MMDC compares the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the upper write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_UP#] . If the comparison passes then MMDC increments the write delay line absolute offset of each byte by 1 (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) and issue measurement process of the write delay-line to update itself with the new value.
18. MMDC repeats steps 14-17. If all data comparisons fail then the MMDC advances to the next step
19. .After the MMDC finds the window boundary (lower and upper) of each write data byte then it stores the average between lower and upper boundaries at the associated MPWRDLCTL[WR_DL_ABS_OFFSET#] and issue measurement process of the write delay-line to update itself with the new value.
20. MMDC indicates that the write data calibration had finished by setting MPWRDLHWCTL[HW_WR_DL_EN] = 0

33.11.5.2 SW Write Calibration

The following steps should be executed:

NOTE

It is recommended to perform the write calibration using the HW method. The SW method is provided for debug purposes only.

1. Make sure that the initial value that is configured in the write delay line absolute offset of each byte (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) will place the write DQS somewhere inside the write DQ window
2. Configure the pre-defined value, which reflects the value that will be written and compared through the write calibration, to MPPDCMPR1[PDV1, PDV2]
3. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
4. Wait 16 DDR cycles till the write delay-line is updated with the absolute delay value for all bytes
5. Issue write command to any legal DDR address of the external DDR device
6. Issue read command, to the address written previously, from the external DDR device
7. Compare the read data byte to the associated byte in the pre-defined value for all bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the initial write DQS isn't inside the write DQ window. If the comparison passes then advance to next step.
8. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
9. Decrement the write delay line absolute offset of each byte by 1 (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#])
10. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
11. Issue write command to any legal DDR address of the external DDR device
12. Issue read command, to the address written previously, from the external DDR device
13. Compare the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed to store the low write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_LOW#] . If the comparison passes then repeat steps 8-12. If all data comparisons fail then advance to the next step.
14. Start seeking the upper boundary and set the write delay line absolute offset of each byte to the initial value + 1
15. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
16. Reset the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
17. Issue write command to any legal DDR address of the external DDR device

18. Issue read command, to the address written previously, from the external DDR device
19. Compare the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed to store the upper write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_UP#]. If the comparison passes then increment the write delay line absolute offset of each byte by 1.
20. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
21. If all read data comparisons fail then advance to the next step else repeat steps 16-20.
22. .After finding the window boundary (lower and upper) of each write data byte then calculate the average between lower and upper boundaries and store the associated average at MPWRDLCTL[WR_DL_ABS_OFFSET#]
23. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1

33.11.6 Write leveling Calibration

The write leveling calibration can generate a delay between the clock and the associate DQS of up to 3 cycles as following: (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). Write leveling calibration can be executed automatically(HW) or manually (SW).

The automatic calibration process can only detect the optimal DQS to clock delay to within 1 cycle. In extreme cases in which the DDR3 memory is placed far from the microcontroller (long address/command/clock trace lengths), the skew between the DQS and clock may exceed 1 cycle. If this is the case, it is the user's responsibility to both understand that their design causes the DQS to clock skew to exceed 1 cycle and to indicate this manually in the MPWLDECTRL0/1[WL_CYC_DEL#]. It is highly recommended to keep the DDR3 memory as close to the microcontroller as possible, especially in embedded system designs. When using fly-by topology, the user should calculate the PCB flight time of the clock signal to the furthest placed DDR3 memory to ensure less than 1 cycle skew between DQS and clock.

NOTE

In LPDDR2 mode Write-leveling calibration should be disabled.

33.11.6.1 Hardware Write Leveling Calibration

The following steps should be executed:

1. Configure the external DDR device to enter write leveling mode through MRS command
2. Activate the DQS output enable by setting MDSCR[WL_EN]
3. Active automatic calibration by setting MPWLGCR[HW_WL_EN]

   The following steps will be executed automatically by the MMDC:

4. MMDC enters write leveling mode, counts 25 + 15 cycles and drives the DQS pads as output while the DQ pads will remain inputs. In parallel the MMDC configures the write leveling delay line to "0" (i.e. MPWLDECTRL0[WL_DL_ABS_OFFSET#] = 0) and issue measurement process of the writ-leveling delay-line to update itself with the new value
5. MMDC drives one DQS pulse to the DDR external device
6. MMDC waits 16 cycles (to guarantee that the DQ prime data is stable) and samples the associated prime DQ bit (for example for DQS1 the MMDC samples DQ[8])
7. MMDC increments the write leveling delay line by 1/8 cycle and perform measurement process in order to load the updated value to the associated delay-line
8. MMDC repeates steps 5-7 till the write leveling delay is 1 cycle
9. MMDC checks the 8 bit prime DQ results for each DQS and finds the first transition from 0 to 1. If no transition is found then the MMDC indicates an error at MPWLGCR[HW_WL_ERR#]
10. MMDC stores the value that issues the last "0" on the prime DQ before the transition and loads it to the write leveling delay-line. The MMDC initiates a fine-tune process by incrementing the delay-line values by 1 step (which is 1/256 part of a cycle) till detecting the most accurate transition from 0 to 1
11. Upon completion of this process the MMDC de-asserts the MPWLGCR[HW_WL_EN] and update the most accurate value of the delay-line at the associated MPWLDECTRL#[WL_DL_ABS_OFFSET#]
12. MMDC perform measurement process in order to load the most accurate value to the associated delay-line
13. User should issue MRS command to exit write leveling mode
14. The user should read the results of the associated delay-line at MPWLDECTRL#[WL_DL_ABS_OFFSET#] and in case the user estimates that the reasonable delay may be above 1 cycle then the user should indicate it at MPWLDECTRL#[WL_CYC_DEL#]. Moreover the user should indicate it in MDMISC[WALAT] field. For example, if the result of the write leveling calibration is 100/256 parts of a cycle, but the user estimates that the delay is above 2 cycles then MPWLDECTRL#[WL_CYC_DEL#] should be configured to 2, so the total delay will be 2 and 100/256 parts of a cycle

15. Return the DQS output enable to functional mode by deasserting MDSCR[WL_EN]

33.11.6.2 SW Write Leveling Calibration

The following steps should be executed:

NOTE

It is recommended to perform the write calibration using the HW method. The SW method is provided for debug purposes only.

1. Configure the external DDR device to enter write leveling mode through MRS command
2. Activate the DQS output enable by setting MDSCR[WL_EN]
3. Set the write-leveling delay-line offset to "0" by configuring MPWLDECTRL0[WL_DL_ABS_OFFSET#] = 0
4. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
5. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1 together with MPWLGCR[SW_WL_CNT_EN] = 1
6. Issue an IP read command from MPWLGCR. If MPWLGCR[SW_WL_EN] = 0 then the SW write-leveling result is valid at MPWLGCR[WL_SW_RES#].
7. Increment the write leveling delay line by 1/8 cycle (i.e add 0x20 to {MPWLDECTRL0[WL_HC_DEL#],MPWLDECTRL0[WL_DL_ABS_OFFSET#]})
8. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
9. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1
10. Repeate steps 6-9 till the edge of CK was detected (i.e the write-leveling result switched from "0" to "1")
11. Store the value that issues the last "0" on the prime DQ before the transition and load it to the write leveling delay-line and start fine tuning process to detect the exact switch from "0" to "1"
12. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
13. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1
14. Issue a IP read command from MPWLGCR. If MPWLGCR[SW_WL_EN] = 0 then the SW write-leveling result is valid at MPWLGCR[WL_SW_RES#].

15. Increment the write leveling delay line by 1 step (i.e add 0x01 to MPWLDECTRL0[WL_DL_ABS_OFFSET#])
16. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
17. Issue an IP read command from MPWLGCR. If MPWLGCR[SW_WL_EN] = 0 then the SW write-leveling result is valid at MPWLGCR[WL_SW_RES#].
18. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1
19. Repeates step 15-18 till the exact edge of CK was detected (i.e the write-leveling result switched from "0" to "1")
20. Issue MRS command to exit write leveling mode
21. Return the DQS output enable to functional mode by deasserting MDSCR[WL_EN]

33.11.7 Write fine tuning

Write fine tuning is an additional circuit that provides the ability to fine tune the timing of each of the DQ/DM bits (relative to DQS) by up to 100 ps. To use the write fine tuning, select the number of delay units in each DQ/DM I/O (maximum 3 delay units of around 30-35 ps each). The delay can be configured independently for each DQ/DM. This configuration is controlled by the MPWRDQBY#DL register.

33.11.8 Read fine tuning

Read fine tuning is an additional circuit that provides the ability to fine tune the timing of each coming dq bits (relative to coming dqs) by up to +/-100 ps.

This is done by reducing the delay between the incoming rd_dqs by 100 ps and adding a configurable delay of up to 200 ps (6 delay units of around 30-35 ps each) for each DQ input. The delay can be configured independently for each DQ. The calibration of this mechanism can be done only by writing & reading data from the memory. Controlled by register MPRDDQBY#DL.

33.11.9 ZQ Fine Tuning

An offset can be added to the PU /PD values determined by the ZQ calibration process. The offset range is programmable from -7 to +7, controlled by the MMDC_MPPDCMPR2[ZQ_PU_OFFSET] and

MMDC_MPPDCMPR2[ZQ_PD_OFFSET] fields. The offset is enabled/disabled by MMDC_MPPDCMPR2[ZQ_OFFSET_EN]. See MMDC_MPPDCMPR2 register for more information.

33.11.10 Duty cycle adjustment

Duty cycle adjustments can be made to the SDCLKx and SDQSx signals, see the MMDCx_MPDCCR register for more information.

33.12 MMDC Memory Map/Register Definition

MMDC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_0000	MMDC Core Control Register (MMDC_MDCTL)	32	R/W	0311_0000h	33.12.1/ 2067
21B_0004	MMDC Core Power Down Control Register (MMDC_MDPDC)	32	R/W	0003_0012h	33.12.2/ 2069
21B_0008	MMDC Core ODT Timing Control Register (MMDC_MDOTC)	32	R/W	1227_2000h	33.12.3/ 2071
21B_000C	MMDC Core Timing Configuration Register 0 (MMDC_MDCFG0)	32	R/W	3236_22D3h	33.12.4/ 2073
21B_0010	MMDC Core Timing Configuration Register 1 (MMDC_MDCFG1)	32	R/W	B6B1_8A23h	33.12.5/ 2075
21B_0014	MMDC Core Timing Configuration Register 2 (MMDC_MDCFG2)	32	R/W	00C7_0092h	33.12.6/ 2077
21B_0018	MMDC Core Miscellaneous Register (MMDC_MDMISC)	32	R/W	0000_1600h	33.12.7/ 2079
21B_001C	MMDC Core Special Command Register (MMDC_MDSCR)	32	R/W	0000_0000h	33.12.8/ 2082
21B_0020	MMDC Core Refresh Control Register (MMDC_MDREF)	32	R/W	0000_C000h	33.12.9/ 2085
21B_002C	MMDC Core Read/Write Command Delay Register (MMDC_MDRWD)	32	R/W	0F9F_26D2h	33.12.10/ 2087
21B_0030	MMDC Core Out of Reset Delays Register (MMDC_MDOR)	32	R/W	009F_0E0Eh	33.12.11/ 2089
21B_0034	MMDC Core MRR Data Register (MMDC_MDMRR)	32	R	0000_0000h	33.12.12/ 2090
21B_0038	MMDC Core Timing Configuration Register 3 (MMDC_MDCFG3LP)	32	R/W	0000_0000h	33.12.13/ 2091
21B_003C	MMDC Core MR4 Derating Register (MMDC_MDMR4)	32	R/W	0000_0000h	33.12.14/ 2093

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_0040	MMDC Core Address Space Partition Register (MMDC_MDASP)	32	R/W	0000_003Fh	33.12.15/ 2095
21B_0400	MMDC Core AXI Reordering Control Register (MMDC_MAARCR)	32	R/W	5142_01F0h	33.12.16/ 2096
21B_0404	MMDC Core Power Saving Control and Status Register (MMDC_MAPSR)	32	R/W	0000_1007h	33.12.17/ 2099
21B_0408	MMDC Core Exclusive ID Monitor Register0 (MMDC_MAEXIDR0)	32	R/W	0020_0000h	33.12.18/ 2101
21B_040C	MMDC Core Exclusive ID Monitor Register1 (MMDC_MAEXIDR1)	32	R/W	0060_0040h	33.12.19/ 2102
21B_0410	MMDC Core Debug and Profiling Control Register 0 (MMDC_MADPCR0)	32	R/W	0000_0000h	33.12.20/ 2103
21B_0414	MMDC Core Debug and Profiling Control Register 1 (MMDC_MADPCR1)	32	R/W	0000_0000h	33.12.21/ 2104
21B_0418	MMDC Core Debug and Profiling Status Register 0 (MMDC_MADPSR0)	32	R	0000_0000h	33.12.22/ 2105
21B_041C	MMDC Core Debug and Profiling Status Register 1 (MMDC_MADPSR1)	32	R	0000_0000h	33.12.23/ 2105
21B_0420	MMDC Core Debug and Profiling Status Register 2 (MMDC_MADPSR2)	32	R	0000_0000h	33.12.24/ 2106
21B_0424	MMDC Core Debug and Profiling Status Register 3 (MMDC_MADPSR3)	32	R	0000_0000h	33.12.25/ 2106
21B_0428	MMDC Core Debug and Profiling Status Register 4 (MMDC_MADPSR4)	32	R	0000_0000h	33.12.26/ 2107
21B_042C	MMDC Core Debug and Profiling Status Register 5 (MMDC_MADPSR5)	32	R	0000_0000h	33.12.27/ 2107
21B_0430	MMDC Core Step By Step Address Register (MMDC_MASBS0)	32	R	0000_0000h	33.12.28/ 2108
21B_0434	MMDC Core Step By Step Address Attributes Register (MMDC_MASBS1)	32	R	0000_0000h	33.12.29/ 2108
21B_0440	MMDC Core General Purpose Register (MMDC_MAGENP)	32	R/W	0000_0000h	33.12.30/ 2109
21B_0800	MMDC PHY ZQ HW control register (MMDC_MPZQHWCTRL)	32	R/W	A138_0000h	33.12.31/ 2110
21B_0804	MMDC PHY ZQ SW control register (MMDC_MPZQSWCTRL)	32	R/W	0000_0000h	33.12.32/ 2113
21B_0808	MMDC PHY Write Leveling Configuration and Error Status Register (MMDC_MPWLGCR)	32	R/W	0000_0000h	33.12.33/ 2115
21B_080C	MMDC PHY Write Leveling Delay Control Register 0 (MMDC_MPWLDECTRL0)	32	R/W	0000_0000h	33.12.34/ 2118
21B_0814	MMDC PHY Write Leveling delay-line Status Register (MMDC_MPWLDLST)	32	R	0000_0000h	33.12.35/ 2121
21B_0818	MMDC PHY ODT control register (MMDC_MPODTCTRL)	32	R/W	0000_0000h	33.12.36/ 2122

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_081C	MMDC PHY Read DQ Byte0 Delay Register (MMDC_MPRDDQBY0DL)	32	R/W	0000_0000h	33.12.37/ 2124
21B_0820	MMDC PHY Read DQ Byte1 Delay Register (MMDC_MPRDDQBY1DL)	32	R/W	0000_0000h	33.12.38/ 2127
21B_082C	MMDC PHY Write DQ Byte0 Delay Register (MMDC_MPWRDQBY0DL)	32	R/W	0000_0000h	33.12.39/ 2130
21B_0830	MMDC PHY Write DQ Byte1 Delay Register (MMDC_MPWRDQBY1DL)	32	R/W	0000_0000h	33.12.40/ 2132
21B_083C	MMDC PHY Read DQS Gating Control Register 0 (MMDC_MPDGCTRL0)	32	R/W	0000_0000h	33.12.41/ 2134
21B_0844	MMDC PHY Read DQS Gating delay-line Status Register (MMDC_MPDGDLST0)	32	R	0000_0000h	33.12.42/ 2137
21B_0848	MMDC PHY Read delay-lines Configuration Register (MMDC_MPRDDLCTL)	32	R/W	4040_4040h	33.12.43/ 2139
21B_084C	MMDC PHY Read delay-lines Status Register (MMDC_MPRDDLST)	32	R	0000_0000h	33.12.44/ 2141
21B_0850	MMDC PHY Write delay-lines Configuration Register (MMDC_MPWRDLCTL)	32	R/W	4040_4040h	33.12.45/ 2143
21B_0854	MMDC PHY Write delay-lines Status Register (MMDC_MPWRDLST)	32	R	0000_0000h	33.12.46/ 2145
21B_0858	MMDC PHY CK Control Register (MMDC_MPSDCTRL)	32	R/W	0000_0000h	33.12.47/ 2146
21B_085C	MMDC ZQ LPDDR2 HW Control Register (MMDC_MPZQLP2CTL)	32	R/W	1B5F_0109h	33.12.48/ 2147
21B_0860	MMDC PHY Read Delay HW Calibration Control Register (MMDC_MPRDDLHWCTL)	32	R/W	0000_0000h	33.12.49/ 2149
21B_0864	MMDC PHY Write Delay HW Calibration Control Register (MMDC_MPWRDLHWCTL)	32	R/W	0000_0000h	33.12.50/ 2152
21B_0868	MMDC PHY Read Delay HW Calibration Status Register 0 (MMDC_MPRDDLHWST0)	32	R	0000_0000h	33.12.51/ 2154
21B_0870	MMDC PHY Write Delay HW Calibration Status Register 0 (MMDC_MPWRDLHWST0)	32	R	0000_0000h	33.12.52/ 2155
21B_0878	MMDC PHY Write Leveling HW Error Register (MMDC_MPWLHWERR)	32	R/W	0000_0000h	33.12.53/ 2156
21B_087C	MMDC PHY Read DQS Gating HW Status Register 0 (MMDC_MPDGHWST0)	32	R	0000_0000h	33.12.54/ 2156
21B_0880	MMDC PHY Read DQS Gating HW Status Register 1 (MMDC_MPDGHWST1)	32	R	0000_0000h	33.12.55/ 2157
21B_088C	MMDC PHY Pre-defined Compare Register 1 (MMDC_MPPDCMPR1)	32	R/W	0000_0000h	33.12.56/ 2157
21B_0890	MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MMDC_MPPDCMPR2)	32	R/W	0040_0000h	33.12.57/ 2159
21B_0894	MMDC PHY SW Dummy Access Register (MMDC_MPSWDAR0)	32	R/W	0000_0000h	33.12.58/ 2162

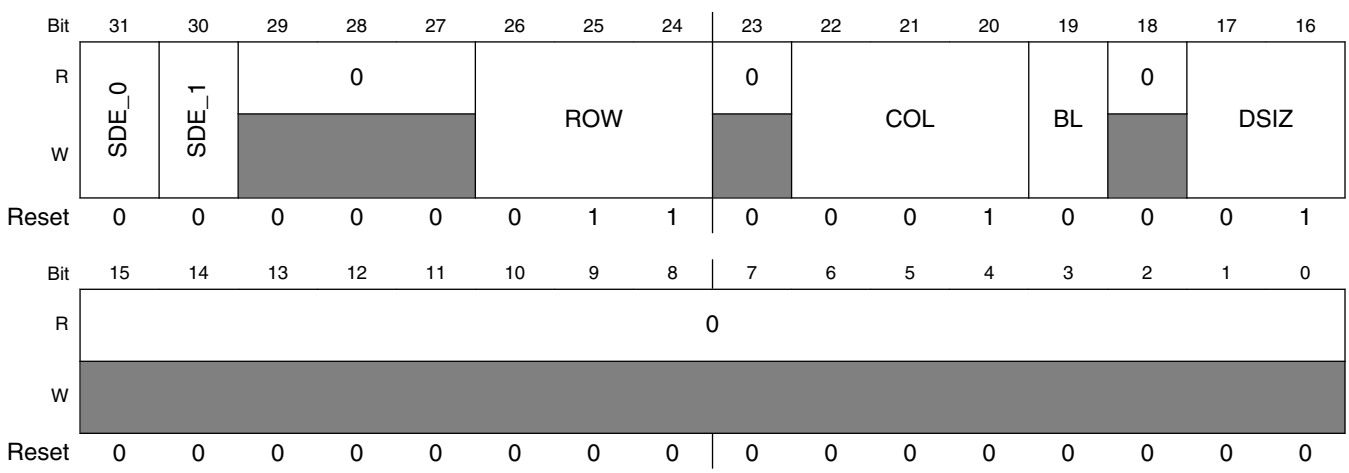
Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_0898	MMDC PHY SW Dummy Read Data Register 0 (MMDC_MPSWDRDR0)	32	R	FFFF_FFFFh	33.12.59/ 2164
21B_089C	MMDC PHY SW Dummy Read Data Register 1 (MMDC_MPSWDRDR1)	32	R	FFFF_FFFFh	33.12.60/ 2164
21B_08A0	MMDC PHY SW Dummy Read Data Register 2 (MMDC_MPSWDRDR2)	32	R	FFFF_FFFFh	33.12.61/ 2165
21B_08A4	MMDC PHY SW Dummy Read Data Register 3 (MMDC_MPSWDRDR3)	32	R	FFFF_FFFFh	33.12.62/ 2165
21B_08A8	MMDC PHY SW Dummy Read Data Register 4 (MMDC_MPSWDRDR4)	32	R	FFFF_FFFFh	33.12.63/ 2165
21B_08AC	MMDC PHY SW Dummy Read Data Register 5 (MMDC_MPSWDRDR5)	32	R	FFFF_FFFFh	33.12.64/ 2166
21B_08B0	MMDC PHY SW Dummy Read Data Register 6 (MMDC_MPSWDRDR6)	32	R	FFFF_FFFFh	33.12.65/ 2166
21B_08B4	MMDC PHY SW Dummy Read Data Register 7 (MMDC_MPSWDRDR7)	32	R	FFFF_FFFFh	33.12.66/ 2167
21B_08B8	MMDC PHY Measure Unit Register (MMDC_MPMUR0)	32	R/W	0000_0000h	33.12.67/ 2167
21B_08BC	MMDC Write CA delay-line controller (MMDC_MPWRCADL)	32	R/W	0000_0000h	33.12.68/ 2168
21B_08C0	MMDC Duty Cycle Control Register (MMDC_MPDCCR)	32	R/W	2492_2492h	33.12.69/ 2170

33.12.1 MMDC Core Control Register (MMDC_MDCTL)

Address: 21B_0000h base + 0h offset = 21B_0000h



MMDC_MDCTL field descriptions

Field	Description
31 SDE_0	MMDC Enable CS0. This bit enables/disables accesses from the MMDC toward Chip Select 0. The reset value of this bit is "0" (i.e No clocks and clock enable will be driven to the memory). At the enabling point the MMDC will perform an initialization process (including a delay on RESET and/or CKE) for both chip selects. The initialization length depends on the configured memory type. 0 Disabled 1 Enabled
30 SDE_1	MMDC Enable CS1. This bit enables/disables accesses from the MMDC toward Chip Select 1. The reset value of this bit is "0" (i.e No clocks and clock enable will be driven to the memory). At the enabling point the MMDC will perform an initialization process (including a delay on RESET and/or CKE) for both chip selects. The initialization length depends on the configured memory type. 0 Disabled 1 Enabled
29–27 Reserved	This read-only field is reserved and always has the value 0.
26–24 ROW	Row Address Width. This field specifies the number of row addresses used by the memory array. It will affect the way an incoming address will be decoded. Settings 110-111 are reserved 000 11 bits Row 001 12 bits Row 010 13 bits Row 011 14 bits Row 100 15 bits Row 101 16 bits Row
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 COL	Column Address Width. This field specifies the number of column addresses used by the memory array. It will determine how an incoming address will be decoded. 0x0 9 bits column 0x1 10 bits column 0x2 11 bits column 0x3 8 bits column 0x4 12 bits column 0x5-0xF Reserved
19 BL	Burst Length. This field determines the burst length of the DDR device. In LPDDR2 mode the MMDC supports burst length 4. In DDR3 mode the MMDC supports burst length 8. 0 Burst Length 4 is used 1 Burst Length 8 is used
18 Reserved	This read-only field is reserved and always has the value 0.
17–16 DSIZ	DDR data bus size. This field determines the size of the data bus of the DDR memory

Table continues on the next page...

MMDC_MDCTL field descriptions (continued)

Field	Description
	0 16-bit data bus — 1 Reserved — — 2-3 Reserved
Reserved	This read-only field is reserved and always has the value 0.

33.12.2 MMDC Core Power Down Control Register (MMDC_MDPDC)

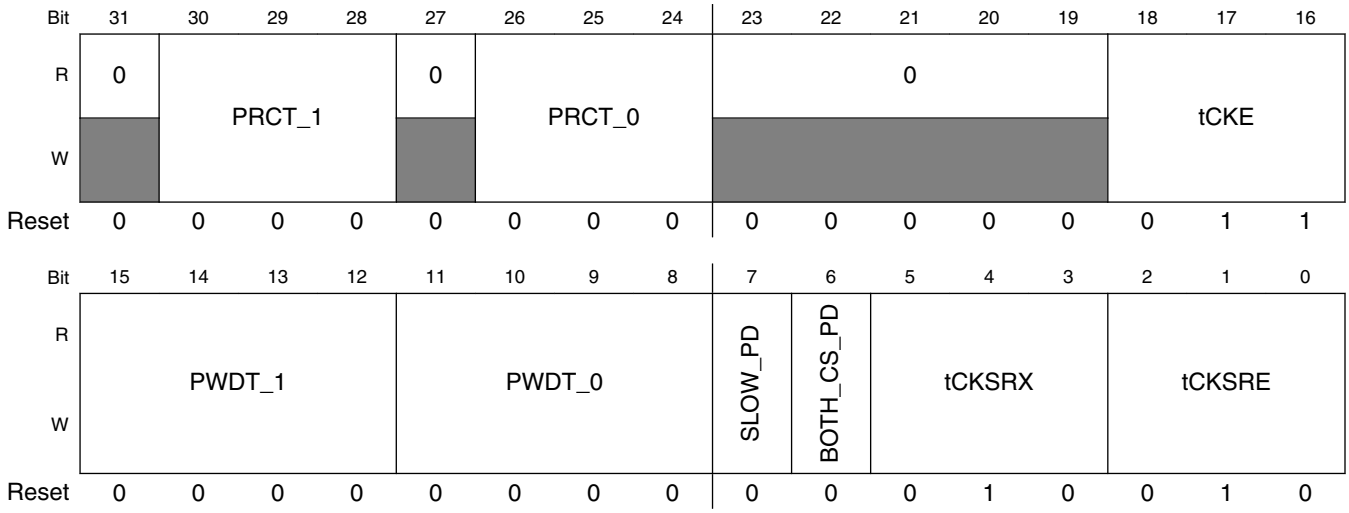
Table 33-13. PRCT field encoding

PRCT[2:0]	Precharge Timer
000	Disabled (Bit field reset value)
001	2 clocks
010	4 clocks
011	8 clocks
100	16 clocks
101	32 clocks
110	64 clocks
111	128 clocks

Table 33-14. PWDT field encoding

PWDT[3:0]	Power Down Time-out
0000	Disabled (bit field reset value)
0001	16 cycles
0010	32 cycles
0011	64 cycles
0100	128 cycles
0101	256 cycles
0110	512 cycles
0111	1024 cycles
1000	2048 cycles
1001	4096 cycles
1010	8196 cycles
1011	16384 cycles
1100	32768 cycles
1101-1111	Reserved

Address: 21B_0000h base + 4h offset = 21B_0004h



MMDC_MDPDC field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 PRCT_1	Precharge Timer - Chip Select 1. This field determines the amount of idle cycle for which chip select 1 will be automatically precharged. The amount of cycles are determined according to the PRCT Field Encoding table above.
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 PRCT_0	Precharge Timer - Chip Select 0. This field determines the amount of idle cycle for which chip select 0 will be automatically precharged. The amount of cycles are determined according to the table below.
23–19 Reserved	This read-only field is reserved and always has the value 0.
18–16 tCKE	CKE minimum pulse width. This field determines the minimum pulse width of CKE. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
15–12 PWDT_1	Power Down Timer - Chip Select 1. This field determines the amount of idle cycle for which chip select 1 will be automatically get into precharge/active power down. The amount of cycles are determined according to the PWDT Field Encoding table above.
11–8 PWDT_0	Power Down Timer - Chip Select 0. This field determines the amount of idle cycle for which chip select 0 will be automatically get into precharge/active power down. The amount of cycles are determined according to the PWDT Field Encoding table above.
7 SLOW_PD	Slow/fast power down. In DDR3 mode this field is referred to slow precharge power-down.

Table continues on the next page...

MMDC_MDPDC field descriptions (continued)

Field	Description
	In LPDDR2 mode this field is not relevant. **NOTE:** Memory should be configured the same. 0 Fast mode. 1 Slow mode.
6 BOTH_CS_PD	Parallel power down entry to both chip selects. When power down timer is used for both chip-selects (i.e PWDT_0 and PWDT1 don't equal "0") , then if this bit is enabled, the MMDC will enter power down only if the amount of idle cycles of both chip selects was obtained. 0 Each chip select can enter power down independently according to its configuration. 1 Chip selects can enter power down only if the amount of idle cycles of both chip selects was obtained.
5–3 tCKSRX	Valid clock cycles before self-refresh exit. This field determines the amount of clock cycles before self-refresh exit 0x0 0 cycle 0x1 1 cycles 0x6 6 cycles 0x7 7 cycles
tCKSRE	Valid clock cycles after self-refresh entry. This field determines the amount of clock cycles after self-refresh entry 0x0 0 cycle 0x1 1 cycles 0x6 6cycles 0x7 7cycles

33.12.3 MMDC Core ODT Timing Control Register (MMDC_MDOTC)

For further information see ODT Configuration .

Address: 21B_0000h base + 8h offset = 21B_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		tAOFPD			tAONPD			tANPD				tAXPD			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1

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

MMDC_MDOTC field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.
29–27 tAOFPD	Asynchronous RTT turn-off delay (power down with DLL frozen). This field determines the time between termination cuircuit starts to turn off the ODT resistance till termination has reached high impedance. In LPDDR2 this field is not relevant. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
26–24 tAONPD	Asynchronous RTT turn-on delay (power down with DLL frozen). This field determines the time between termination cuircuit gets out of high impedance and begins to turn on till ODT resistance are fully on. In LPDDR2 this field is not relevant. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
23–20 tANPD	Asynchronous ODT to power down entry delay. In DDR3 should be set to tCWL-1 In LPDDR2 this field is not relevant. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
19–16 tAXPD	Asynchronous ODT to power down exit delay. In DDR3 should be set to tCWL-1 In LPDDR2 this field is not relevant. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 tODTLon	ODT turn on latency. This field determines the delay between ODT signal and the associated RTT, where according to JEDEC standard it equals WL(write latency) - 2. Therefore, the value that is configured to tODTLon field should correspond the value that is configured to MDCGFG1[tCWL] In LPDDR2 this field is not relevant. 0x0 - 0x1 Reserved 0x2 2 cycles 0x3 3 cycles 0x4 4 cycles 0x5 5 cycles

Table continues on the next page...

MMDC_MDOTC field descriptions (continued)

Field	Description
	0x6 6 cycles 0x7 Reserved
11–9 Reserved	This read-only field is reserved and always has the value 0.
8–4 tODT_idle_off	ODT turn off latency. This field determines the Idle period before turning memory ODT off. **NOTE:** This field is not relevant in LPDDR2 mode. 0x0 0 cycle (turned off at the earliest possible time) 0x1 1 cycle 0x2 2 cycles 0x1E 30 cycles 0x1F 31 cycles
Reserved	This read-only field is reserved and always has the value 0.

33.12.4 MMDC Core Timing Configuration Register 0 (MMDC_MDCFG0)

Address: 21B_0000h base + Ch offset = 21B_000Ch

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

MMDC_MDCFG0 field descriptions

Field	Description
31–24 tRFC	Refresh command to Active or Refresh command time. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xFE 255 clocks 0xFF 256 clocks
23–16 tXS	Exit self refresh to non READ command. In LPDDR2 it is called tXSR, self-refresh exit to next valid command delay. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 - 0x15 reserved 0x16 23 clocks 0x17 24 clocks

Table continues on the next page...

MMDC_MDCFG0 field descriptions (continued)

Field	Description
	0xFE 255 clocks 0xFF 256 clocks
15–13 tXP	Exit power down with DLL-on to any valid command. Exit power down with DLL-frozen to commands not requiring a locked DLL In LPDDR2/LPDDR3 mode this field is referred to Exit power-down to next valid command delay. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
12–9 tXPDLL	Exit precharge power down with DLL frozen to commands requiring DLL. In LPDDR2 this field is not relevant. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
8–4 tFAW	Four Active Window (all banks). See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x1E 31 clocks 0x1F 32 clocks
tCL	CAS Read Latency. In DDR3 mode this field is referred to CL. In LPDDR2 mode this field is referred to RL. **NOTE:** In LPDDR2/LPDDR3 mode only the RL/WL pairs are allowed as specified in MR2 register See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 3 cycles 0x1 4 cycles 0x2 5 cycles 0x3 6 cycles 0x4 7 cycles 0x5 8 cycles 0x6 9 cycles

Table continues on the next page...

MMDC_MDCFG0 field descriptions (continued)

Field	Description
	0x7 10 cycles 0x8 11 cycles 0x9 - 0xF Reserved

33.12.5 MMDC Core Timing Configuration Register 1 (MMDC_MDCFG1)

Address: 21B_0000h base + 10h offset = 21B_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	tRCD			tRP			tRC					tRAS				
Reset	1	0	1	1	0	1	1	0	1	0	1	1	0	0	0	1

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

MMDC_MDCFG1 field descriptions

Field	Description
31–29 tRCD	Active command to internal read or write delay time (same bank). This field is valid only for DDR2/DDR3 memories In LPDDR2/LPDDR3 mode, this parameter should be configured at tRCD_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x3 4 clocks 0x4 5 clocks 0x5 6 clocks 0x6 7 clocks 0x7 8 clocks
28–26 tRP	Precharge command period (same bank). This field is valid only for DDR2/DDR3 memories IIn LPDDR2 mode this parameter should be configured at tRPpb_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks

Table continues on the next page...

MMDC_MDCFG1 field descriptions (continued)

Field	Description
	0x3 4 clocks 0x4 5 clocks 0x5 6 clocks 0x6 7 clocks 0x7 8 clocks
25–21 tRC	Active to Active or Refresh command period (same bank). This field is valid only for DDR2/DDR3 memories In LPDDR2/LPDDR3 mode, this parameter should be configured at tRC_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x1E 31 clocks 0x1F 32 clocks
20–16 tRAS	Active to Precharge command period (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x1E 31 clocks 0x1F Reserved
15 tRPA	Precharge-all command period. This field is valid only for DDR2/DDR3 memories In LPDDR2/LPDDR3 mode, this parameter should be configured at tRPab_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0 Will be equal to: tRP. 1 Will be equal to: tRP+1.
14–12 Reserved	This read-only field is reserved and always has the value 0.
11–9 tWR	WRITE recovery time (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles

Table continues on the next page...

MMDC_MDCFG1 field descriptions (continued)

Field	Description
	0x6 7cycles 0x7 8 cycles
8–5 tMRD	Mode Register Set command cycle (all banks). In DDR3 mode this field shoud be set to max (tMRD,tMOD). In LPDDR2/LPDDR3 mode this field should be set to max(tMRR,tMRW) See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
4–3 Reserved	This read-only field is reserved and always has the value 0.
tCWL	CAS Write Latency. In DDR3 mode this field is referred to CWL. In LPDDR2/LPDDR3 mode this field is referred to WL. 0x0 2 cycles (DDR2/DDR3) ,1 cycles (LPDDR2/LPDDR3) 0x1 3 cycles (DDR2/DDR3) ,2 cycles (LPDDR2/LPDDR3) 0x2 4 cycles (DDR2/DDR3) ,3 cycles (LPDDR2/LPDDR3) 0x3 5 cycles (DDR2/DDR3) ,4 cycles (LPDDR2/LPDDR3) 0x4 6 cycles (DDR2/DDR3) ,5 cycles (LPDDR2/LPDDR3) 0x5 7 cycles (DDR2/DDR3) ,6 cycles (LPDDR2/LPDDR3) 0x6 8 cycles (DDR2/DDR3) ,7 cycles (LPDDR2/LPDDR3) 0x7 Reserved

33.12.6 MMDC Core Timing Configuration Register 2 (MMDC_MDCFG2)

Address: 21B_0000h base + 14h offset = 21B_0014h

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

MMDC_MDCFG2 field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MDCFG2 field descriptions (continued)

Field	Description
24–16 tDLLK	DLL locking time. In LPDDR2 this field is not relevant. See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 cycle. 0x1 2 cycles. 0x2 3 cycles. 0xC7 200 cycles 0x1FE 511 cycles. 0x1FF 512 cycles (JEDEC value for DDR3).
15–9 Reserved	This read-only field is reserved and always has the value 0.
8–6 tRTP	Internal READ command to Precharge command delay (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles 0x6 7cycles 0x7 8 cycles
5–3 tWTR	Internal WRITE to READ command delay (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles 0x6 7cycles 0x7 8 cycles
tRRD	Active to Active command period (all banks). See DDR3 SDRAM Specification JESD79-3E (July 2010) , LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles

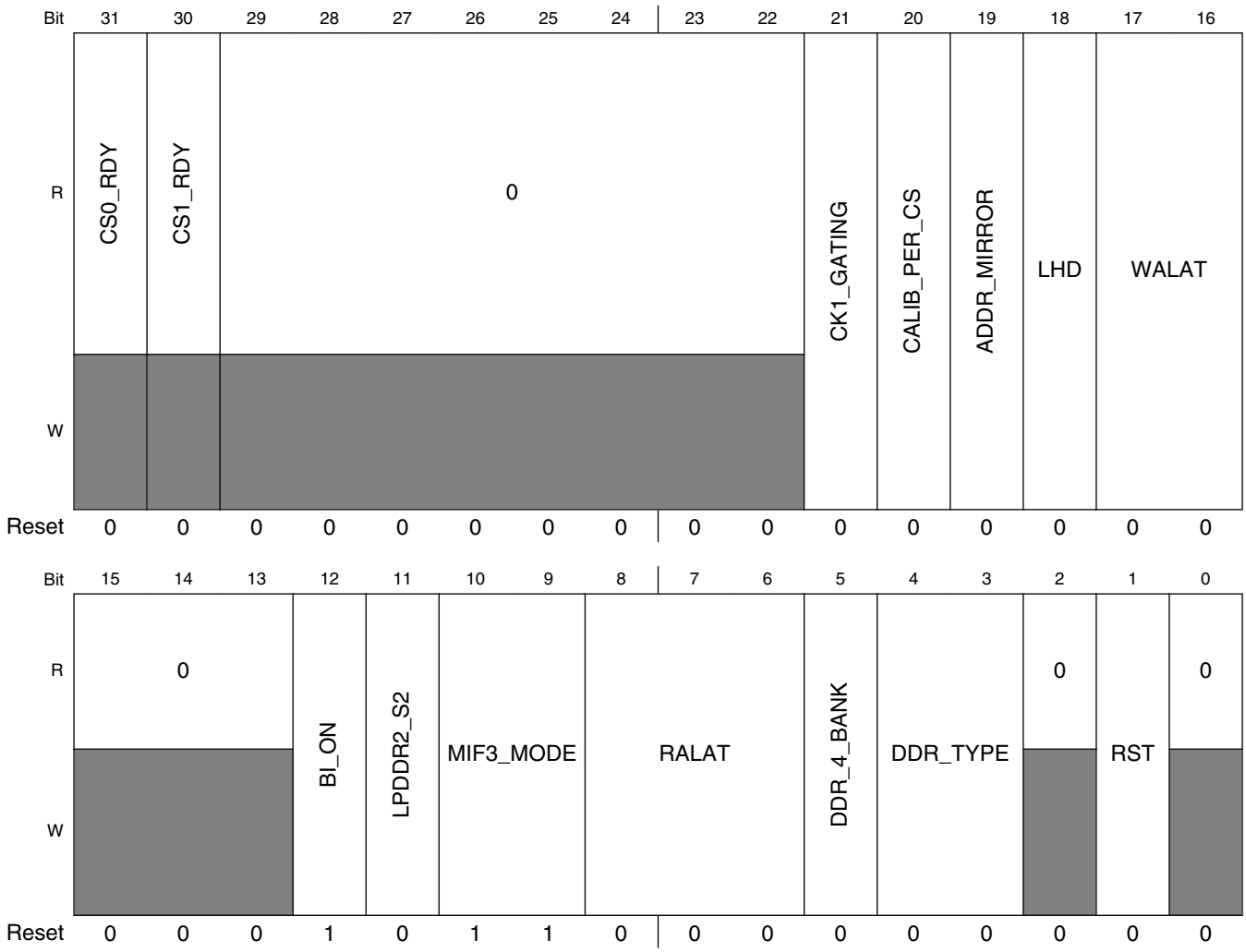
Table continues on the next page...

MMDC_MDCFG2 field descriptions (continued)

Field	Description
	0x6 7cycles 0x7 Reserved

33.12.7 MMDC Core Miscellaneous Register (MMDC_MDMISC)

Address: 21B_0000h base + 18h offset = 21B_0018h



MMDC_MDMISC field descriptions

Field	Description
31 CS0_RDY	External status device on CS0. This is a read-only status bit, that indicates whether the external memory is in wake-up period. 0 Device in wake-up period. 1 Device is ready for initialization.

Table continues on the next page...

MMDC_MDMISC field descriptions (continued)

Field	Description
30 CS1_RDY	External status device on CS1. This is a read-only status bit, that indicates whether the external memory is in wake-up period. 0 Device in wake-up period. 1 Device is ready for initialization.
29–22 Reserved	This read-only field is reserved and always has the value 0.
21 CK1_GATING	Gating the secondary DDR clock. When this bit is asserted then the MMDC will disable the secondary DDR clock 0 MMDC drives two clocks toward the DDR memory 1 MMDC drives only one clock toward the DDR memory (CK0)
20 CALIB_PER_CS	Number of chip-select for calibration process. This bit determines the chip-select index that the associated calibration is targetted to. Relevant for read, write, write leveling and read DQS gating calibrations 0 Calibration is targetted to CS0 1 Calibration is targetted to CS1
19 ADDR_MIRROR	Address mirroring. **NOTE:** This feature is not supported for LPDDR2/LPDDR3 memories. But only for DDR2/DDR3 memories. For further information see Address mirroring . 0 Address mirroring disabled. 1 Address mirroring enabled.
18 LHD	Latency hiding disable. This is a debug feature. When set to "1" the MMDC will handle one read/write access at a time. Meaning that the MMDC pipe-line will be limitted to 1 open access (next AXI address phase will be acknowledged if the current AXI data phase had finished) 0 Latency hiding on. 1 Latency hiding disable.
17–16 WALAT	Write Additional latency. In case the write-leveling calibration process indicates a delay of greater than one-eighth a clock cycle (between CK and any of the DQS strobe lines), then this field must be configured accordingly. This field will add delay on the obe I/O control, which will compensate on the additional write leveling delay on DQS and prevent the DQS from being cropped. **NOTE:** The purpose of WALAT is to add time delay at the end of a burst write operation to ensure that the JEDEC time specification for Write Post Amble Delay (tWPST) is met (DQS strobe is held low at the end of a write burst for > 30% a clock cycle before it is released). If the value of any of the WL_DL_ABS_OFFSETn register fields are greater than ‘1F’, WALAT should be set to ‘1’ (cycle additional delay). WALAT should be further increased for any full cycle delays added by the WL_CYC_DELn register fields. 0x0 No additional latency required. 0x1 1 cycle additional delay 0x2 2 cycles additional delay 0x3 3 cycles additional delay

Table continues on the next page...

MMDC_MDMISC field descriptions (continued)

Field	Description
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 BI_ON	Bank Interleaving On. This bit controls the organization of the bank, row and column address bits. For further information see Address decoding . 0 Banks are not interleaved, and address will be decoded as bank-row-column 1 Banks are interleaved, and address will be decoded as row-bank-column
11 LPDDR2_S2	LPDDR2 S2 device type indication. In case LPDDR2 device is used (DDR_TYPE = 0x1), this bit will indicate whether S2 or S4 device is used. This bit should be cleared in DDR3 mode 0x0 LPDDR2-S4 device is used. 0x1 LPDDR2-S2 device is used.
10–9 MIF3_MODE	Command prediction working mode. This field determines the level of command prediction that will be used by the MMDC 00 Disable prediction. 01 Enable prediction based on : Valid access on first pipe line stage. 10 Enable prediction based on: Valid access on first pipe line stage, Valid access on axi bus. 11 Enable prediction based on: Valid access on first pipe line stage, Valid access on axi bus, Next miss access from access queue.
8–6 RALAT	Read Additional Latency. This field determines the additional read latency which is added to CAS latency and internal delays for which the MMDC will retrieve the read data from the internal FIFO. This field is used to compensate on board/chip delays. **NOTE:** In LPDDR2 mode 2 extra cycles will be added internally in order to compensate tDQSCK delay. 0x0 no additional latency. 0x1 1 cycle additional latency. 0x2 2 cycles additional latency. 0x3 3 cycles additional latency. 0x4 4 cycles additional latency. 0x5 5 cycles additional latency. 0x6 6 cycles additional latency. 0x7 7 cycles additional latency.
5 DDR_4_BANK	Number of banks per DDR device. When this bit is set to "1" then the MMDC will work with DDR device of 4 banks. 0 8 banks device is being used. (Default) 1 4 banks device is being used
4–3 DDR_TYPE	DDR TYPE. This field determines the type of the external DDR device. 0x0 DDR3 device is used. 0x1 LPDDR2 device is used. 0x1 Reserved 0x2 Reserved 0x3 Reserved
2 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MDMISC field descriptions (continued)

Field	Description
1 RST	Software Reset. When this bit is asserted then the internal FSMs and registers of the MMDC will be initialized. **NOTE:** This bit once asserted gets deasserted automatically. 0 Do nothing. 1 Assert reset to the MMDC.
0 Reserved	This read-only field is reserved and always has the value 0.

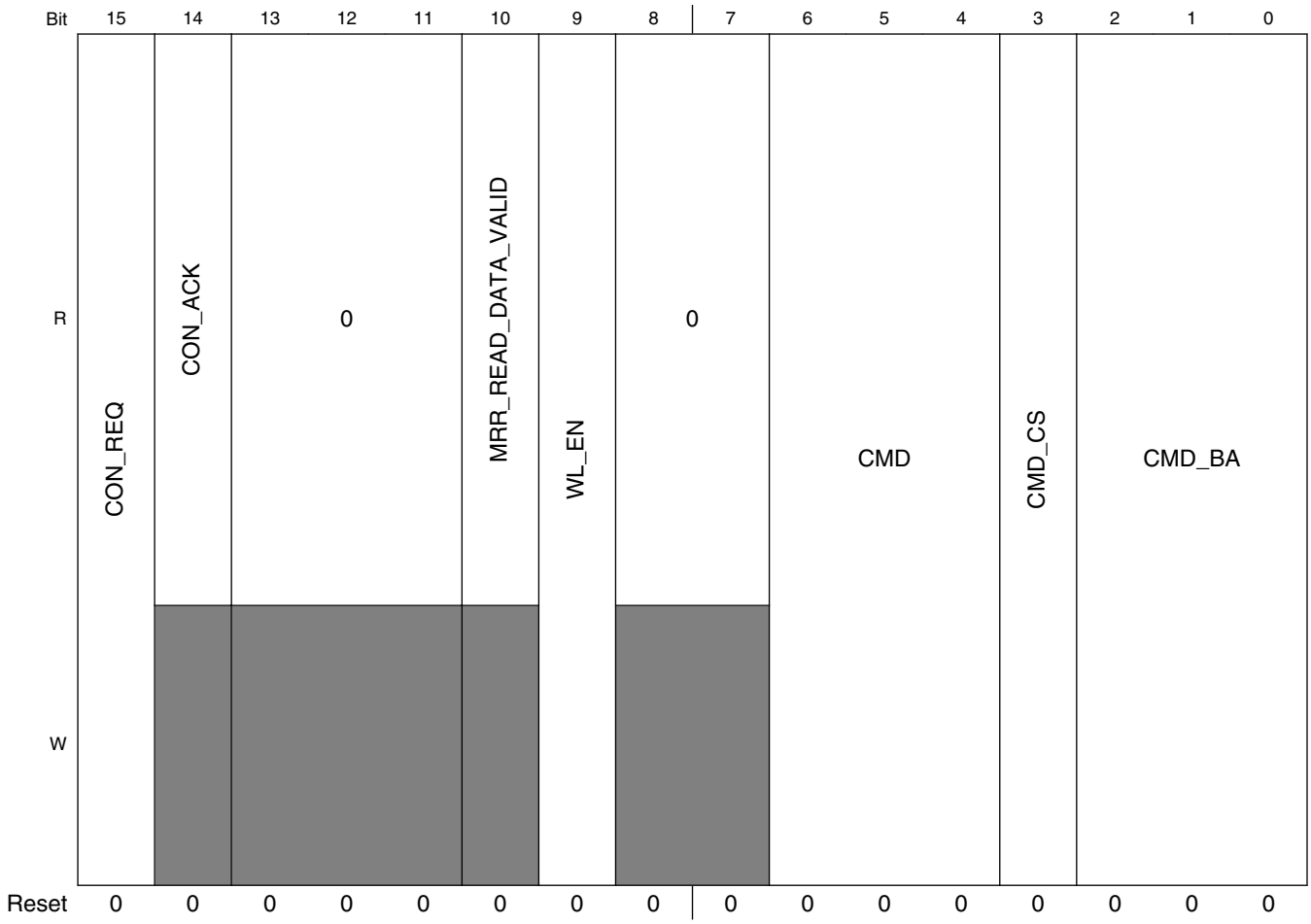
33.12.8 MMDC Core Special Command Register (MMDC_MDSCR)

This register is used to issue special commands manually toward the external DDR device (such as load mode register, manual self refresh, manual precharge and so on). Every write to this register will be interpreted as a command, and a read from this register will show the last command that was executed.

Every write to this register will result in one special command, and the IP bus will assert ips_xfr_wait as long as the special command is being carried out.

Address: 21B_0000h base + 1Ch offset = 21B_001Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CMD_ADDR_MSB_MR_OP								CMD_ADDR_LSB_MR_ADDR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MMDC_MDSCR field descriptions

Field	Description
31–24 CMD_ADDR_MSB_MR_OP	Command/Address MSB. This field indicates the MSB of the command/Address. In LPDDR2/LPDDR3 this field indicates the MRW operand
23–16 CMD_ADDR_LSB_MR_ADDR	Command/Address LSB. This field indicates the LSB of the command/Address In LPDDR2/LPDDR3 this field indicates the MRR/MRW address
15 CON_REQ	Configuration request. When this bit is set then the MMDC will clean the pending AXI accesses and will prevent from further AXI accesses to be acknowledged. This field guarantee safe configuration (or change configuration) of the MMDC while no access is in process and prevents an unexpected behaviour. After setting this bit, it is needed to poll on CON_ACK until it is set to "1". When CON_ACK is asserted then configuration is permitted. After configuration is completed then this bit must be deasserted in order to process further AXI accesses. **NOTE:** This bit is asserted at the end of the reset sequence, meaning that the MMDC is waiting to configure and intialize the external memory before accepting any AXI accesses. Configuration request/acknowledge mechanism should be used for the following procedures: changing of timing parameters , during calibration process or driving commands via MDSCR[CMD]

Table continues on the next page...

MMDC_MDSCR field descriptions (continued)

Field	Description
	0 No request to configure MMDC. 1 A request to configure MMDC is valid
14 CON_ACK	Configuration acknowledge. Whenever this bit is set, it is permitted to configure MMDC IP registers. 0 Configuration of MMDC registers is forbidden. 1 Configuration of MMDC registers is permitted.
13–11 Reserved	This read-only field is reserved and always has the value 0.
10 MRR_READ_ DATA_VALID	MRR read data valid. This field indicates that read data is valid at MDMRR register **NOTE:** This field is relevant only for LPDDR2/LPDDR3 mode 0 Cleared upon the assertion of MRR command 1 Set after MRR data is valid and stored at MDMRR register.
9 WL_EN	DQS pads direction. This bit controls the DQS pads direction during write-leveling calibration process. Before starting the write-leveling calibration process this bit should be set to "1". It should be set to "0" when sending write leveling exit command. For further information see Write leveling Calibration . 0 Exit write leveling mode or stay in normal mode. 1 Write leveling entry command was sent.
8–7 Reserved	This read-only field is reserved and always has the value 0.
6–4 CMD	Command. This field contains the command to be executed. This field will be automatically cleared after the command will be send to the DDR memory. **NOTE:** Only the 0x5 Precharge all command should be used, memory operation is unpredictable when using the 0x1 command 0x0 Normal operation 0x1 Reserved 0x2 Auto-Refresh Command (set correct CMD_CS). 0x3 Load Mode Register Command DDR2/DDR3, set correct CMD_CS, CMD_BA, CMD_ADDR_LSB, CMD_ADDR_MSB), MRW Command (LPDDR2/LPDDR3, set correct CMD_CS, MR_OP, MR_ADDR) 0x4 ZQ calibration (DDR2/DDR3, set correct CMD_CS, {CMD_ADDR_MSB,CMD_ADDR_LSB} = 0x400 or 0x0) 0x5 Precharge all, only if banks open (set correct CMD_CS). 0x6 MRR command (LPDDR2/LPDDR3, set correct CMD_CS, MR_ADDR) 0x7 Reserved
3 CMD_CS	Chip Select. This field determines which chip select the command is targeted to 0 to Chip-select 0 1 to Chip-select 1
CMD_BA	Bank Address. This field determines the address of the bank within the selected chip-select where the command is targetted to. 0x0 bank address 0

Table continues on the next page...

MMDC_MDSCR field descriptions (continued)

Field	Description
	0x1 bank address 1 0x2 bank address 2 0x7 bank address 7

33.12.9 MMDC Core Refresh Control Register (MMDC_MDREF)

This register determines the refresh scheme that will be executed toward the DDR device. It specifies how often a refresh cycle occurs and how many refresh commands will be executed every refresh cycle.

For further information see Refresh Scheme .

The following tables show examples of possible refresh schemes.

Table 33-15. Refresh rate example for REF_SEL = 0

REFR[2:0]	Number of refresh commands every 64KHz	Average periodic refresh rate (tREFI)	System Refresh period
0x0	1	15.6 µs	tRFC
0x1	2	7.8 µs	2*tRFC
0x3	4	3.9µs	4*tRFC
0x7	8	1.95 µs	8*tRFC

Table 33-16. Refresh rate example for REF_SEL = 1

REFR[2:0]	Number of refresh commands every 32KHz	Average periodic refresh rate (tREFI)	System Refresh period
0x1	2	15.6 µs	2*tRFC
0x3	4	7.8 µs	4*tRFC
0x7	8	3.9µs	8*tRFC

Table 33-17. Refresh rate example for REF_SEL = 2@ 400MHz

REFR[2:0]	Number of refresh commands every refresh cycle	REF_CNT	Average periodic refresh rate (tREFI)	System Refresh period
0x0	1	0x618	3.9 µs	tRFC
0x1	2	0xC30	3.9 µs	2*tRFC

Table continues on the next page...

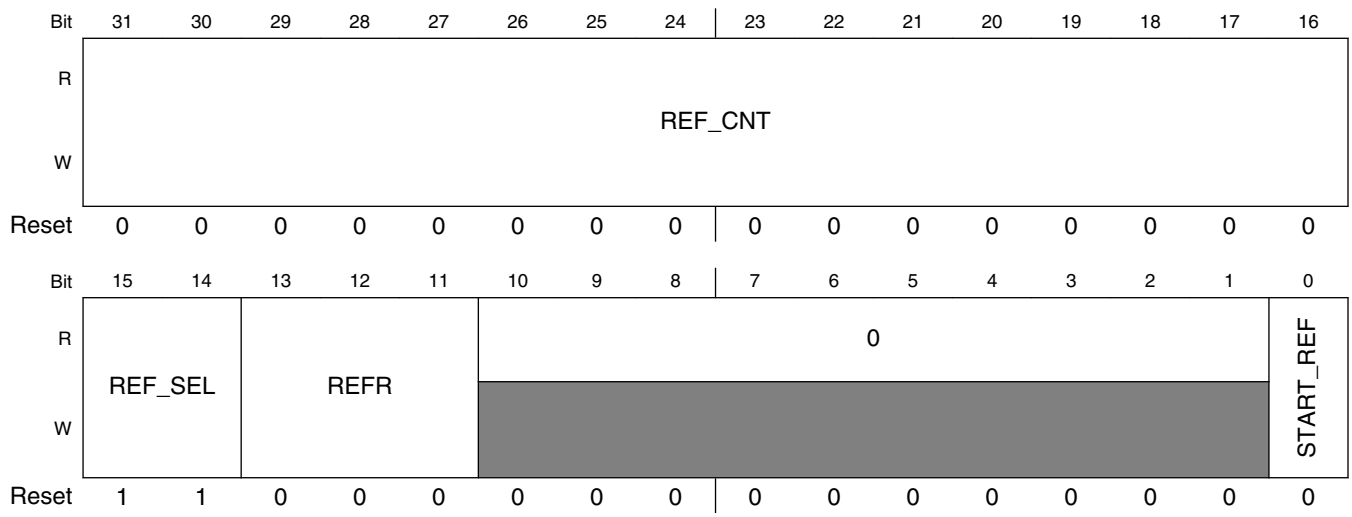
Table 33-17. Refresh rate example for REF_SEL = 2@ 400MHz (continued)

REFR[2:0]	Number of refresh commands every refresh cycle	REF_CNT	Average periodic refresh rate (tREFI)	System Refresh period
0x2	3	0x1248	3.9µs	3*tRFC
0x3	4	0x1860	3.9 µs	4*tRFC

Other refresh configurations are also allowed; the configuration values in the tables above are only examples for obtaining the desired average periodic refresh rate.

If the required average periodic refresh rate (tREFI) is kept, all of the rows will be refreshed in every refresh window. Because the memory device issues additional refresh commands for every refresh it receives, the tREFI remains the same across the device, regardless of its number of rows. This is particularly relevant in the tRFC parameter, which becomes bigger as the density increases.

Address: 21B_0000h base + 20h offset = 21B_0020h



MMDC_MDREF field descriptions

Field	Description
31–16 REF_CNT	Refresh Counter at DDR clock period If REF_SEL equals '2' a refresh cycle will begin every amount of DDR cycles configured in this field. 0x0 Reserved. 0x1 1 cycle. 0xFFFE 65534 cycles. 0xFFFF 65535 cycles.
15–14 REF_SEL	Refresh Selector. This bit selects the source of the clock that will trigger each refresh cycle:

Table continues on the next page...

MMDC_MDREF field descriptions (continued)

Field	Description
	0 Periodic refresh cycles will be triggered in frequency of 64KHz. 1 Periodic refresh cycles will be triggered in frequency of 32KHz. 2 Periodic refresh cycles will be triggered every amount of cycles that are configured in REF_CNT field. 3 No refresh cycles will be triggered.
13–11 REFR	Refresh Rate. This field determines how many refresh commands will be issued every refresh cycle. After every refresh command the MMDC won't drive any command to the DDR device untill satisfying tRFC period 0x0 1 refresh 0x1 2 refreshes 0x2 3 refreshes 0x3 4 refreshes 0x4 5 refreshes 0x5 6 refreshes 0x6 7 refreshes 0x7 8 refreshes
10–1 Reserved	This read-only field is reserved and always has the value 0.
0 START_REF	Manual start of refresh cycle. When this field is set to '1' the MMDC will start a refresh cycle immediately according to number of refresh commands that are configured in 'REFR' field. This bit returns to zero automatically. 0 Do nothing. 1 Start a refresh cycle.

33.12.10 MMDC Core Read/Write Command Delay Register (MMDC_MDRWD)

This register determines the delay between back to back read and write accesses. The register reset values are set to the minimum required value. As the default values are set to achieve optimal results, changing them is discouraged.

Address: 21B_0000h base + 2Ch offset = 21B_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			tDAI												
W																
Reset	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1

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

MMDC_MDRWD field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–16 tDAI	Device auto initialization period.(maximum) **NOTE:** This field is relevant only to LPDDR2 mode 0x0 1 cycle 0xF9F 4000 cycles (Default, JEDEC value for LPDDR2, gives 10us at 400MHz clock). 0x1FFF 8192 cycles
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 RTW_SAME	Read to write delay for the same chip-select. This field controls the delay between read to write commands toward the same chip select. The total delay is calculated according to: BL/2 + RTW_SAME + (tCL-tCWL) + RALAT 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles (Default) 0x3 3 cycles 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles
11–9 WTR_DIFF	Write to read delay for different chip-select. This field controls the delay between write to read commands toward different chip select. The total delay is calculated according to: BL/2 + WTR_DIFF + (tCL-tCWL) + RALAT 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles 0x3 3 cycles (Default) 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles
8–6 WTW_DIFF	Write to write delay for different chip-select. This field controls the delay between write to write commands toward different chip select. The total delay is calculated according to: BL/2 + WTW_DIFF 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles 0x3 3 cycles (Default) 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles

Table continues on the next page...

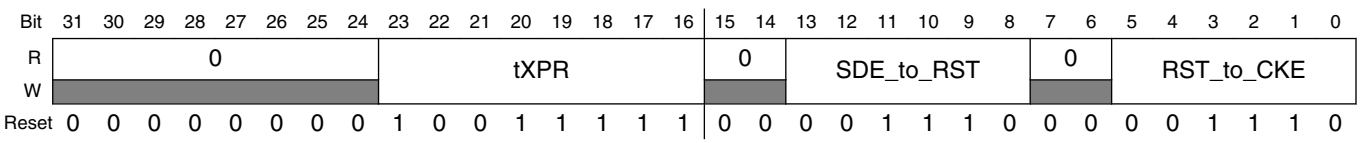
MMDC_MDRWD field descriptions (continued)

Field	Description
5–3 RTW_DIFF	Read to write delay for different chip-select. This field controls the delay between read to write commands toward different chip select. The total delay is calculated according to: BL/2 + RTW_DIFF + (tCL - tCWL) + RALAT 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles (Default) 0x3 3 cycles 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles
RTR_DIFF	Read to read delay for different chip-select. This field controls the delay between read to read commands toward different chip select. The total delay is calculated according to: BL/2 + RTR_DIFF 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles (Default) 0x3 3 cycles 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles

33.12.11 MMDC Core Out of Reset Delays Register (MMDC_MDOR)

This register defines delays that must be kept when MMDC exits reset.

Address: 21B_0000h base + 30h offset = 21B_0030h



MMDC_MDOR field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MDOR field descriptions (continued)

Field	Description
23–16 tXPR	DDR2/DDR3: CKE HIGH to a valid command. LPDDR2: Not relevant to this mode. Please leave in default reset value. DDR3: As defined in timing parameter table. 0x0 Reserved 0x1 2 cycles 0x2 3 cycles 0xFE 255 cycles 0xFF 256 cycles
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 SDE_to_RST	DDR3: Time from SDE enable until DDR reset# is high. In LPDDR2 mode this field is not relevant. **NOTE:** Each cycle in this field is 15.258 us. 0x0 Reserved 0x1 Reserved 0x2 Reserved 0x3 1 cycles 0x4 2 cycles 0x10 14 cycles (JEDEC value for DDR3) - total of 200 us 0x3E 60 cycles 0x3F 61 cycles
7–6 Reserved	This read-only field is reserved and always has the value 0.
RST_to_CKE	DDR3: Time from SDE enable to CKE rise. In case that DDR reset# is low, will wait until it's high and thenwait this period until rising CKE. (JEDEC value is 500 us) LPDDR2: Idle time ater first CKE assertion. (JEDEC value is 200 us) **NOTE:** Each cycle in this field is 15.258 us. 0x0 Reserved 0x1 Reserved 0x2 Reserved 0x3 1 cycles 0x10 14 cycles (JEDEC value for LPDDR2) - total of 200 us 0x23 33 cycles (JEDEC value for DDR3) - total of 500 us 0x3E 60 cycles 0x3F 61 cycles

33.12.12 MMDC Core MRR Data Register (MMDC_MDMRR)

This register contains data that was collected after issuing MRR command. The data in this register is valid only when MDSCR[MRR_READ_DATA_VALID] is set to "1".

This register is relevant only in LPDDR2 mode. For further information see LPDDR2 Refresh Rate Update and Timing Derating .

Address: 21B_0000h base + 34h offset = 21B_0034h

Bit	31–24	23–16	15–8	7–0
R	Reserved	Reserved	MRR_READ_DATA1	MRR_READ_DATA0
W				
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

MMDC_MDMRR field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 -	This field is reserved. Reserved
15–8 MRR_READ_DATA1	MRR DATA that arrived on DQ[15:8]
MRR_READ_DATA0	MRR DATA that arrived on DQ[7:0]

33.12.13 MMDC Core Timing Configuration Register 3 (MMDC_MDCFG3LP)

This register is relevant only for LPDDR2 mode.

Address: 21B_0000h base + 38h offset = 21B_0038h

Bit	31–22	21–16	15–12	11–8	7–4	3–0
R	0	RC_LP	0	tRCD_LP	tRPpb_LP	tRPab_LP
W						
Reset	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

MMDC_MDCFG3LP field descriptions

Field	Description
31–22 Reserved	This read-only field is reserved and always has the value 0.
21–16 RC_LP	Active to Active or Refresh command period (same bank). This field is valid only for LPDDR2 memories 0x0 1 clock 0x1 2 clocks 0x2 3 clocks

Table continues on the next page...

MMDC_MDCFG3LP field descriptions (continued)

Field	Description
	0x3E 63 clocks 0x3F Reserved
15–12 Reserved	This read-only field is reserved and always has the value 0.
11–8 tRCD_LP	Active command to internal read or write delay time (same bank). This field is valid only for LPDDR2 memories 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF Reserved
7–4 tRPpb_LP	Precharge (per bank) command period (same bank). This field is valid only for LPDDR2 memories 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF Reserved
tRPab_LP	Precharge (all banks) command period. This field is valid only for LPDDR2 memories 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF Reserved

33.12.14 MMDC Core MR4 Derating Register (MMDC_MDMR4)

This register is relevant only for LPDDR2 mode. It is used to dynamically change certain values depending on MR4 read result, which is based on memory temperature sensor result.

Address: 21B_0000h base + 3Ch offset = 21B_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							tRRD_DE	tRP_DE	tRAS_DE	tRC_DE	tRCD_DE	0		UPDATE_DE_ACK	UPDATE_DE_REQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

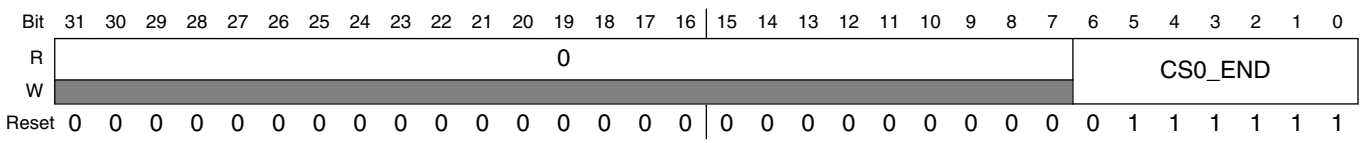
MMDC_MDMR4 field descriptions

Field	Description
31–9 Reserved	This read-only field is reserved and always has the value 0.
8 tRRD_DE	tRRD derating value. 0 Original tRRD is used. 1 tRRD is derated in 1 cycle.
7 tRP_DE	tRP derating value. 0 Original tRP is used. 1 tRP is derated in 1 cycle.
6 tRAS_DE	tRAS derating value. 0 Original tRAS is used. 1 tRAS is derated in 1 cycle.
5 tRC_DE	tRC derating value. 0 Original tRC is used. 1 tRC is derated in 1 cycle.
4 tRCD_DE	tRCD derating value. 0 Original tRCD is used. 1 tRCD is derated in 1 cycle.
3–2 Reserved	This read-only field is reserved and always has the value 0.
1 UPDATE_DE_ACK	Update Derated Values Acknowledge. This read only bit will be cleared upon UPDATE_DE_REQ assertion and will be set after the new values are taken.
0 UPDATE_DE_REQ	Update Derated Values Request. This read modify write field is automatically cleared after the request is issued. 0 Do nothing. 1 Request to update the following values: tRRD, tRCD, tRP, tRC, tRAS and refresh related fields(MDREF register): REF_CNT, REF_SEL, REFR

33.12.15 MMDC Core Address Space Partition Register (MMDC_MDASP)

This register defines the partitioning between chip select 0 and chip select 1. For further information see Chip select settings .

Address: 21B_0000h base + 40h offset = 21B_0040h



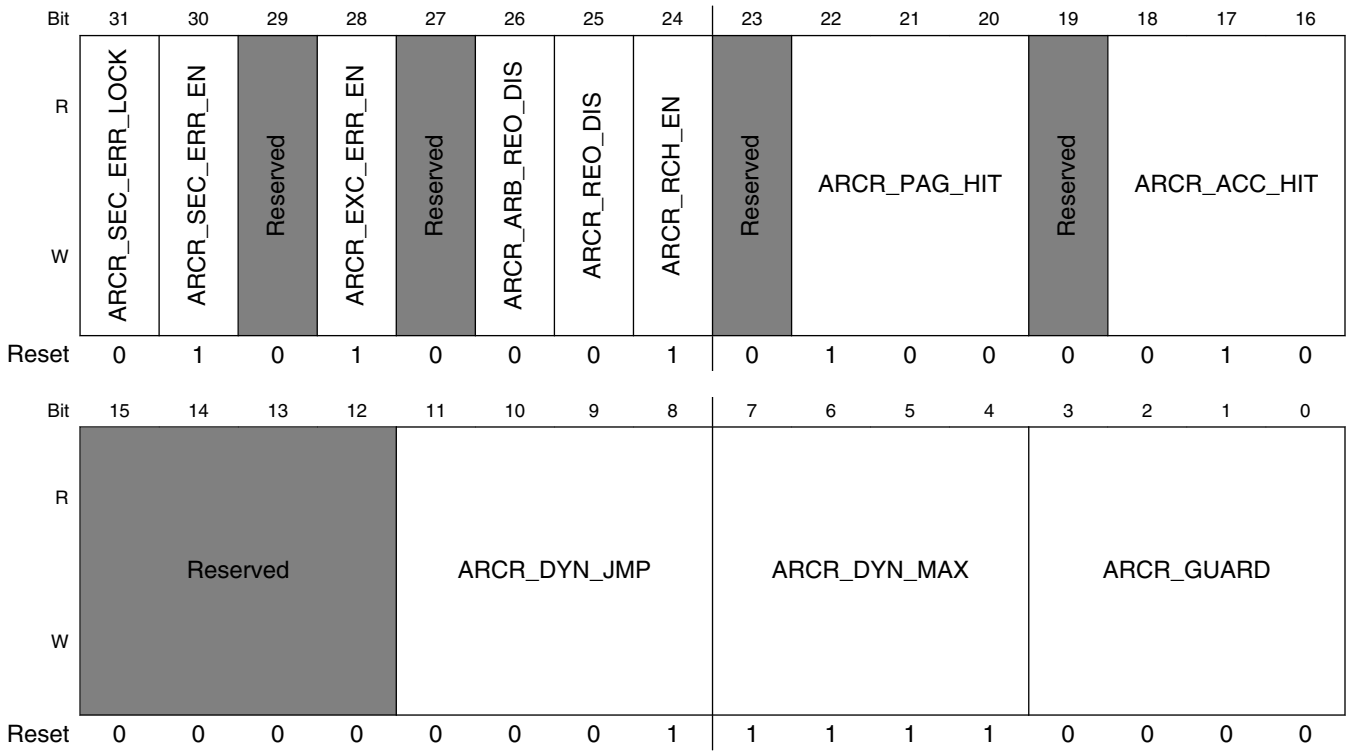
MMDC_MDASP field descriptions

Field	Description
31–7 Reserved	This read-only field is reserved and always has the value 0.
CS0_END	CS0_END. Defines the absolute last address associated with CS0 with increments of 256Mb. CS0_END=AXI_ADDRESS[31:25] bits. MMDC0_MDASP[CS0_END] should be set to DDR_CS_SIZE/32M + 0x3f (channel 0 base address begins at 0x80000000)

33.12.16 MMDC Core AXI Reordering Control Register (MMDC_MAARCR)

This register determines the values of the weights used for the re-ordering arbitration engine. For further information see Performance .

Address: 21B_0000h base + 400h offset = 21B_0400h



MMDC_MAARCR field descriptions

Field	Description
31 ARCR_SEC_ ERR_LOCK	Once set, this bit locks ARCR_SEC_ERR_EN and prevents from its updating. This bit can be only cleared by reset Default value is 0x0 - encoding 0 (unlocked) 0 ARCR_SEC_ERR_EN is unlocked, so can be updated any moment 1 ARCR_SEC_ERR_EN is locked, so it can't be updated
30 ARCR_SEC_ ERR_EN	This bit defines whether security read/write access violation result in SLV Error response or in OKAY response Default value is 0x1 - encoding 1(response is SLV Error, rresp/bresp=2'b10) 0 security violation results in OKAY response (rresp/bresp=2'b00) 1 security violation results in SLAVE Error response (rresp/bresp=2'b10)

Table continues on the next page...

MMDC_MAARCR field descriptions (continued)

Field	Description
29 -	This field is reserved. Reserved
28 ARCR_EXC_ERR_EN	This bit defines whether exclusive read/write access violation of AXI 6.2.4 rule result in SLV Error response or in OKAY response Default value is 0x1 - encoding 1(response is SLV Error) 0 violation of AXI exclusive rules (6.2.4) result in OKAY response (rresp/bresp=2'b00) 1 violation of AXI exclusive rules (6.2.4) result in SLAVE Error response (rresp/bresp=2'b10)
27 -	This field is reserved. Reserved
26 ARCR_ARB_REO_DIS	0 MMDC arbitration and reordering controls enabled 1 MMDC arbitration and reordering controls disabled
25 ARCR_REO_DIS	0 MMDC reordering controls only enabled 1 MMDC reordering controls only disabled
24 ARCR_RCH_EN	This bit defines whether Real time channel is activated and bypassed all other pending accesses, So accesses with QoS=='F' will be granted the highest priority in the optimization/reordering mechanism Default value is 0x1 - encoding 1 (Enabled) 0 normal prioritization, no bypassing 1 accesses with QoS=='F' bypass the arbitration
23 -	This field is reserved. Reserved
22–20 ARCR_PAG_HIT	ARCR Page Hit Rate. This value will be added by the optimization/reordering mechanism to any pending access that is targeted to an open DDR row. Default value of ARCR_PAG_HIT is 0x00100 - encoding 4.
19 -	This field is reserved. Reserved
18–16 ARCR_ACC_HIT	ARCR Access Hit Rate. This value will be added by the optimization/reordering mechanism to any pending access that has the same access type (read/write) as the previous access. Default value of is ARCR_ACC_HIT 0x0010 - encoding 2.
15–12 -	This field is reserved. Reserved
11–8 ARCR_DYN_JMP	ARCR Dynamic Jump. Each time an access wan't chosen by the optimization/reordering mechanism then its dynamic score will be incremented by ARCR_DYN_JMP value. **NOTE:** Setting ARCR_DYN_JMP may cause starvation of low priority accesses **NOTE:** ARCR_DYN_JMP must be smaller than ARCR_DYN_MAX Default ARCR_DYN_JMP value is 0x0001 - encoding 1
7–4 ARCR_DYN_MAX	ARCR Dynamic Maximum. ARCR_DYN_MAX is the maximum dynamic score value that each access inside the optimization/reordering mechanism can get. 0000 0 0001 1 1111 15 (default)

Table continues on the next page...

MMDC_MAARCR field descriptions (continued)

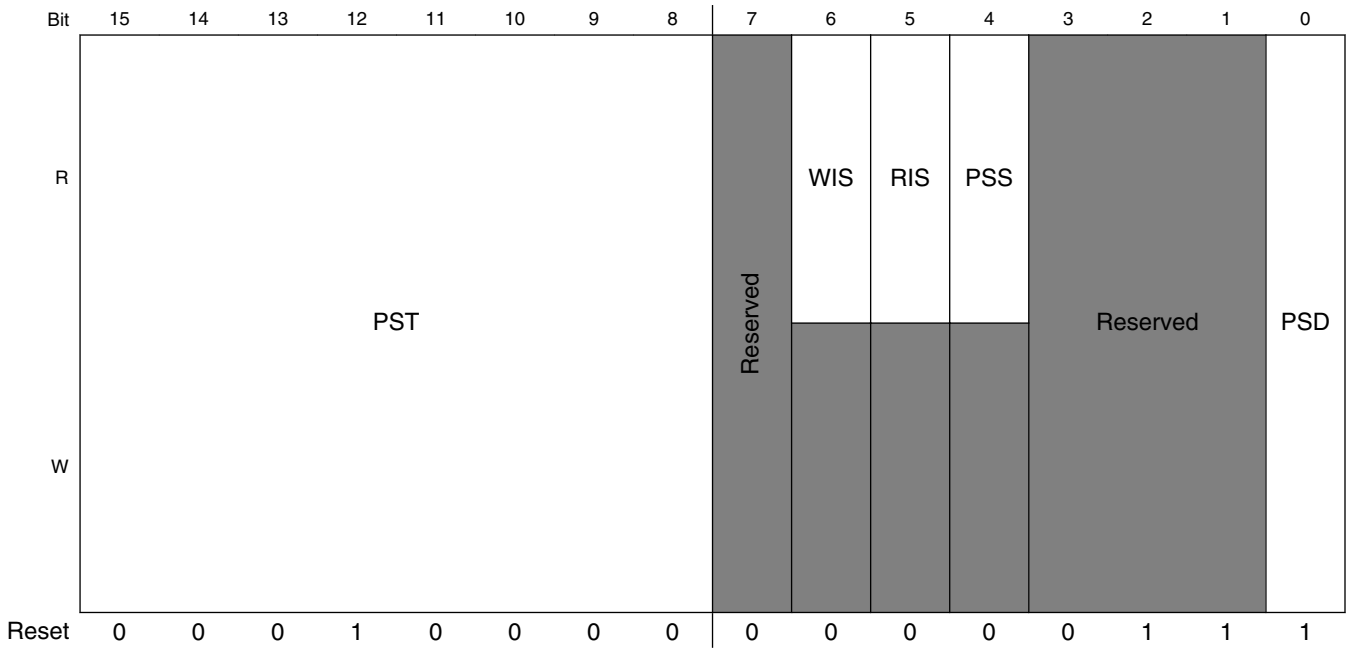
Field	Description
ARCR_GUARD	ARCR Guard. After an access reached the maximum dynamic score value, it will wait additional ARCR_GUARD arbitration cycles and then will gain the highest priority in the optimization/reordering mechanism. 0000 15 (default) 0001 16 1111 30

33.12.17 MMDC Core Power Saving Control and Status Register (MMDC_MAPSR)

The MAPSR determines the power saving features of MMDC. For further information see Power Saving and Clock Frequency Change modes .

Address: 21B_0000h base + 404h offset = 21B_0404h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						DVACK	LPACK	Reserved		DVFS	LPMD	Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MMDC_MAPSR field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 DVACK	General DVFS acknowledge. This read only bit indicates whether a dvfs acknowledge was asserted and that MMDC is in self-refresh mode
24 LPACK	General low-power acknowledge. This read only bit indicates whether a low-power acknowledge was asserted and that MMDC is in self-refresh mode
23–22 -	This field is reserved. Reserved
21 DVFS	General DVFS request. SW request for DVFS. Assertion of this bit will yield in self-refresh entry sequence 0 no dvfs request 1 dvfs request
20 LPMD	General LPMD request. SW request for LPMD. Assertion of this bit will yield in self-refresh entry sequence 0 no lpmd request 1 lpmd request
19–16 -	This field is reserved. Reserved
15–8 PST	Automatic Power saving timer. Valid only when PSD is set to "0". When the MMDC is idle for amount of cycles specified in that field then the DDR device will be entered automatically into self-refresh mode. The real value which is used is register-value multiplied by 64. 00000000 Reserved - this value is forbidden. 00000001 timer is configured to 64 clock cycles. 00000010 timer is configured to 128 clock cycles.

Table continues on the next page...

MMDC_MAPSR field descriptions (continued)

Field	Description
	00010000 (Default)- 1024 clock cycles. 11111111 timer clock is configured to 16320 clock cycles.
7 -	This field is reserved. Reserved.
6 WIS	Write Idle Status. This read only bit indicates whether write request buffer is idle (empty) or not. 0 idle 1 not idle
5 RIS	Read Idle Status. This read only bit indicates whether read request buffer is idle (empty) or not. 0 idle 1 not idle
4 PSS	Power Saving Status. This read only bit indicates whether the MMDC is in automatic power saving mode. 0 not in power saving 1 power saving
3–1 -	This field is reserved. Reserved.
0 PSD	Automatic Power Saving Disable. When the value of PSD is "0" (i.e automatic power saving is enabled) then the PST is activated and MMDC will enter automatically to self-refresh while the number of idle cycle reached. **NOTE:** This bit must be disabled (i.e set to "1") during calibration process 0 power saving enabled 1 power saving disabled (default)

33.12.18 MMDC Core Exclusive ID Monitor Register0 (MMDC_MAEXIDR0)

This register defines the ID to be monitored for exclusive accesses of monitor0 and monitor1. For further information see Exclusive accesses handling .

Address: 21B_0000h base + 408h offset = 21B_0408h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EXC_ID_MONITOR1																EXC_ID_MONITOR0															
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MAEXIDR0 field descriptions

Field	Description
31–16 EXC_ID_ MONITOR1	This field defines ID for Exclusive monitor#1. Default value is 0x0020
EXC_ID_ MONITOR0	This field defines ID for Exclusive monitor#0. Default value is 0x0000

33.12.19 MMDC Core Exclusive ID Monitor Register1 (MMDC_MAEXIDR1)

This register defines the ID to be monitored for exclusive accesses of monitor2 and monitor3. For further information see Exclusive accesses handling .

Address: 21B_0000h base + 40Ch offset = 21B_040Ch

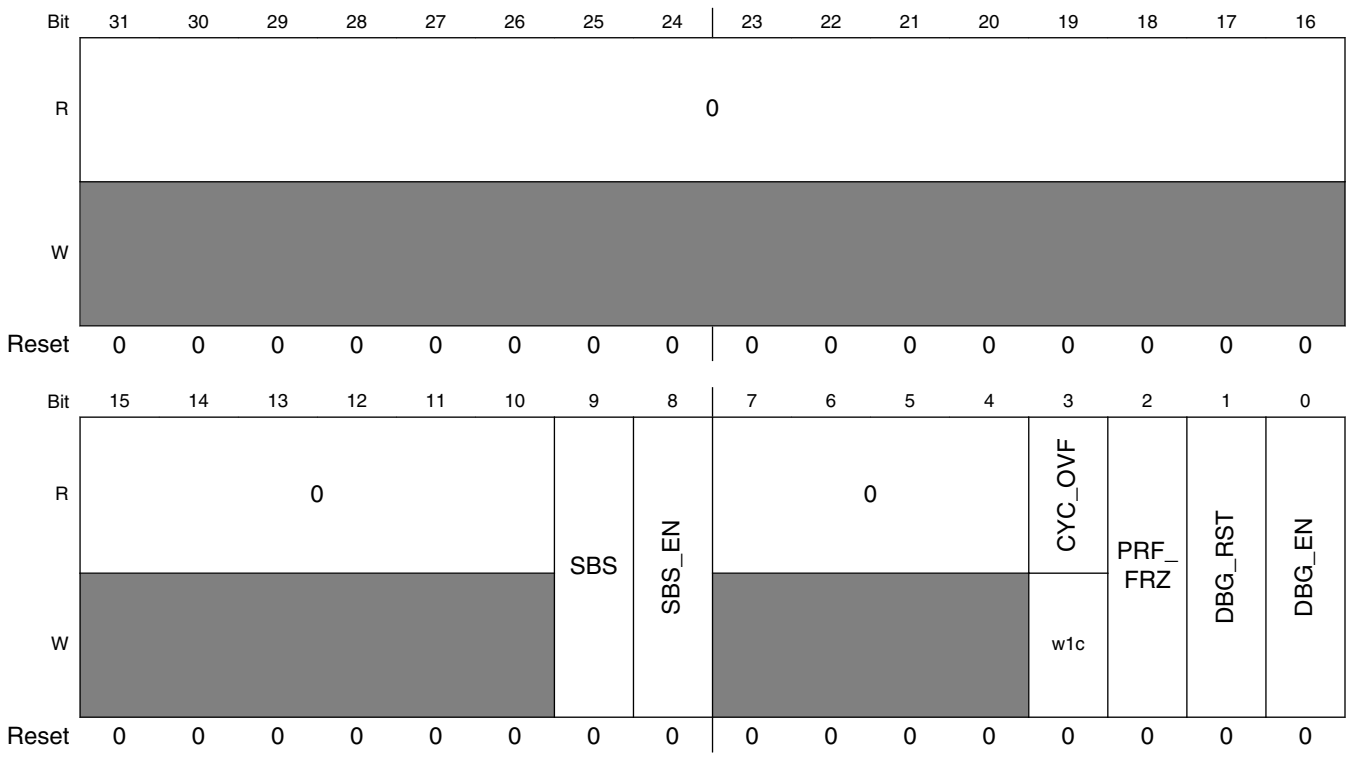
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EXC_ID_MONITOR3																EXC_ID_MONITOR2															
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

MMDC_MAEXIDR1 field descriptions

Field	Description
31–16 EXC_ID_ MONITOR3	This field defines ID for Exclusive monitor#3. Default value is 0x0060
EXC_ID_ MONITOR2	This field defines ID for Exclusive monitor#2. Default value is 0x0040

33.12.20 MMDC Core Debug and Profiling Control Register 0 (MMDC_MADPCR0)

Address: 21B_0000h base + 410h offset = 21B_0410h



MMDC_MADPCR0 field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
9 SBS	Step By Step trigger. If SBS_EN is set to "1" then dispatching AXI pending access toward the DDR will done only if this bit is set to "1", otherwise no access will be dispatched toward the DDR. This bit is cleared when the pending access has been issued toward the DDR device. 1 Launch AXI pending access toward the DDR 0 No access will be launched toward the DDR
8 SBS_EN	Step By Step debug Enable. Enable step by step mode. Every time this mechanism is enabled then setting SBS to "1" will dispatch one pending AXI access to the DDR and in parallel its attributes will be observed in the status registes (MASBS0 and MASBS1). For further information see Step By Step (SBS) software monitor . 0 disable 1 enable
7–4 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MADPCR0 field descriptions (continued)

Field	Description
3 CYC_OVF	Total Profiling Cycles Count Overflow. When profiling mechanism is enabled (DBG_EN is set to "1") then this bit is asserted when overflow of CYC_COUNT occurred. Cleared by writing 1 to it. 0 no overflow 1 overflow
2 PRF_FRZ	Profiling freeze. When this bit is asserted then the profiling mechanism will be froze and the associated status registers (MADPSR0-MADPSR5) will hold the the current profiling values. 0 profiling counters are not frozen 1 profiling counters are frozen
1 DBG_RST	Debug and Profiling Reset. Reset all debug and profiling counters and components. 0 no reset 1 reset
0 DBG_EN	Debug and Profiling Enable. Enable debug and profiling mechanism. When this bit is asserted then the MMDC will perform a profiling based on the ID that is configured to MADPCR1. Upon assertion of PRF_FRZ the profiling will be froze and the profiling results will be sampled to the status registers (MADPSR0-MADPSR5). For further information see MMDC Profiling . default is "disable" 0 disable 1 enable

33.12.21 MMDC Core Debug and Profiling Control Register 1 (MMDC_MADPCR1)

Address: 21B_0000h base + 414h offset = 21B_0414h

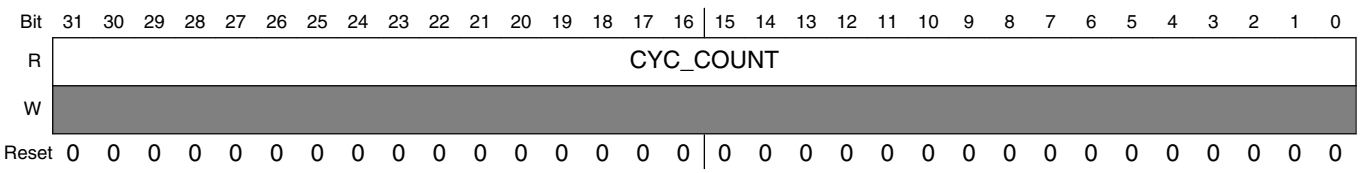
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRF_AXI_ID_MASK																PRF_AXI_ID															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MADPCR1 field descriptions

Field	Description
31–16 PRF_AXI_ID_ MASK	Profiling AXI ID Mask. AXI ID bits which masked by this value are chosen for profiling. 1 AXI ID specific bit is chosen for profiling 0 AXI ID specific bit is ignored (don't care)
PRF_AXI_ID	Profiling AXI ID. AXI IDs that matches a bit-wise AND logic operation between PRF_AXI_ID and PRF_AXI_ID_MASK are chosen for profiling. Default value is 0x0, to choose any ID-s for profiling

33.12.22 MMDC Core Debug and Profiling Status Register 0 (MMDC_MADPSR0)

Address: 21B_0000h base + 418h offset = 21B_0418h



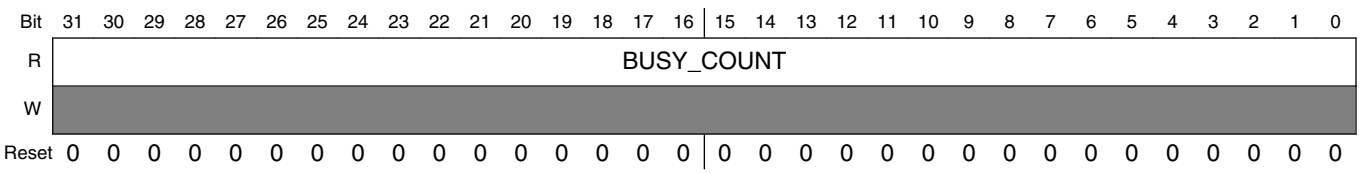
MMDC_MADPSR0 field descriptions

Field	Description
CYC_COUNT	Total Profiling cycle Count. This field reflects the total cycle count in case the profiling mechanism is enabled from assertion of DBG_EN and until PRF_FRZ is asserted

33.12.23 MMDC Core Debug and Profiling Status Register 1 (MMDC_MADPSR1)

The register reflects the total cycles during which the MMDC state machines were busy (both writes and reads). This information can be used for DDR Utilization calculation.

Address: 21B_0000h base + 41Ch offset = 21B_041Ch



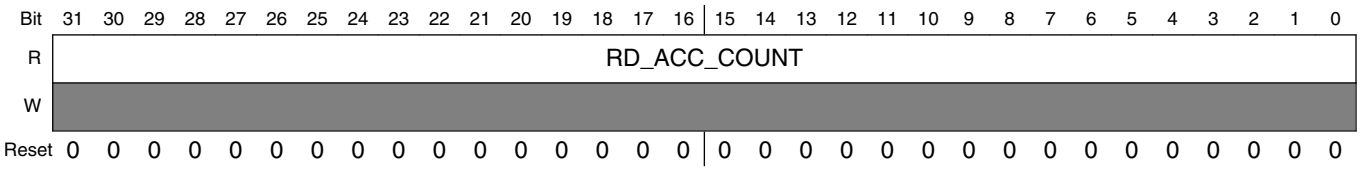
MMDC_MADPSR1 field descriptions

Field	Description
BUSY_COUNT	Profiling Busy Cycles Count. This field reflects the total number of cycles where the MMDC read and write state machines were busy during the profiling period. Can be used for DDR utilization calculations. Busy cycles are any MMDC clock cycles where the internal state machine is not idle. If any read or write requests are pending in the FIFOs, the MMDC is not idle.

33.12.24 MMDC Core Debug and Profiling Status Register 2 (MMDC_MADPSR2)

This register reflects the total number of read accesses (per AXI ID) toward MMDC.

Address: 21B_0000h base + 420h offset = 21B_0420h



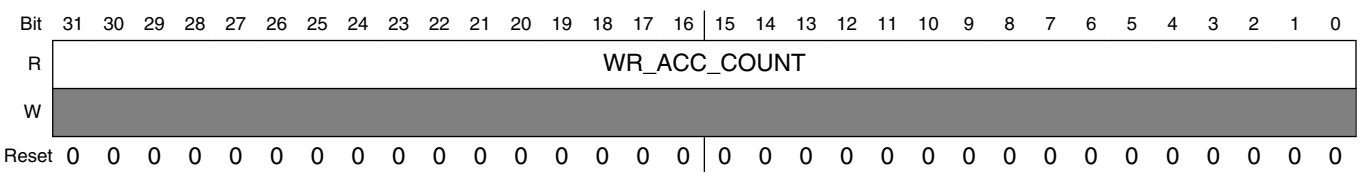
MMDC_MADPSR2 field descriptions

Field	Description
RD_ACC_ COUNT	Profiling Read Access Count. This register reflects the total number of read accesses (per AXI ID) toward MMDC.

33.12.25 MMDC Core Debug and Profiling Status Register 3 (MMDC_MADPSR3)

This register reflects the total number of write accesses (per AXI ID) toward MMDC.

Address: 21B_0000h base + 424h offset = 21B_0424h



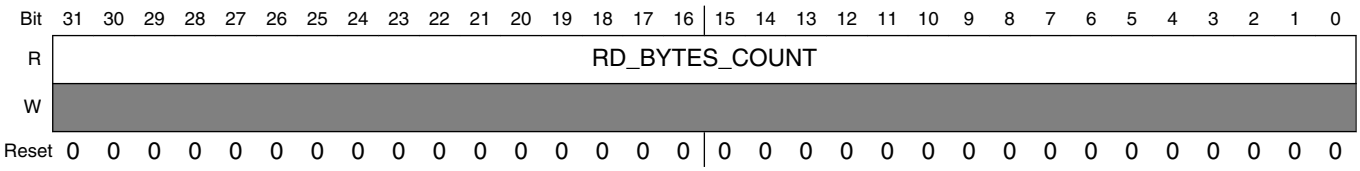
MMDC_MADPSR3 field descriptions

Field	Description
WR_ACC_ COUNT	Profiling Write Access Count. This register reflects the total number of write accesses (per AXI ID) toward MMDC.

33.12.26 MMDC Core Debug and Profiling Status Register 4 (MMDC_MADPSR4)

This register reflects the total number of bytes that were transferred during read access (per AXI ID) toward MMDC.

Address: 21B_0000h base + 428h offset = 21B_0428h



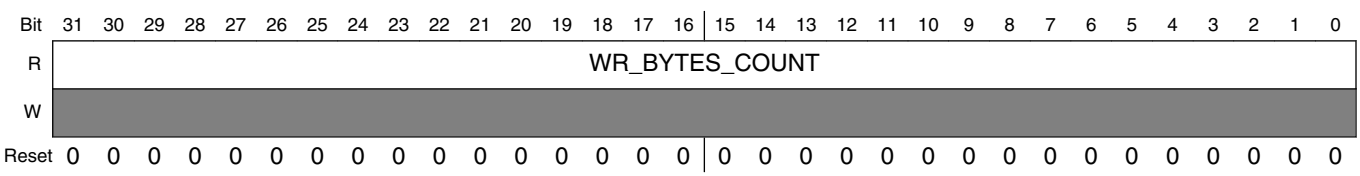
MMDC_MADPSR4 field descriptions

Field	Description
RD_BYTES_ COUNT	Profiling Read Bytes Count. This register reflects the total number of bytes that were transferred during read access (per AXI ID) toward MMDC.

33.12.27 MMDC Core Debug and Profiling Status Register 5 (MMDC_MADPSR5)

This register reflects the total number of bytes that were transferred during write access (per AXI ID) toward MMDC.

Address: 21B_0000h base + 42Ch offset = 21B_042Ch

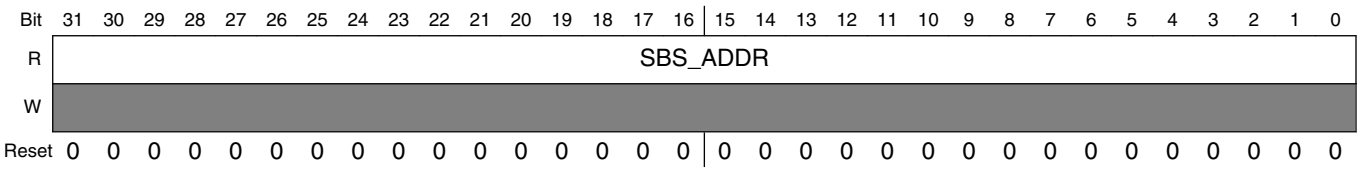


MMDC_MADPSR5 field descriptions

Field	Description
WR_BYTES_ COUNT	Profiling Write Bytes Count. This register reflects the total number of bytes that were transferred during write access (per AXI ID) toward MMDC.

33.12.28 MMDC Core Step By Step Address Register (MMDC_MASBS0)

Address: 21B_0000h base + 430h offset = 21B_0430h

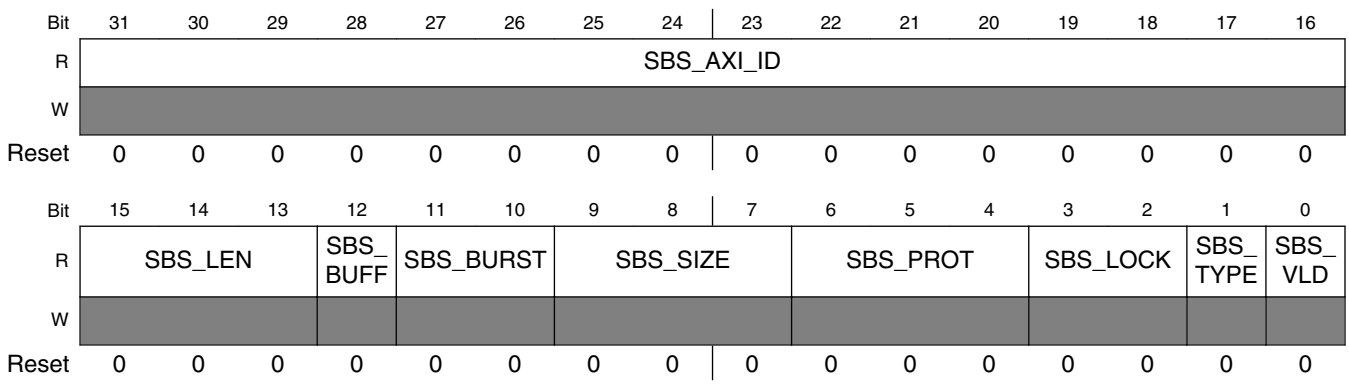


MMDC_MASBS0 field descriptions

Field	Description
SBS_ADDR	Step By Step Address. These bits reflect the address of the pending request in case of step by step mode.

33.12.29 MMDC Core Step By Step Address Attributes Register (MMDC_MASBS1)

Address: 21B_0000h base + 434h offset = 21B_0434h



MMDC_MASBS1 field descriptions

Field	Description
31–16 SBS_AXI_ID	Step By Step AXI ID. These bits reflect the AXI ID of the pending request in case of step by step mode.
15–13 SBS_LEN	Step By Step Length. These bits reflect the AXI LENGTH of the pending request in case of step by step mode. 000 burst of length 1 001 burst of length 2 111 burst of length 8
12 SBS_BUFF	Step By Step Buffered. This bit reflect the AXI CACHE[0] of the pending request in case of step by step mode. Relevant only for write requests

Table continues on the next page...

MMDC_MASBS1 field descriptions (continued)

Field	Description
11–10 SBS_BURST	Step By Step Burst. These bits reflect the AXI BURST of the pending request in case of step by step mode. 00 FIXED 01 INCR burst 10 WRAP burst 11 reserved
9–7 SBS_SIZE	Step By Step Size. These bits reflect the AXI SIZE of the pending request in case of step by step mode. 000 8 bits 001 16 bits 010 32 bits 011 64 bits 100 128bits 101-111 Reserved
6–4 SBS_PROT	Step By Step Protection. These bits reflect the AXI PROT of the pending request in case of step by step mode.
3–2 SBS_LOCK	Step By Step Lock. These bits reflect the AXI LOCK of the pending request in case of step by step mode.
1 SBS_TYPE	Step By Step Request Type. These bits reflect the type (read/write) of the pending request in case of step by step mode. 0 write 1 read
0 SBS_VLD	Step By Step Valid. This bit reflects whether there is a pending request in case of step by step mode. 0 not valid 1 valid

33.12.30 MMDC Core General Purpose Register (MMDC_MAGENP)

This register is a general 32 bit read/write register.

Address: 21B_0000h base + 440h offset = 21B_0440h

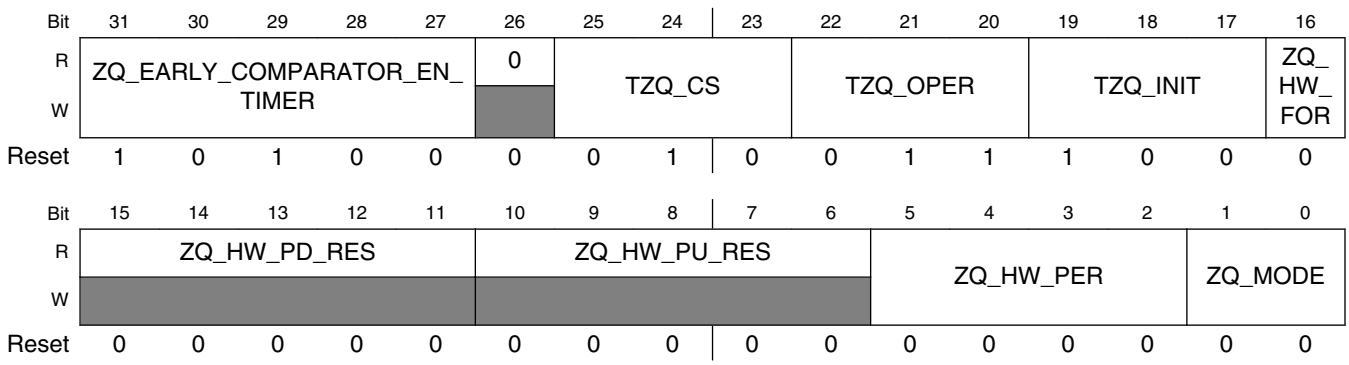
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP31_GP0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MAGENP field descriptions

Field	Description
GP31_GP0	General purpose read/write bits.

33.12.31 MMDC PHY ZQ HW control register (MMDC_MPZQHWCTRL)

Address: 21B_0000h base + 800h offset = 21B_0800h



MMDC_MPZQHWCTRL field descriptions

Field	Description
31–27 ZQ_EARLY_COMPARATOR_EN_TIMER	ZQ early comparator enable timer. This timer defines the interval between the warming up of the comparator of the i.MX ZQ calibration pad and the beginning of the ZQ calibration process with the pad 0x0 - 0x6 Reserved 0x7 8 cycles 0x14 21 cycles (Default) 0x1E 31 cycles 0x1F 32 cycles
26 Reserved	This read-only field is reserved and always has the value 0.
25–23 TZQ_CS	Device ZQ short time. This field holds the number of cycles that are required by the external DDR device to perform ZQ short calibration. Upon driving the command to the DDR device then no further accesses will be issued to the DDR device till satisfying that time. **NOTE:** In LPDDR2 the ZQ short time is taken from MPZQLP2CTL[ZQ_LP2_HW_ZQCS] **NOTE:** This field should not be update during ZQ calibration. 000 Reserved 001 Reserved 010 128 cycles (Default) 011 256 cycles 100 512 cycles 101 1024 cycles 110- 111 Reserved
22–20 TZQ_OPER	Device ZQ long/oper time. This field holds the number of cycles that are required by the external DDR device to perform ZQ long calibration except the first ZQ long command that is issued after reset. Upon driving the command to the DDR device then no further accesses will be issued to the DDR device till satisfying that time. **NOTE:** In LPDDR2 the ZQ oper time is taken from MPZQLP2CTL[ZQ_LP2_HW_ZQCL]

Table continues on the next page...

MMDC_MPZQHWCTRL field descriptions (continued)

Field	Description
	NOTE: This field should not be update during ZQ calibration. 000 Reserved 001 Reserved 010 128 cycles 011 256 cycles - Default (JEDEC value for DDR3) 100 512 cycles 101 1024 cycles 110- 111 Reserved
19–17 TZQ_INIT	Device ZQ long/init time. This field holds the number of cycles that are required by the external DDR device to perform ZQ long calibration right after reset. Upon driving the command to the DDR device then no further accesses will be issued to the DDR device till satisfying that time. **NOTE:** In LPDDR2 the ZQ init time is taken from MPZQLP2CTL[ZQ_LP2_HW_ZQINIT] **NOTE:** This field should not be update during ZQ calibration. 000 Reserved 001 Reserved 010 128 cycles 011 256 cycles 100 512 cycles - Default (JEDEC value for DDR3) 101 1024 cycles 110- 111 Reserved
16 ZQ_HW_FOR	Force ZQ automatic calibration process with the i.MX ZQ calibration pad. When this bit is asserted then the MMDC will issue one ZQ automatic calibration process with the i.MX ZQ calibration pad. It is the user responsibility to make sure that all the accesses to DDR will be finished before asserting this bit using CON_REQ/CON_ACK mechanism. HW will negate this bit upon completion of the ZQ calibration process. Upon negation of this bit the ZQ HW calibration pull-up and pull-down results (ZQ_HW_PU_RES and ZQ_HW_PD_RES respectively) are valid **NOTE:** In order to enable this bit ZQ_MODE must be set to either "1" or "3"
15–11 ZQ_HW_PD_ RES	ZQ HW calibration pull-down result. This field holds the pull-down resistor value calculated at the end of the ZQ automatic calibration process with the i.MX ZQ calibration pad. **NOTE:** An offset can be applied to this result, see MMDC_MPPDCMPR2[ZQ_PD_OFFSET]. 00000 Max. resistance. 11111 Min. resistance.
10–6 ZQ_HW_PU_ RES	ZQ automatic calibration pull-up result. This field holds the pull-up resistor value calculated at the end of the ZQ automatic calibration process with the i.MX ZQ calibration pad. **NOTE:** An offset can be applied to this result, see MMDC_MPPDCMPR2[ZQ_PU_OFFSET] 00000 Min. resistance. 11111 Max. resistance.
5–2 ZQ_HW_PER	ZQ periodic calibration time. This field determines how often the periodic ZQ calibration is performed. This field is applied for both ZQ short calibration and ZQ automatic calibration process with i.MX ZQ calibration pad. Whenever this timer is expired then according to ZQ_MODE the ZQ automatic calibration process with the i.MX ZQ calibration pad will be issued and/or short/long command will be issued to the external DDR device.

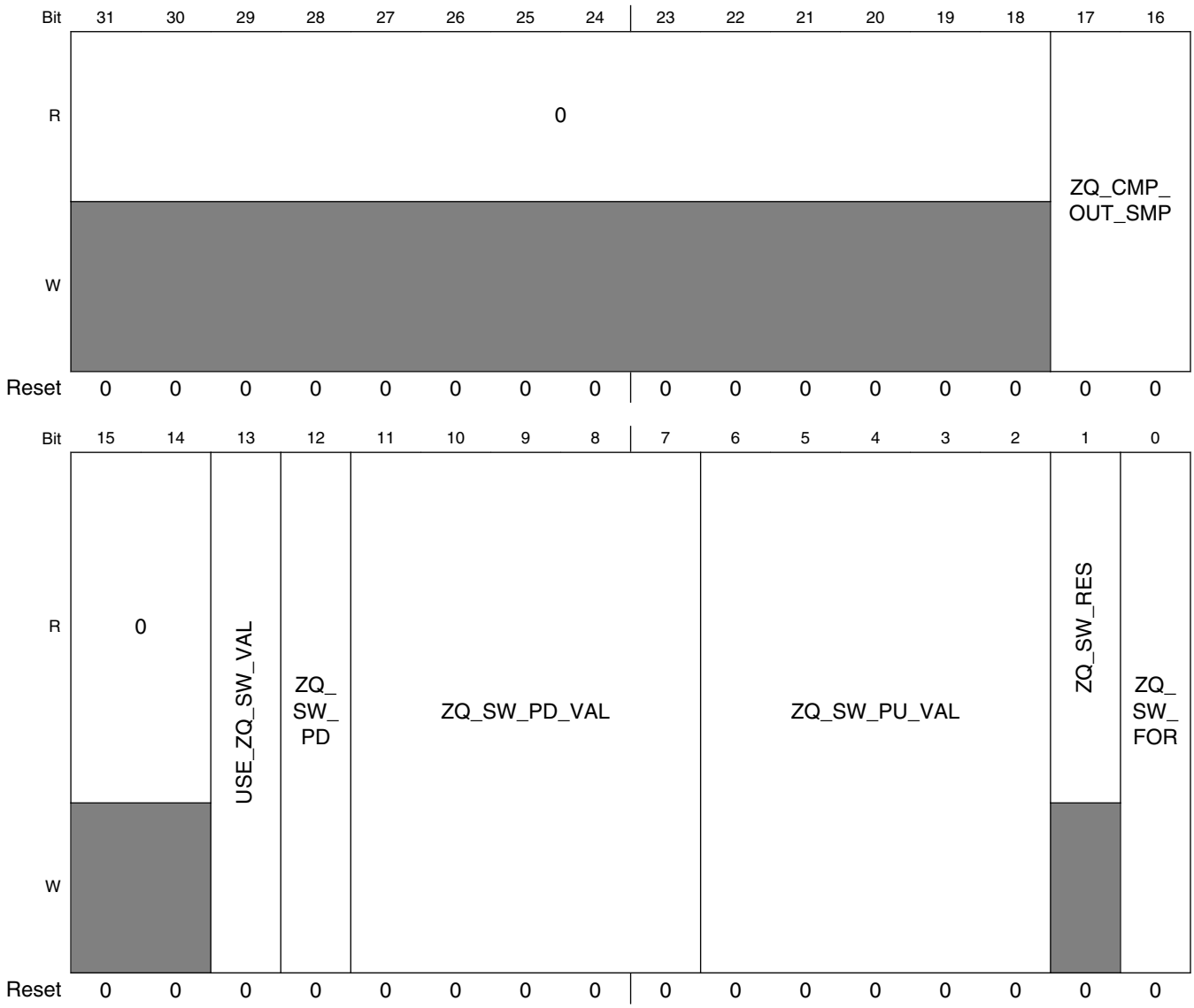
Table continues on the next page...

MMDC_MPZQHWCTRL field descriptions (continued)

Field	Description
	This field is ignored if ZQ_MODE equals "00" 0000 ZQ calibration is performed every 1 ms. 0001 ZQ calibration is performed every 2 ms. 0010 ZQ calibration is performed every 4 ms. 1010 ZQ calibration is performed every 1 sec. 1110 ZQ calibration is performed every 16 sec. 1111 ZQ calibration is performed every 32 sec.
ZQ_MODE	ZQ calibration mode: 0x0 No ZQ calibration is issued. (Default) 0x1 ZQ calibration is issued to i.MX ZQ calibration pad together with ZQ long command to the external DDR device only when exiting self refresh. 0x2 ZQ calibration command long/short is issued only to the external DDR device periodically and when exiting self refresh 0x3 ZQ calibration is issued to i.MX ZQ calibration pad together with ZQ calibration command long/ short to the external DDR device periodically and when exiting self refresh

33.12.32 MMDC PHY ZQ SW control register (MMDC_MPZQSWCTRL)

Address: 21B_0000h base + 804h offset = 21B_0804h



MMDC_MPZQSWCTRL field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 ZQ_CMP_OUT_SMP	Defines the amount of cycles between driving the ZQ signals to the ZQ pad and till sampling the comparator enable output while performing ZQ calibration process with the i.MX ZQ calibration pad 00 7 cycles 01 15 cycles

Table continues on the next page...

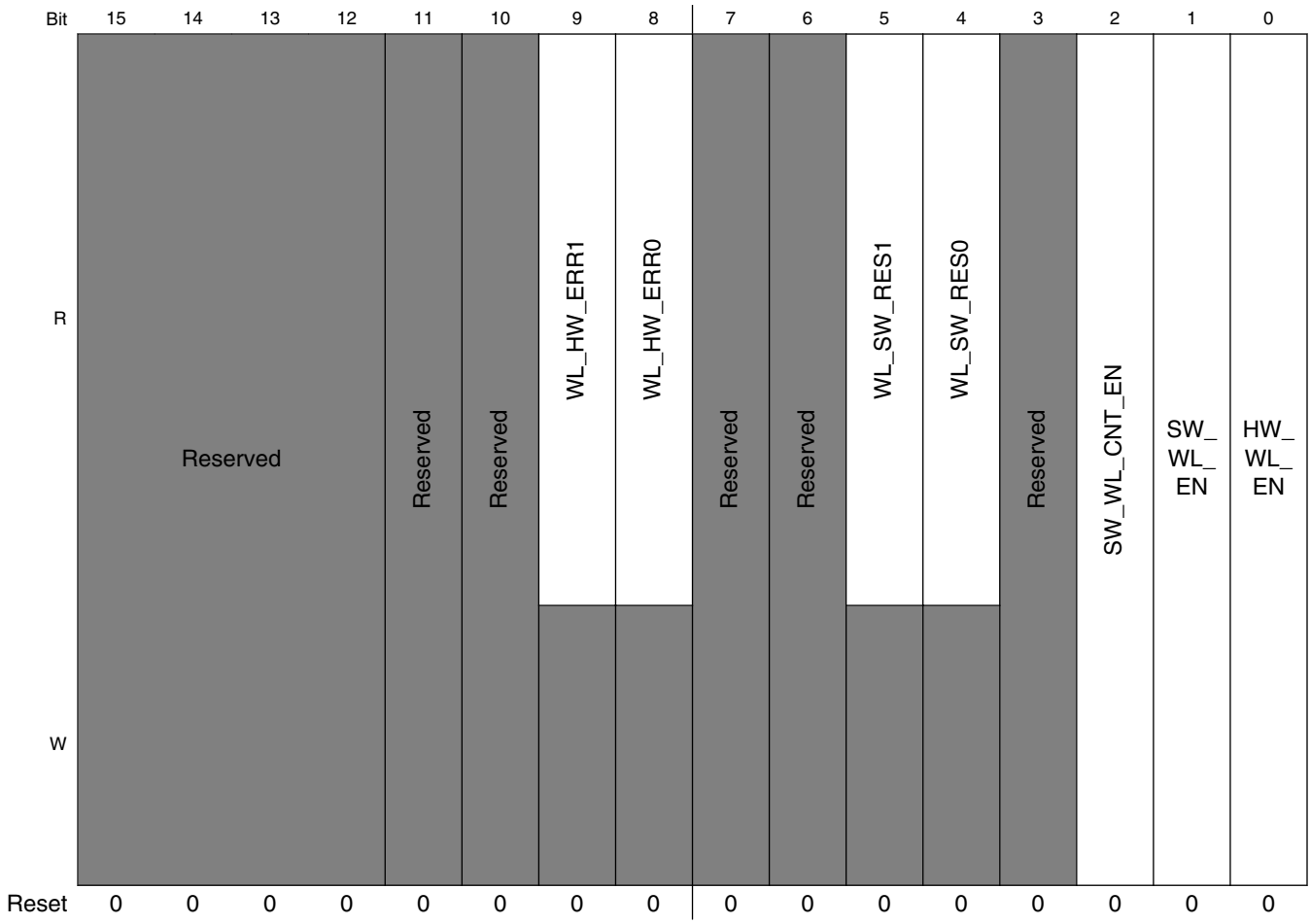
MMDC_MPZQSWCTRL field descriptions (continued)

Field	Description
	10 23 cycles 11 31 cycles
15–14 Reserved	This read-only field is reserved and always has the value 0.
13 USE_ZQ_SW_VAL	Use SW ZQ configured value for I/O pads resistor controls. This bit selects whether ZQ SW value or ZQ HW value will be driven to the I/O pads resistor controls. By default this bit is cleared and MMDC drives the HW ZQ status bits on the resistor controls of the I/O pads. **NOTE:** This bit should not be updated during ZQ calibration. 0 Fields ZQ_HW_PD_VAL & ZQ_HW_PU_VAL will be driven to I/O pads resistor controls. 1 Fields ZQ_SW_PD_VAL & ZQ_SW_PU_VAL will be driven to I/O pads resistor controls.
12 ZQ_SW_PD	ZQ software PU/PD calibration. This bit determines the calibration stage (PU or PD). 0 PU resistor calibration 1 PD resistor calibration
11–7 ZQ_SW_PD_VAL	ZQ software pull-down resistence. This field determines the value of the PD resistor during SW ZQ calibration. 00000 Max. resistance. 11111 Min. resistance.
6–2 ZQ_SW_PU_VAL	ZQ software pull-up resistence. This field determines the value of the PU resistor during SW ZQ calibration. 00000 Min. resistance. 11111 Max. resistance.
1 ZQ_SW_RES	ZQ software calibration result. This bit reflects the ZQ calibration voltage comparator value. 0 Current ZQ calibration voltage is less than VDD/2. 1 Current ZQ calibration voltage is more than VDD/2
0 ZQ_SW_FOR	ZQ SW calibration enable. This bit when asserted enables ZQ SW calibration. HW negates this bit upon completion of the ZQ SW calibration. Upon negation of this bit the ZQ SW calibration result (i.e ZQ_SW_RES) is valid

33.12.33 MMDC PHY Write Leveling Configuration and Error Status Register (MMDC_MPWLGCR)

Address: 21B_0000h base + 808h offset = 21B_0808h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MMDC_MPWLGCR field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9 WL_HW_ERR1	Byte1 write-leveling HW calibration error. This bit is asserted when an error was found on byte1 during write-leveling HW calibration. This bit is valid only upon completion of the write-leveling HW calibration (i.e. HW_WL_EN bit is de-asserted) 0 No error was found on byte1 during write-leveling HW calibration. 1 An error was found on byte1 during write-leveling HW calibration.
8 WL_HW_ERR0	Byte0 write-leveling HW calibration error. This bit is asserted when an error was found on byte0 during write-leveling HW calibration. This bit is valid only upon completion of the write-leveling HW calibration (i.e. HW_WL_EN bit is de-asserted)

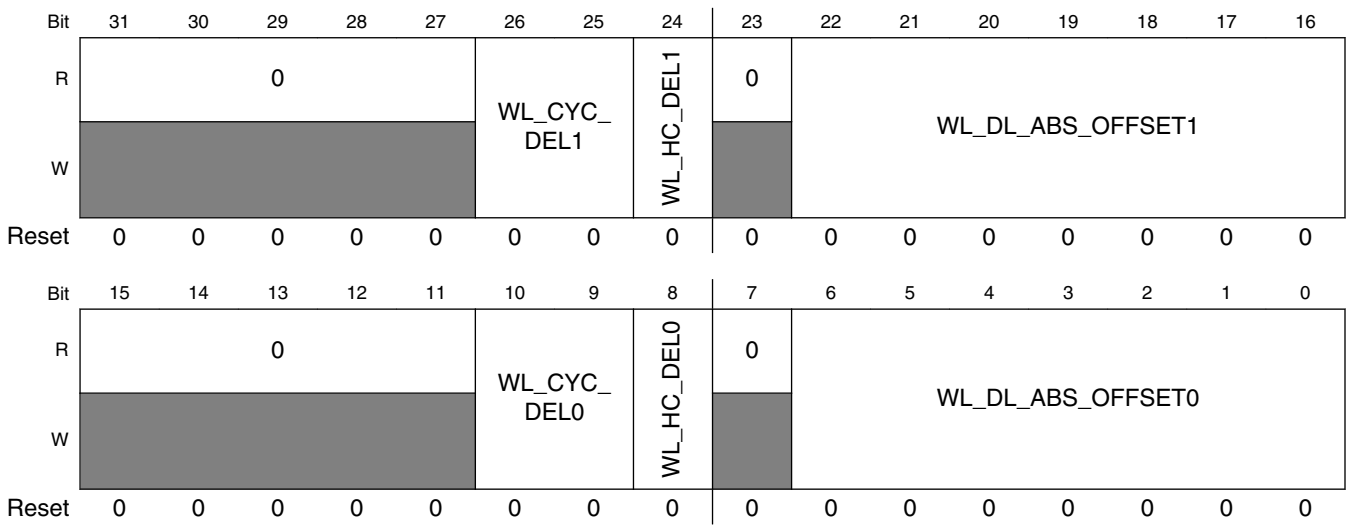
Table continues on the next page...

MMDC_MPWLGCR field descriptions (continued)

Field	Description
	0 No error was found on byte0 during write-leveling HW calibration. 1 An error was found on byte0 during write-leveling HW calibration.
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5 WL_SW_RES1	Byte1 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ8 during SW write-leveling. 0 DQS1 sampled low CK during SW write-leveling. 1 DQS1 sampled high CK during SW write-leveling.
4 WL_SW_RES0	Byte0 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ0 during SW write-leveling. 0 DQS0 sampled low CK during SW write-leveling. 1 DQS0 sampled high CK during SW write-leveling.
3 -	This field is reserved. Reserved
2 SW_WL_CNT_EN	SW write-leveling count down enable. This bit when asserted set a certain delay of (25+15) cycles from the setting of SW_WL_EN and before driving the DQS to the DDR device. This bit should be asserted before the first SW write-leveling request and after issuing the write leveling MRS command 0 MMDC doesn't count 25+15 cycles before issuing write-leveling DQS. 1 MMDC counts 25+15 cycles before issuing write-leveling DQS.
1 SW_WL_EN	Write-Leveling SW enable. If this bit is asserted then the MMDC will perform one write-leveling iteration with the DDR device (assuming that Write-Leveling procedure is already enabled in the DDR device through MRS command). HW negate this bit upon completion of the SW write-leveling. Negation of this bit also points that the write-leveling SW calibration result is valid **NOTE:** If this bit and the SW_WL_CNT_EN are enabled the MMDC counts 25 + 15 cycles before issuing the SW write-leveling DQS.
0 HW_WL_EN	Write-Leveling HW (automatic) enable. If this bit is asserted then the MMDC will perform the whole Write-Leveling sequence with the DDR device (assuming that Write-Leveling procedure is already enabled in the DDR device through MRS command). HW negates this bit upon completion of the HW write-leveling. Negation of this bit also points that the write-leveling HW calibration results are valid **NOTE:** Before issuing the first DQS the MMDC counts 25 + 15 cycles automatically as required by the standard.

33.12.34 MMDC PHY Write Leveling Delay Control Register 0 (MMDC_MPWLDECTRL0)

Address: 21B_0000h base + 80Ch offset = 21B_080Ch



MMDC_MPWLDECTRL0 field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26–25 WL_CYC_DEL1	Write leveling cycle delay for Byte 1. This field indicates whether a delay of 1 or 2 cycles between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_HC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When both SW write-leveling is enabled (i.e. SW_WL_EN = 1) or HW write-leveling is enabled (i.e. HW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_HC_DEL. Note that in HW write-leveling this field is not used for indication, as in WL_DL_OFFSET and WL_HC_DEL, but for configuration. 0 No delay is added. 1 1 cycle delay is added. 2 2 cycles delay is added. 3 Reserved.
24 WL_HC_DEL1	Write leveling half cycle delay for Byte 1. This field indicates whether a delay of half cycle between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_CYC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When SW write-leveling is enabled (i.e. SW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_CYC_DEL. When HW write-leveling is enabled (i.e. HW_WL_EN = 1) then this value will indicate (status) whether a delay of half cycle was added or not to the associated WL_DL_OFFSET and WL_CYC_DEL.

Table continues on the next page...

MMDC_MPWLDECTRL0 field descriptions (continued)

Field	Description
	0 No delay is added. 1 Half cycle delay is added.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 WL_DL_ABS_ OFFSET1	Absolute write-leveling delay offset for Byte 1. This field indicates the absolute delay between CK and write DQS of Byte1 with fractions of a clock period and up to half cycle. This value is process and frequency independent. The value of the delay can be calculated using the following equation (WR_DL_ABS_OFFSET1 / 256) * clock period When SW write-leveling is enabled (i.e. SW_WL_EN = 1) then this value will be taken as is to the associated delay-line. When HW write-leveling is enabled (i.e. HW_WL_EN = 1) then this value will indicate (status) the value that is taken to the associated delay-line at the end of the write-leveling calibration. **NOTE:** The delay-line has a resolution that may vary between device to device, therefore is some cases an increment of the delay by 1 step may be smaller than the delay-line resolution.
15–11 Reserved	This read-only field is reserved and always has the value 0.
10–9 WL_CYC_DEL0	Write leveling cycle delay for Byte 0. This field indicates whether a delay of 1 or 2 cycles between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_HC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When both SW write-leveling is enabled (i.e. SW_WL_EN = 1) or HW write-leveling is enabled (i.e. HW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_HC_DEL. Note that in HW write-leveling this field is not used for indication, as in WL_DL_OFFSET and WL_HC_DEL, but for configuration. 0 No delay is added. 1 1 cycle delay is added. 2 2 cycles delay is added. 3 Reserved.
8 WL_HC_DEL0	Write leveling half cycle delay for Byte 0. This field indicates whether a delay of half cycle between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_CYC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When SW write-leveling is enabled (i.e. SW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_CYC_DEL. When HW write-leveling is enabled (i.e. HW_WL_EN = 1) then this value will indicate (status) whether a delay of half cycle was added or not to the associated WL_DL_OFFSET and WL_CYC_DEL. 0 No delay is added. 1 Half cycle delay is added.
7 Reserved	This read-only field is reserved and always has the value 0.
WL_DL_ABS_ OFFSET0	Absolute write-leveling delay offset for Byte 0. This field indicates the absolute delay between CK and write DQS of Byte0 with fractions of a clock period and up to half cycle. This value is process and frequency independent. The value of the delay can be calculated using the following equation (WR_DL_ABS_OFFSET1 / 256) * clock period

Table continues on the next page...

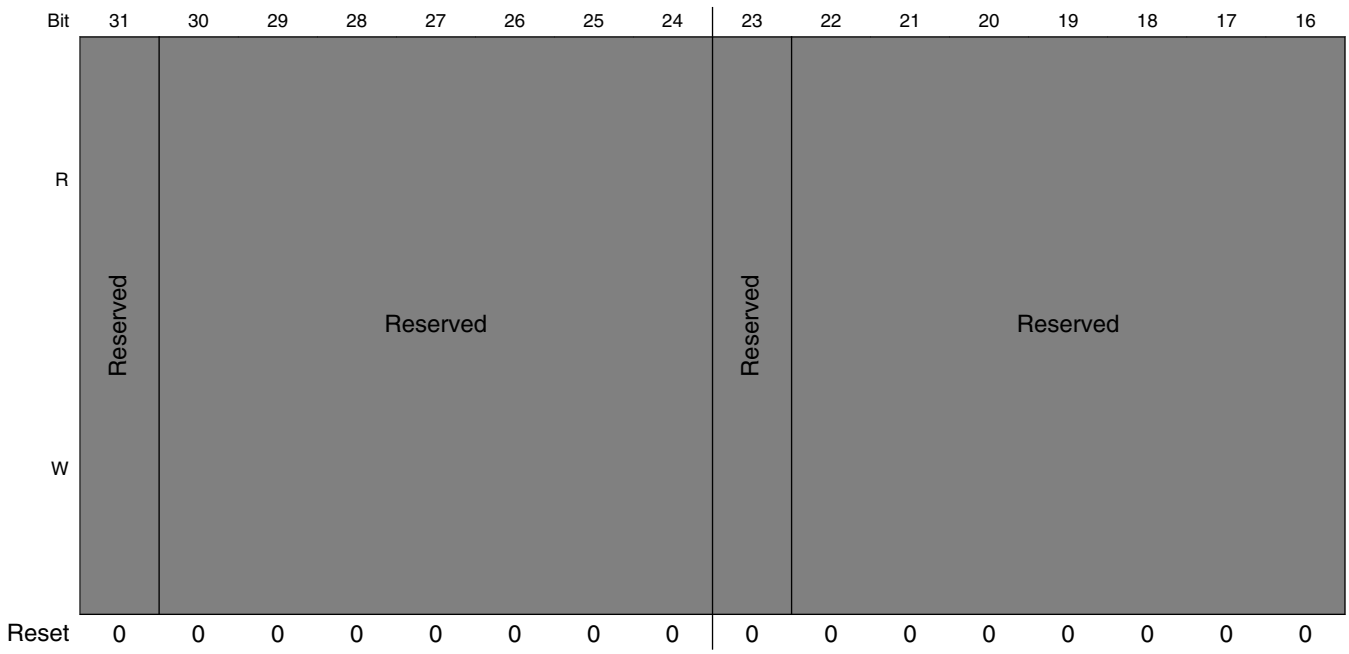
MMDC_MPWLDECTRL0 field descriptions (continued)

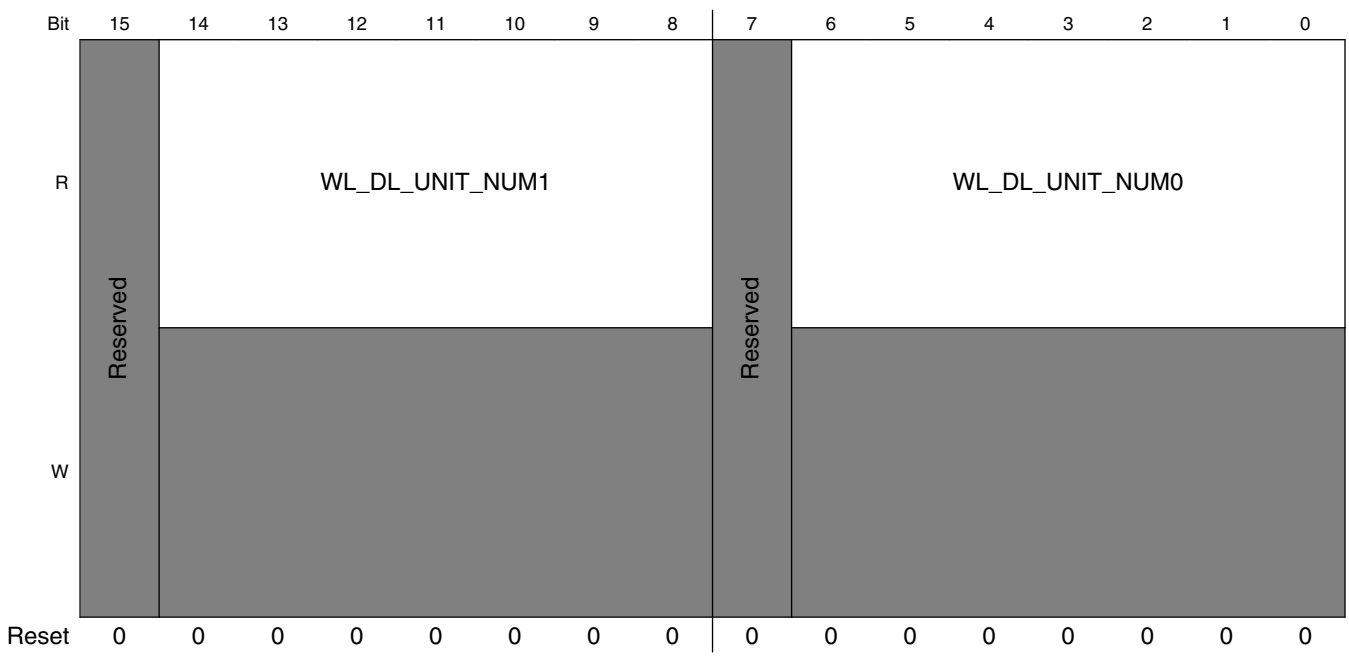
Field	Description
	When SW write-leveling is enabled (i.e. SW_WL_EN = 1) then this value will be taken as is to the associated delay-line. When HW write-leveling is enabled (i.e. HW_WL_EN = 1) then this value will indicate (status) the value that is taken to the associated delay-line at the end of the write-leveling calibration. **NOTE:** The delay-line has a resolution that may vary between device to device, therefore is some cases an increment of the delay by 1 step may be smaller than the delay-line resolution.

33.12.35 MMDC PHY Write Leveling delay-line Status Register (MMDC_MPWLDLST)

This register holds the status of the four write leveling delay-lines.

Address: 21B_0000h base + 814h offset = 21B_0814h





MMDC_MPWLDLST field descriptions

Field	Description
31 -	This field is reserved. Reserved
30–24 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
22–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14–8 WL_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by write leveling delay-line 1.
7 -	This field is reserved. Reserved
WL_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by write leveling delay-line 0.

33.12.36 MMDC PHY ODT control register (MMDC_MPODTCTRL)

NOTE

In LPDDR2 mode this register should be cleared, so no termination will be activated

Address: 21B_0000h base + 818h offset = 21B_0818h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	Reserved			0	ODT1_INT_RES			0	ODT0_INT_RES			ODT_RD_ACT_EN	ODT_RD_PAS_EN	ODT_WR_ACT_EN	ODT_WR_PAS_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPODTCTRL field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value 0.
18–16 -	This field is reserved. Reserved
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 -	This field is reserved. Reserved
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 ODT1_INT_RES	On chip ODT byte1 resistor - This field determines the Rtt_Nom of the on chip ODT byte1 resistor during read accesses. 0000 Rtt_Nom Disabled. 001 Rtt_Nom 120 Ohm 010 Rtt_Nom 60 Ohm 011 Rtt_Nom 40 Ohm 100 Rtt_Nom 30 Ohm 101 Rtt_Nom 24 Ohm 110 Rtt_Nom 20 Ohm 111 Rtt_Nom 17 Ohm
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 ODT0_INT_RES	On chip ODT byte0 resistor - This field determines the Rtt_Nom of the on chip ODT byte0 resistor during read accesses. 000 Rtt_Nom Disabled. 001 Rtt_Nom 120 Ohm 010 Rtt_Nom 60 Ohm 011 Rtt_Nom 40 Ohm 100 Rtt_Nom 30 Ohm 101 Rtt_Nom 24 Ohm 110 Rtt_Nom 20 Ohm 111 Rtt_Nom 17 Ohm
3 ODT_RD_ACT_EN	Active read CS ODT enable. The bit determines if ODT pin of the active CS will be asserted during read accesses. 0 Active CS ODT pin is disabled during read access. 1 Active CS ODT pin is enabled during read access.
2 ODT_RD_PAS_EN	Inactive read CS ODT enable. The bit determines if ODT pin of the inactive CS will be asserted during read accesses. 0 Inactive CS ODT pin is disabled during read accesses to other CS. 1 Inactive CS ODT pin is enabled during read accesses to other CS.
1 ODT_WR_ACT_EN	Active write CS ODT enable. The bit determines if ODT pin of the active CS will be asserted during write accesses.

Table continues on the next page...

MMDC_MPODTCTRL field descriptions (continued)

Field	Description
	0 Active CS ODT pin is disabled during write access. 1 Active CS ODT pin is enabled during write access.
0 ODT_WR_PAS_EN	Inactive write CS ODT enable. The bit determines if ODT pin of the inactive CS will be asserted during write accesses. 0 Inactive CS ODT pin is disabled during write accesses to other CS. 1 Inactive CS ODT pin is enabled during write accesses to other CS.

33.12.37 MMDC PHY Read DQ Byte0 Delay Register (MMDC_MPRDDQBY0DL)

This register is used to add fine-tuning adjustment to every bit in the read DQ byte0 relative to the read DQS. This delay is in addition to the read data calibration. If operating in 64-bit mode, there is an identical register that is mapped at the second base address.

Address: 21B_0000h base + 81Ch offset = 21B_081Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	rd_dq7_del			0	rd_dq6_del			0	rd_dq5_del			0	rd_dq4_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	rd_dq3_del			0	rd_dq2_del			0	rd_dq1_del			0	rd_dq0_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPRDDQBY0DL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 rd_dq7_del	Read dqs0 to dq7 delay fine-tuning. This field holds the number of delay units that are added to dq7 relative to dqs0. 000 No change in dq7 delay 001 Add dq7 delay of 1 delay unit 010 Add dq7 delay of 2 delay units. 011 Add dq7 delay of 3 delay units. 100 Add dq7 delay of 4 delay units. 101 Add dq7 delay of 5 delay units. 110 Add dq7 delay of 6 delay units. 111 Add dq7 delay of 7 delay units.

Table continues on the next page...

MMDC_MPRDDQBY0DL field descriptions (continued)

Field	Description
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 rd_dq6_del	Read dqs0 to dq6 delay fine-tuning. This field holds the number of delay units that are added to dq6 relative to dqs0. 000 No change in dq6 delay 001 Add dq6 delay of 1 delay unit 010 Add dq6 delay of 2 delay units. 011 Add dq6 delay of 3 delay units. 100 Add dq6 delay of 4 delay units. 101 Add dq6 delay of 5 delay units. 110 Add dq6 delay of 6 delay units. 111 Add dq6 delay of 7 delay units.
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 rd_dq5_del	Read dqs0 to dq5 delay fine-tuning. This field holds the number of delay units that are added to dq5 relative to dqs0. 000 No change in dq5 delay 001 Add dq5 delay of 1 delay unit 010 Add dq5 delay of 2 delay units. 011 Add dq5 delay of 3 delay units. 100 Add dq5 delay of 4 delay units. 101 Add dq5 delay of 5 delay units. 110 Add dq5 delay of 6 delay units. 111 Add dq5 delay of 7 delay units.
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 rd_dq4_del	Read dqs0 to dq4 delay fine-tuning. This field holds the number of delay units that are added to dq4 relative to dqs0. 000 No change in dq4 delay 001 Add dq4 delay of 1 delay unit 010 Add dq4 delay of 2 delay units. 011 Add dq4 delay of 3 delay units. 100 Add dq4 delay of 4 delay units. 101 Add dq4 delay of 5 delay units. 110 Add dq4 delay of 6 delay units. 111 Add dq4 delay of 7 delay units.
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 rd_dq3_del	Read dqs0 to dq3 delay fine-tuning. This field holds the number of delay units that are added to dq3 relative to dqs0. 000 No change in dq3 delay 001 Add dq3 delay of 1 delay unit 010 Add dq3 delay of 2 delay units. 011 Add dq3 delay of 3 delay units.

Table continues on the next page...

MMDC_MPRDDQBY0DL field descriptions (continued)

Field	Description
	100 Add dq3 delay of 4 delay units. 101 Add dq3 delay of 5 delay units. 110 Add dq3 delay of 6 delay units. 111 Add dq3 delay of 7 delay units.
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 rd_dq2_del	Read dqs0 to dq2 delay fine-tuning. This field holds the number of delay units that are added to dq2 relative to dqs0. 000 No change in dq2 delay 001 Add dq2 delay of 1 delay unit 010 Add dq2 delay of 2 delay units. 011 Add dq2 delay of 3 delay units. 100 Add dq2 delay of 4 delay units. 101 Add dq2 delay of 5 delay units. 110 Add dq2 delay of 6 delay units. 111 Add dq2 delay of 7 delay units.
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 rd_dq1_del	Read dqs0 to dq1 delay fine-tuning. This field holds the number of delay units that are added to dq1 relative to dqs0. 000 No change in dq1 delay 001 Add dq1 delay of 1 delay unit 010 Add dq1 delay of 2 delay units. 011 Add dq1 delay of 3 delay units. 100 Add dq1 delay of 4 delay units. 101 Add dq1 delay of 5 delay units. 110 Add dq1 delay of 6 delay units. 111 Add dq1 delay of 7 delay units.
3 Reserved	This read-only field is reserved and always has the value 0.
rd_dq0_del	Read dqs0 to dq0 delay fine-tuning. This field holds the number of delay units that are added to dq0 relative to dqs0. 000 No change in dq0 delay 001 Add dq0 delay of 1 delay unit 010 Add dq0 delay of 2 delay units. 011 Add dq0 delay of 3 delay units. 100 Add dq0 delay of 4 delay units. 101 Add dq0 delay of 5 delay units. 110 Add dq0 delay of 6 delay units. 111 Add dq0 delay of 7 delay units.

33.12.38 MMDC PHY Read DQ Byte1 Delay Register (MMDC_MPRDDQBY1DL)

This register is used to add fine-tuning adjustment to every bit in the read DQ byte1 relative to the read DQS

Address: 21B_0000h base + 820h offset = 21B_0820h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	rd_dq15_del			0	rd_dq14_del			0	rd_dq13_del			0	rd_dq12_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	rd_dq11_del			0	rd_dq10_del			0	rd_dq9_del			0	rd_dq8_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPRDDQBY1DL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 rd_dq15_del	Read dqs1 to dq15 delay fine-tuning. This field holds the number of delay units that are added to dq15 relative to dqs1. 000 No change in dq15 delay 001 Add dq15 delay of 1 delay unit 010 Add dq15 delay of 2 delay units. 011 Add dq15 delay of 3 delay units. 100 Add dq15 delay of 4 delay units. 101 Add dq15 delay of 5 delay units. 110 Add dq15 delay of 6 delay units. 111 Add dq15 delay of 7 delay units.
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 rd_dq14_del	Read dqs1 to dq14 delay fine-tuning. This field holds the number of delay units that are added to dq14 relative to dqs1. 000 No change in dq14 delay 001 Add dq14 delay of 1 delay unit 010 Add dq14 delay of 2 delay units. 011 Add dq14 delay of 3 delay units. 100 Add dq14 delay of 4 delay units. 101 Add dq14 delay of 5 delay units. 110 Add dq14 delay of 6 delay units. 111 Add dq14 delay of 7 delay units.

Table continues on the next page...

MMDC_MPRDDQBY1DL field descriptions (continued)

Field	Description
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 rd_dq13_del	Read dqs1 to dq13 delay fine-tuning. This field holds the number of delay units that are added to dq13 relative to dqs1. 000 No change in dq13 delay 001 Add dq13 delay of 1 delay unit 010 Add dq13 delay of 2 delay units. 011 Add dq13 delay of 3 delay units. 100 Add dq13 delay of 4 delay units. 101 Add dq13 delay of 5 delay units. 110 Add dq13 delay of 6 delay units. 111 Add dq13 delay of 7 delay units.
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 rd_dq12_del	Read dqs1 to dq12 delay fine-tuning. This field holds the number of delay units that are added to dq12 relative to dqs1. 000 No change in dq12 delay 001 Add dq12 delay of 1 delay unit 010 Add dq12 delay of 2 delay units. 011 Add dq12 delay of 3 delay units. 100 Add dq12 delay of 4 delay units. 101 Add dq12 delay of 5 delay units. 110 Add dq12 delay of 6 delay units. 111 Add dq12 delay of 7 delay units.
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 rd_dq11_del	Read dqs1 to dq11 delay fine-tuning. This field holds the number of delay units that are added to dq11 relative to dqs1. 000 No change in dq11 delay 001 Add dq11 delay of 1 delay unit 010 Add dq11 delay of 2 delay units. 011 Add dq11 delay of 3 delay units. 100 Add dq11 delay of 4 delay units. 101 Add dq11 delay of 5 delay units. 110 Add dq11 delay of 6 delay units. 111 Add dq11 delay of 7 delay units.
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 rd_dq10_del	Read dqs1 to dq10 delay fine-tuning. This field holds the number of delay units that are added to dq10 relative to dqs1. 000 No change in dq10 delay 001 Add dq10 delay of 1 delay unit 010 Add dq10 delay of 2 delay units. 011 Add dq10 delay of 3 delay units.

Table continues on the next page...

MMDC_MPRDDQBY1DL field descriptions (continued)

Field	Description
	100 Add dq10 delay of 4 delay units. 101 Add dq10 delay of 5 delay unit 110 Add dq10 delay of 6 delay units. 111 Add dq10 delay of 7 delay units.
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 rd_dq9_del	Read dqs1 to dq9 delay fine-tuning. This field holds the number of delay units that are added to dq9 relative to dqs1. 000 No change in dq9 delay 001 Add dq9 delay of 1 delay unit 010 Add dq9 delay of 2 delay units. 011 Add dq9 delay of 3 delay units. 100 Add dq9 delay of 4 delay units. 101 Add dq9 delay of 5 delay units. 110 Add dq9 delay of 6 delay units. 111 Add dq9 delay of 7 delay units.
3 Reserved	This read-only field is reserved and always has the value 0.
rd_dq8_del	Read dqs1 to dq8 delay fine-tuning. This field holds the number of delay units that are added to dq8 relative to dqs1. 000 No change in dq8 delay 001 Add dq8 delay of 1 delay unit 010 Add dq8 delay of 2 delay units. 011 Add dq8 delay of 3 delay units. 100 Add dq8 delay of 4 delay units. 101 Add dq8 delay of 5 delay units. 110 Add dq8 delay of 6 delay units. 111 Add dq8 delay of 7 delay units.

33.12.39 MMDC PHY Write DQ Byte0 Delay Register (MMDC_MPWRDQBY0DL)

This register is used to add fine-tuning adjustment to every bit in the write DQ byte0 relative to the write DQS

Address: 21B_0000h base + 82Ch offset = 21B_082Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	wr_dm0_del		wr_dq7_del		0		wr_dq6_del		0		wr_dq5_del		0		wr_dq4_del	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	0		wr_dq3_del		0		wr_dq2_del		0		wr_dq1_del		0		wr_dq0_del	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPWRDQBY0DL field descriptions

Field	Description
31–30 wr_dm0_del	Write dm0 delay fine-tuning. This field holds the number of delay units that are added to dm0 relative to dqs0. 00 No change in dm0 delay 01 Add dm0 delay of 1 delay unit. 10 Add dm0 delay of 2 delay units. 11 Add dm0 delay of 3 delay units.
29–28 wr_dq7_del	Write dq7 delay fine-tuning. This field holds the number of delay units that are added to dq7 relative to dqs0. 00 No change in dq7 delay 01 Add dq7 delay of 1 delay unit. 10 Add dq7 delay of 2 delay units. 11 Add dq7 delay of 3 delay units.
27–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 wr_dq6_del	Write dq6 delay fine-tuning. This field holds the number of delay units that are added to dq6 relative to dqs0. 00 No change in dq6 delay 01 Add dq6 delay of 1 delay unit. 10 Add dq6 delay of 2 delay units. 11 Add dq6 delay of 3 delay units.
23–22 Reserved	This read-only field is reserved and always has the value 0.
21–20 wr_dq5_del	Write dq5 delay fine-tuning. This field holds the number of delay units that are added to dq5 relative to dqs0.

Table continues on the next page...

MMDC_MPWRDQBY0DL field descriptions (continued)

Field	Description
	00 No change in dq5 delay 01 Add dq5 delay of 1 delay unit. 10 Add dq5 delay of 2 delay units. 11 Add dq5 delay of 3 delay units.
19–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 wr_dq4_del	Write dq4 delay fine-tuning. This field holds the number of delay units that are added to dq4 relative to dqs0. 00 No change in dq4 delay 01 Add dq4 delay of 1 delay unit.. 10 Add dq4 delay of 2 delay units. 11 Add dq4 delay of 3 delay units.
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–12 wr_dq3_del	Write dq3 delay fine-tuning. This field holds the number of delay units that are added to dq3 relative to dqs0. 00 No change in dq3 delay 01 Add dq3 delay of 1 delay unit. 10 Add dq3 delay of 2 delay units. 11 Add dq3 delay of 3 delay units.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 wr_dq2_del	Write dq2 delay fine-tuning. This field holds the number of delay units that are added to dq2 relative to dqs0. 00 No change in dq2 delay 01 Add dq2 delay of 1 delay unit. 10 Add dq2 delay of 2 delay units. 11 Add dq2 delay of 3 delay units.
7–6 Reserved	This read-only field is reserved and always has the value 0.
5–4 wr_dq1_del	Write dq1 delay fine-tuning. This field holds the number of delay units that are added to dq1 relative to dqs0. 00 No change in dq1 delay 01 Add dq1 delay of 1 delay unit. 10 Add dq1 delay of 2 delay units. 11 Add dq1 delay of 3 delay units.
3–2 Reserved	This read-only field is reserved and always has the value 0.
wr_dq0_del	Write dq0 delay fine-tuning. This field holds the number of delay units that are added to dq0 relative to dqs0. 00 No change in dq0 delay 01 Add dq0 delay of 1 delay unit.

Table continues on the next page...

MMDC_MPWRDQBY0DL field descriptions (continued)

Field	Description
	10 Add dq0 delay of 2 delay units. 11 Add dq0 delay of 3 delay units.

33.12.40 MMDC PHY Write DQ Byte1 Delay Register (MMDC_MPWRDQBY1DL)

This register is used to add fine-tuning adjustment to every bit in the write DQ byte1 relative to the write DQS

Address: 21B_0000h base + 830h offset = 21B_0830h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	wr_dm1_del		wr_dq15_del		0		wr_dq14_del		0		wr_dq13_del		0		wr_dq12_del	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	0		wr_dq11_del		0		wr_dq10_del		0		wr_dq9_del		0		wr_dq8_del	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPWRDQBY1DL field descriptions

Field	Description
31–30 wr_dm1_del	Write dm1 delay fine-tuning. This field holds the number of delay units that are added to dm1 relative to dqs1. 00 No change in dm1 delay 01 Add dm1 delay of 1 delay unit. 10 Add dm1 delay of 2 delay units. 11 Add dm1 delay of 3 delay units.
29–28 wr_dq15_del	Write dq15 delay fine-tuning. This field holds the number of delay units that are added to dq15 relative to dqs1. 00 No change in dq15 delay 01 Add dq15 delay of 1 delay unit. 10 Add dq15 delay of 2 delay units. 11 Add dq15 delay of 3 delay units.
27–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 wr_dq14_del	Write dq14 delay fine-tuning. This field holds the number of delay units that are added to dq14 relative to dqs1. 00 No change in dq14 delay

Table continues on the next page...

MMDC_MPWRDQBY1DL field descriptions (continued)

Field	Description
	01 Add dq14 delay of 1 delay unit. 10 Add dq14 delay of 2 delay units. 11 Add dq14 delay of 3 delay units.
23–22 Reserved	This read-only field is reserved and always has the value 0.
21–20 wr_dq13_del	Write dq13 delay fine-tuning. This field holds the number of delay units that are added to dq13 relative to dqs1. 00 No change in dq13 delay 01 Add dq13 delay of 1 delay unit. 10 Add dq13 delay of 2 delay units. 11 Add dq13 delay of 3 delay units.
19–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 wr_dq12_del	Write dq12 delay fine-tuning. This field holds the number of delay units that are added to dq12 relative to dqs1. 00 No change in dq12 delay 01 Add dq12 delay of 1 delay unit. 10 Add dq12 delay of 2 delay units. 11 Add dq12 delay of 3 delay units.
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–12 wr_dq11_del	Write dq11 delay fine-tuning. This field holds the number of delay units that are added to dq11 relative to dqs1. 00 No change in dq11 delay 01 Add dq11 delay of 1 delay unit. 10 Add dq11 delay of 2 delay units. 11 Add dq11 delay of 3 delay units.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 wr_dq10_del	Write dq10 delay fine-tuning. This field holds the number of delay units that are added to dq10 relative to dqs1. 00 No change in dq10 delay 01 Add dq10 delay of 1 delay unit. 10 Add dq10 delay of 2 delay units. 11 Add dq10 delay of 3 delay units.
7–6 Reserved	This read-only field is reserved and always has the value 0.
5–4 wr_dq9_del	Write dq9 delay fine-tuning. This field holds the number of delay units that are added to dq9 relative to dqs1. 00 No change in dq9 delay 01 Add dq9 delay of 1 delay unit. 10 Add dq9 delay of 2 delay units. 11 Add dq9 delay of 3 delay units.

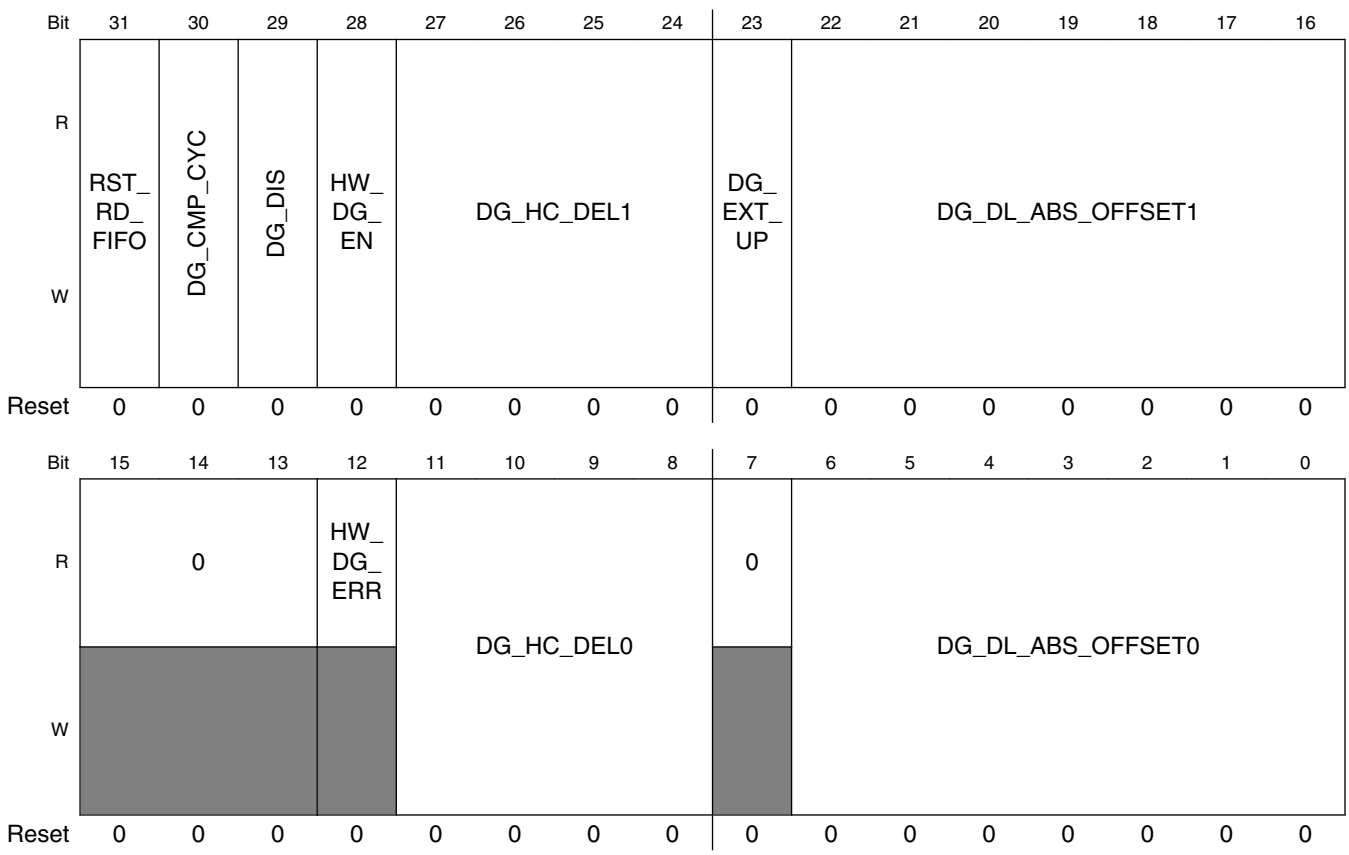
Table continues on the next page...

MMDC_MPWRDQBY1DL field descriptions (continued)

Field	Description
3–2 Reserved	This read-only field is reserved and always has the value 0.
wr_dq8_del	Write dq8 delay fine-tuning. This field holds the number of delay units that are added to dq8 relative to dqs1. 00 No change in dq8 delay 01 Add dq8 delay of 1 delay unit. 10 Add dq8 delay of 2 delay units. 11 Add dq8 delay of 3 delay units.

33.12.41 MMDC PHY Read DQS Gating Control Register 0 (MMDC_MPDGCTRL0)

Address: 21B_0000h base + 83Ch offset = 21B_083Ch



MMDC_MPDGCTRL0 field descriptions

Field	Description
31 RST_RD_FIFO	Reset Read Data FIFO and associated pointers. If this bit is asserted then the MMDC resets the read data FIFO and the associated pointers. This bit is self cleared after the FIFO reset is done.

Table continues on the next page...

MMDC_MPDGCTRL0 field descriptions (continued)

Field	Description
30 DG_CMP_CYC	Read DQS gating sample cycle. If this bit is asserted then the MMDC waits 32 cycles before comparing the read data, Otherwise it waits 16 DDR cycles. 0 MMDC waits 16 DDR cycles 1 MMDC waits 32 DDR cycles
29 DG_DIS	Read DQS gating disable. If this bit is asserted then the MMDC disables the read DQS gating mechanism. If this bits is asserted (read DQS gating is disabled) then pulll-up and pull-down resistors suppose to be used on DQS and DQS# respectively 0 Read DQS gating mechanism is enabled 1 Read DQS gating mechanism is disabled
28 HW_DG_EN	Enable automatic read DQS gating calibration. If this bit is asserted then the MMDC performs automatic read DQS gating calibration. HW negates this bit upon completion of the automatic read DQS gating. Note: Before issuing the first read command the MMDC counts 12 cycles. In LPDDR2 mode automatic (HW) read DQS gating should be disabled and Pull-up/pull-down resistors on DQS/DQS# should be enabled while ODT resistors must be disconnected. 0 Disable automatic read DQS gating calibration 1 Start automatic read DQS gating calibration
27–24 DG_HC_DEL1	Read DQS gating half cycles delay for Byte1 . This field indicates the delay in half cycles between read DQS gate and the middle of the read DQS preamble of Byte1. This delay is added to the delay that is generated by the read DQS1 gating delay-line, So the total read DQS gating delay is (DG_HC_DEL#)*0.5*cycle + (DG_DL_ABS_OFFSET#)*1/256*cycle Upon completion of the automatic read DQS gating calibration this field gets the value of the 4 MSB of ((HW_DG_LOW1 + HW_DG_UP1) /2). 0000 0 cycles delay. 0001 Half cycle delay. 0010 1 cycle delay 1101 6.5 cycles delay 1110 Reserved 1111 Reserved
23 DG_EXT_UP	DG extend upper boundary. By default the upper boundary of DQS gating HW calibration is set according to first failing comparison after at least one passing comparison. If this bit is asserted then the upper boundary is set according to the last passing comparison.
22–16 DG_DL_ABS_ OFFSET1	Absolute read DQS gating delay offset for Byte1. This field indicates the absolute delay between read DQS gate and the middle of the read DQS preamble of Byte1 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (DG_DL_ABS_OFFSET1 / 256)* MMDC AXI clock (fast clock). This field can also bit written by HW. Upon completion of the automatic read DQS gating calibration this field gets the value of the 7 LSB of ((HW_DG_LOW1 + HW_DG_UP1) /2). Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
15–13 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

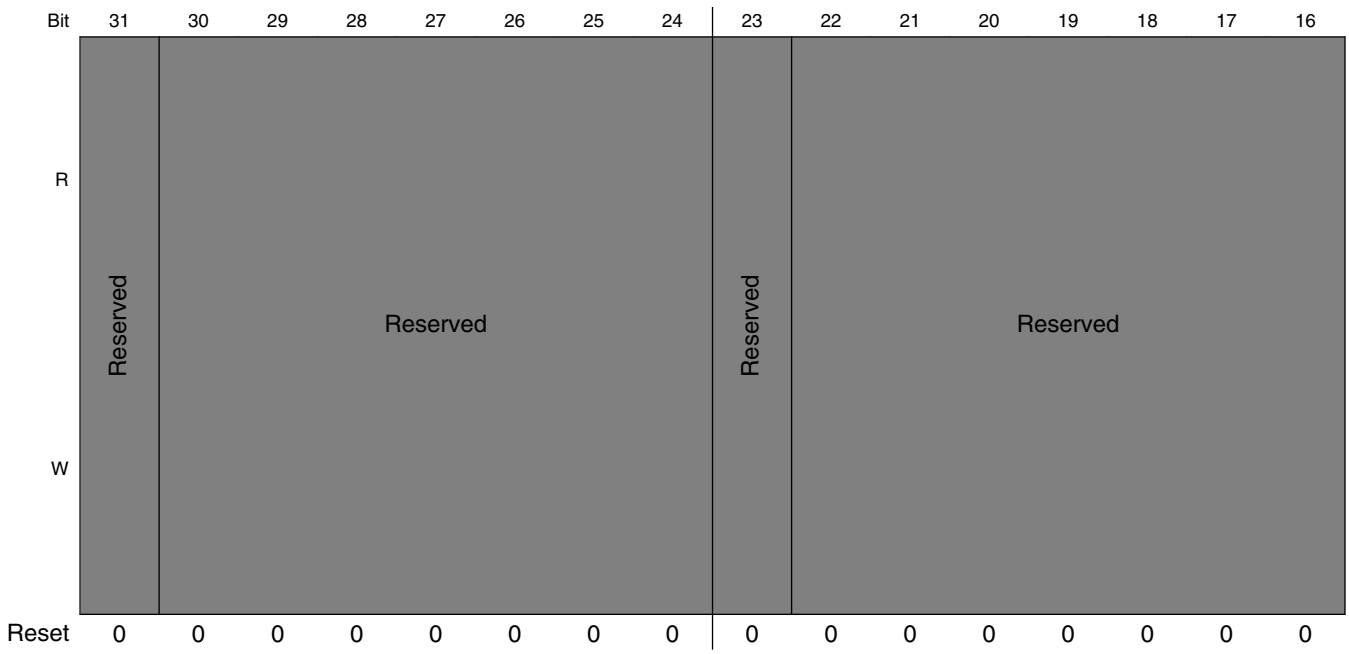
MMDC_MPDGCTRL0 field descriptions (continued)

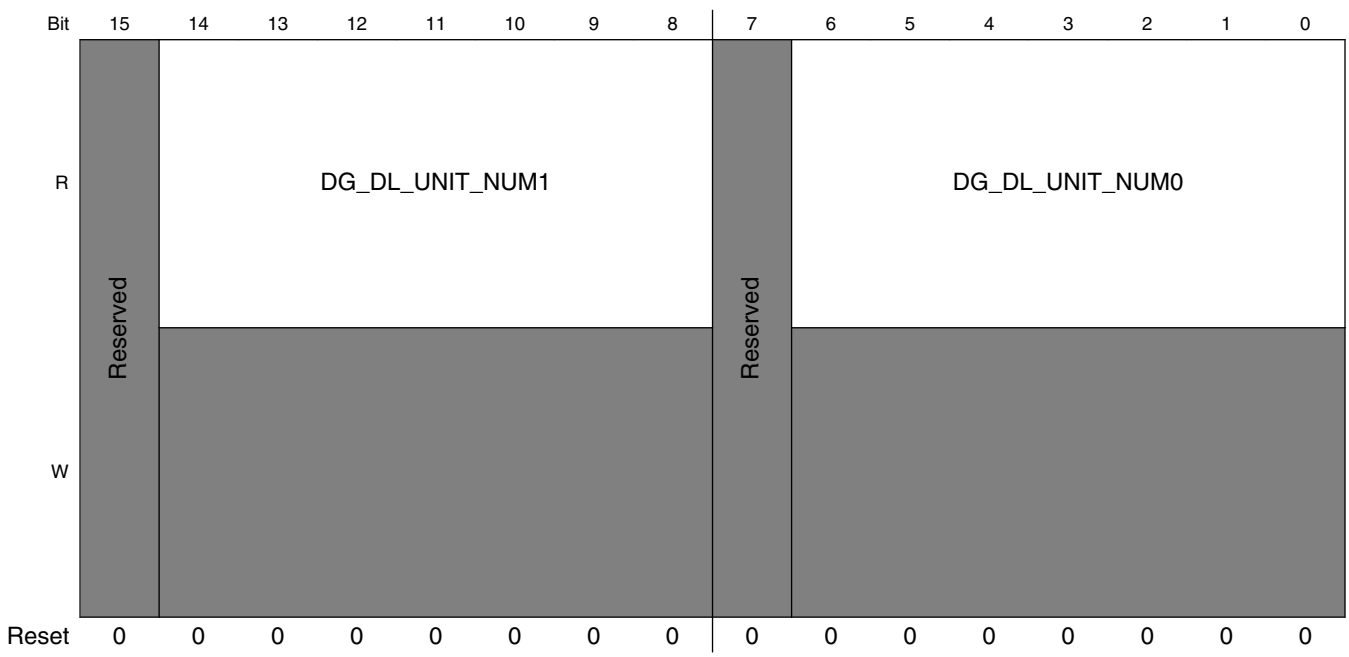
Field	Description
12 HW_DG_ERR	HW DQS gating error. This bit valid is asserted when an error was found during the read DQS gating HW calibration process. Error can occur when no valid value was found during HW calibration. This bit is valid only after HW_DG_EN is de-asserted. 0 No error was found during the DQS gating HW calibration process. 1 An error was found during the DQS gating HW calibration process.
11–8 DG_HC_DEL0	Read DQS gating half cycles delay for Byte0 . This field indicates the delay in half cycles between read DQS gate and the middle of the read DQS preamble of Byte0/4. This delay is added to the delay that is generated by the read DQS1 gating delay-line, So the total read DQS gating delay is (DG_HC_DEL#)*0.5*cycle + (DG_DL_ABS_OFFSET#)*1/256*cycle Upon completion of the automatic read DQS gating calibration this field gets the value of the 4 MSB of ((HW_DG_LOW1 + HW_DG_UP1) /2). 0000 0 cycles delay. 0001 Half cycle delay. 0010 1 cycle delay 1101 6.5 cycles delay 1110 Reserved 1111 Reserved
7 Reserved	This read-only field is reserved and always has the value 0.
DG_DL_ABS_ OFFSET0	Absolute read DQS gating delay offset for Byte0. This field indicates the absolute delay between read DQS gate and the middle of the read DQS preamble of Byte0 with fractions of a clock period and up to half cycle.The fraction is process and frequency independent. The delay of the delay-line would be (DG_DL_ABS_OFFSET0 / 256)* MMDC AXI clock (fast clock). This field can also bit written by HW. Upon completion of the automatic read DQS gating calibration this field gets the value of the 7 LSB of ((HW_DG_LOW0 + HW_DG_UP0) /2). Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.

33.12.42 MMDC PHY Read DQS Gating delay-line Status Register (MMDC_MPDGDLST0)

This register holds the status of the 4 dqs gating delay-lines.

Address: 21B_0000h base + 844h offset = 21B_0844h





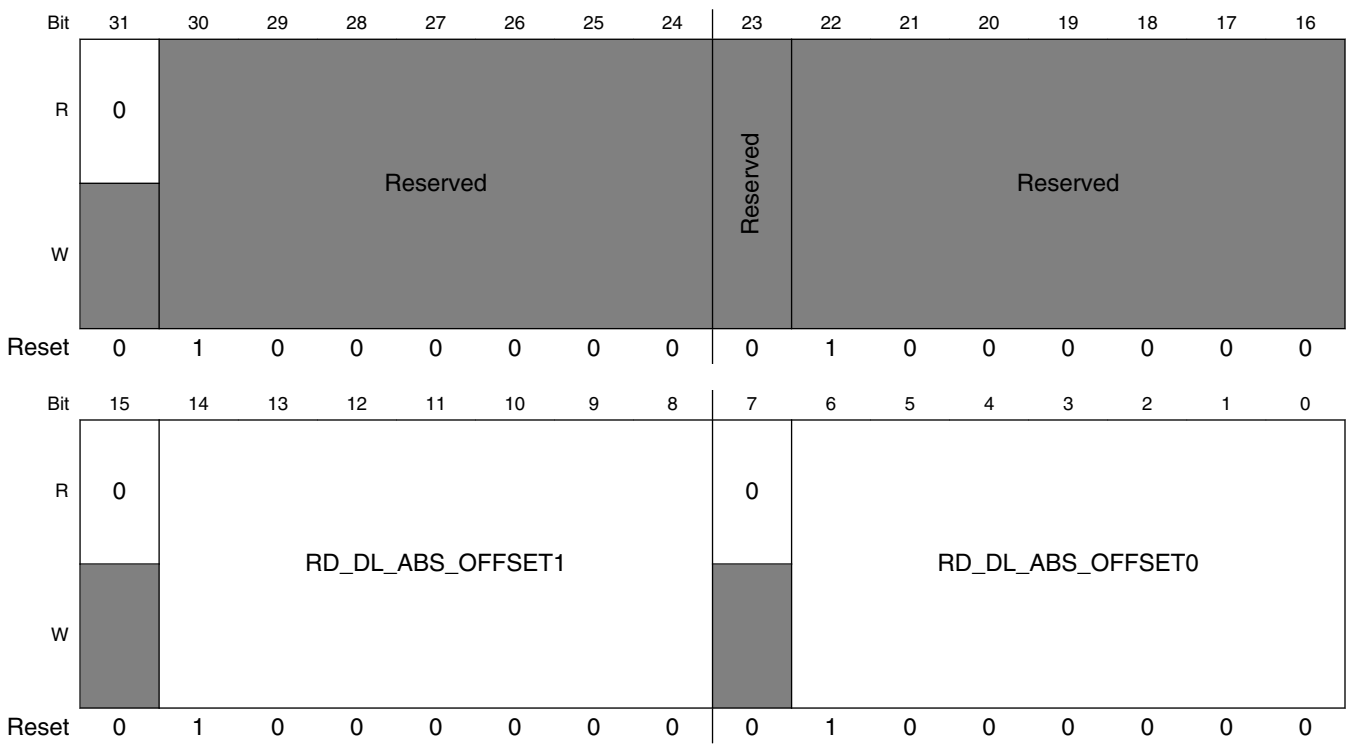
MMDC_MPDGDLST0 field descriptions

Field	Description
31 -	This field is reserved. Reserved
30–24 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
22–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14–8 DG_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by read DQS gating delay-line 1.
7 -	This field is reserved. Reserved
DG_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by read DQS gating delay-line 0.

33.12.43 MMDC PHY Read delay-lines Configuration Register (MMDC_MPRDDLCTL)

This register controls read delay-lines functionality; it determines DQS delay relative to the associated DQ read access. The delay-line compensates for process variations and produces a constant delay regardless of the process, temperature and voltage.

Address: 21B_0000h base + 848h offset = 21B_0848h



MMDC_MPRDDLCTL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
22–16 -	This field is reserved. Reserved
15 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MPRDDLCTL field descriptions (continued)

Field	Description
14–8 RD_DL_ABS_ OFFSET1	Absolute read delay offset for Byte1. This field indicates the absolute delay between read DQS strobe and the read data of Byte1 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (RD_DL_ABS_OFFSET1 / 256) * MMDC AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the read delay-line HW calibration this field gets the value of (HW_RD_DL_LOW1 + HW_RD_DL_UP1) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
7 Reserved	This read-only field is reserved and always has the value 0.
RD_DL_ABS_ OFFSET0	Absolute read delay offset for Byte0. This field indicates the absolute delay between read DQS strobe and the read data of Byte0 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (RD_DL_ABS_OFFSET0 / 256) * MMDC AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the read delay-line HW calibration this field gets the value of (HW_RD_DL_LOW0 + HW_RD_DL_UP0) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.

33.12.44 MMDC PHY Read delay-lines Status Register (MMDC_MPRDDLST)

This register holds the status of the 4 read delay-lines.

Address: 21B_0000h base + 84Ch offset = 21B_084Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	Reserved							Reserved	Reserved						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RD_DL_UNIT_NUM1							0	RD_DL_UNIT_NUM0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

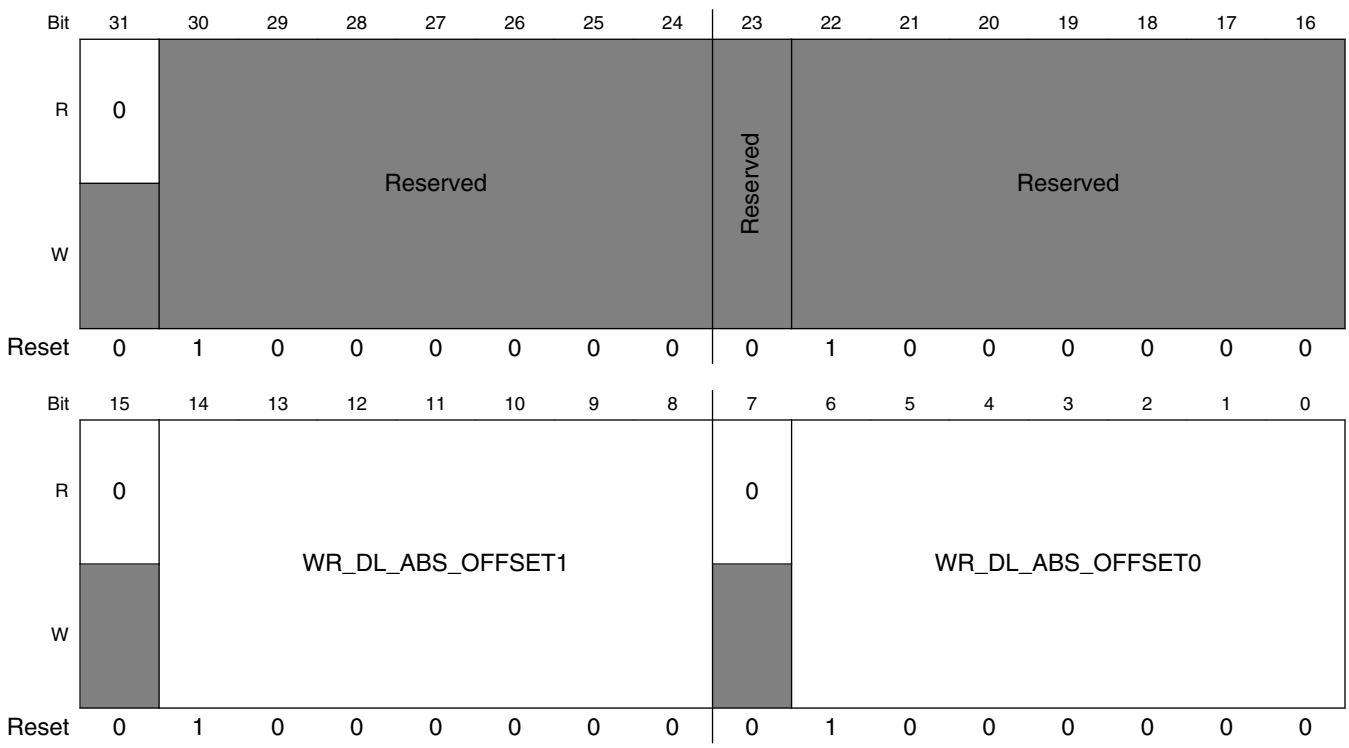
MMDC_MPRDDLST field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
22–16 -	This field is reserved. Reserved
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 RD_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by read delay-line 1.
7 Reserved	This read-only field is reserved and always has the value 0.
RD_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by read delay-line 0.

33.12.45 MMDC PHY Write delay-lines Configuration Register (MMDC_MPWRDLCTL)

This register controls write delay-lines functionality, it determines DQ/DM delay relative to the associated DQS in write access. The delay-line compensates for process variations, and produces a constant delay regardless of the process, temperature and voltage.

Address: 21B_0000h base + 850h offset = 21B_0850h



MMDC_MPWRDLCTL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
22–16 -	This field is reserved. Reserved
15 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MPWRDLCTL field descriptions (continued)

Field	Description
14–8 WR_DL_ABS_ OFFSET1	Absolute write delay offset for Byte1. This field indicates the absolute delay between write DQS strobe and the write data of Byte1 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (WR_DL_ABS_OFFSET1 / 256) * MMDC AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the write delay-line HW calibration this field gets the value of (HW_WR_DL_LOW1 + HW_WR_DL_UP1) /2 Note that not all changes of this value will affect the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
7 Reserved	This read-only field is reserved and always has the value 0.
WR_DL_ABS_ OFFSET0	Absolute write delay offset for Byte0. This field indicates the absolute delay between write DQS strobe and the write data of Byte3 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (WR_DL_ABS_OFFSET0 / 256) * MMDC AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the write delay-line HW calibration this field gets the value of (HW_WR_DL_LOW0 + HW_WR_DL_UP0) /2 Note that not all changes of this value will affect the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.

33.12.46 MMDC PHY Write delay-lines Status Register (MMDC_MPWRDLST)

This register holds the status of the 4 write delay-line.

Address: 21B_0000h base + 854h offset = 21B_0854h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	Reserved							Reserved	Reserved						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WR_DL_UNIT_NUM1							0	WR_DL_UNIT_NUM0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPWRDLST field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
22–16 -	This field is reserved. Reserved
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 WR_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by write delay-line 1.
7 Reserved	This read-only field is reserved and always has the value 0.
WR_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by write delay-line 0.

33.12.47 MMDC PHY CK Control Register (MMDC_MPSDCTRL)

This register controls the fine tuning of the primary clock (CK0).

Address: 21B_0000h base + 858h offset = 21B_0858h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				SDCLK1_del		SDclk0_del		0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPSDCTRL field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value 0.
11–10 SDCLK1_del	DDR clock1 delay fine tuning. This field holds the number of delay units that are added to DDR clock1 (CK1). 00 No change in DDR clock delay 01 Add DDR clock delay of 1 delay unit.

Table continues on the next page...

MMDC_MPSDCTRL field descriptions (continued)

Field	Description
	10 Add DDR clock delay of 2 delay units. 11 Add DDR clock delay of 3 delay units.
9–8 SDclk0_del	DDR clock0 delay fine tuning. This field holds the number of delay units that are added to DDR clock (CK0). 00 No change in DDR clock0 delay 01 Add DDR clock0 delay of 1 delay unit. 10 Add DDR clock0 delay of 2 delay units. 11 Add DDR clock0 delay of 3 delay units.
Reserved	This read-only field is reserved and always has the value 0.

33.12.48 MMDC ZQ LPDDR2 HW Control Register (MMDC_MPZQLP2CTL)

This register controls the idle time that takes the LPDDR2 device to perform ZQ calibration

Address: 21B_0000h base + 85Ch offset = 21B_085Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	0	ZQ_LP2_HW_ZQCS							ZQ_LP2_HW_ZQCL							
Reset	0	0	0	0	1	1	0	1	1	0	1	0	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	0								ZQ_LP2_HW_ZQINIT							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1

MMDC_MPZQLP2CTL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 ZQ_LP2_HW_ ZQCS	This register defines the period in cycles that it takes the memory device to perform a long ZQ calibration. This is the period of time that the MMDC has to wait after sending a short ZQ calibration and before sending other commands. This delay will also be used if ZQ reset is sent. 0x0-0x1A Reserved 0x1B 112 cycles (default) 0x1C 116 cycles

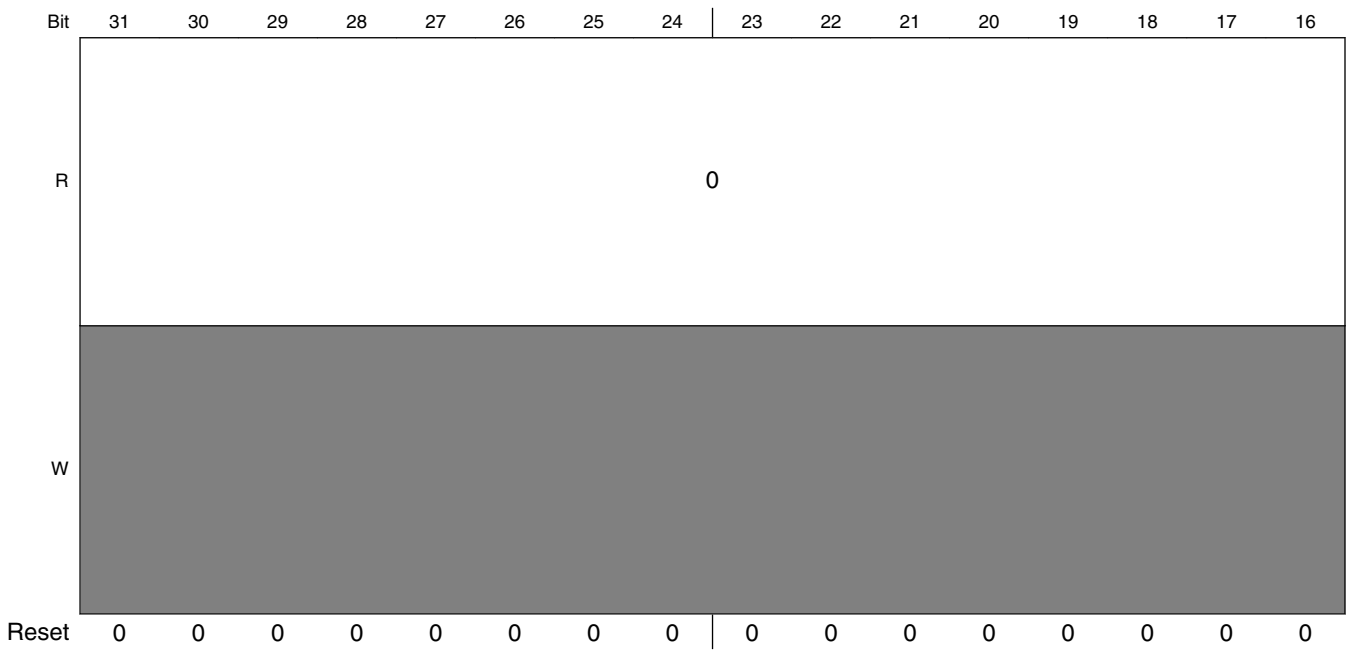
Table continues on the next page...

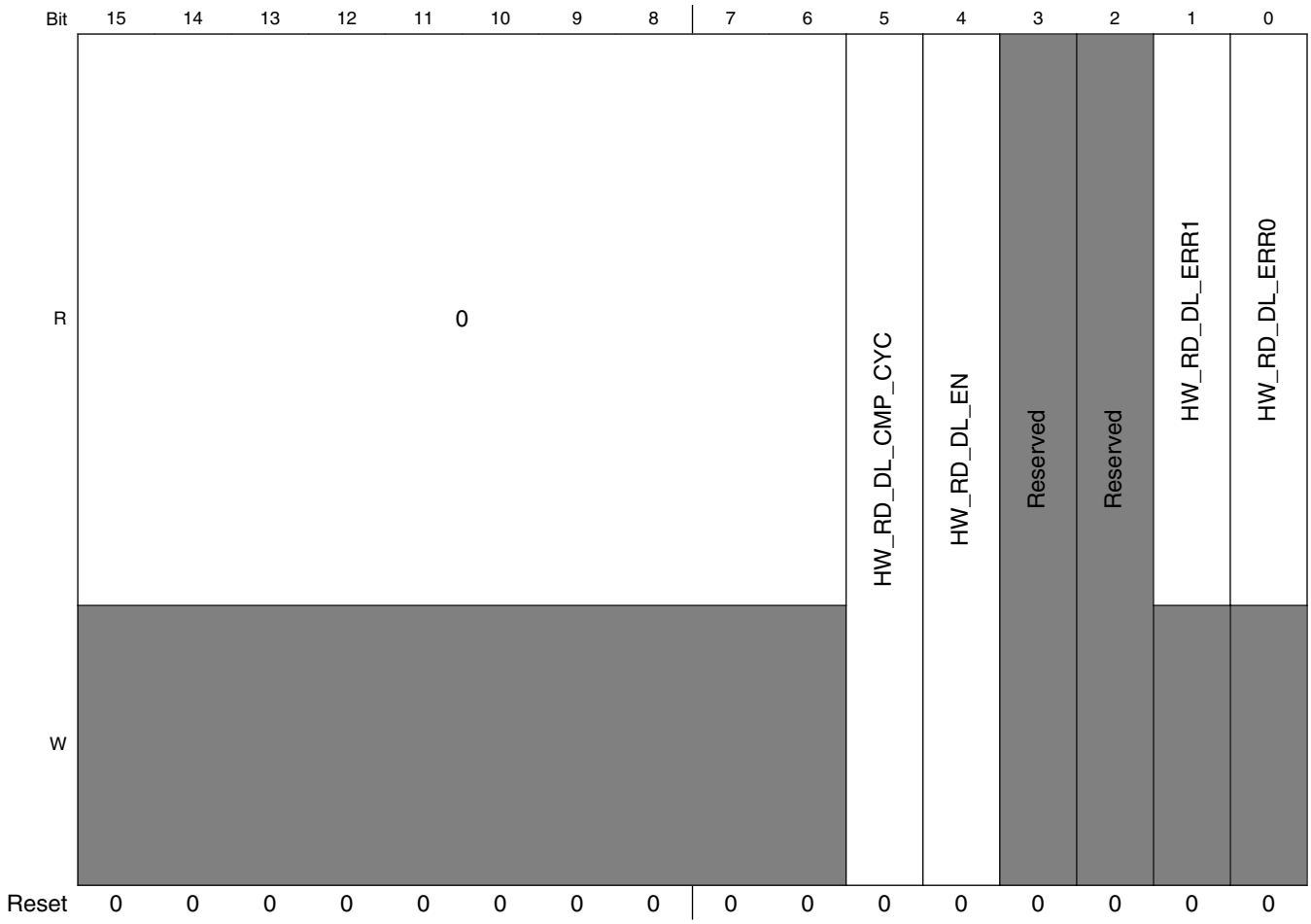
MMDC_MPZQLP2CTL field descriptions (continued)

Field	Description
	0x7E 508 cycles 0x7F 512 cycles
23–16 ZQ_LP2_HW_ZQCL	This register defines the period in cycles that it takes the memory device to perform a short ZQ calibration. This is the period of time that the MMDC has to wait after sending a long ZQ calibration and before sending other commands. 0x0-0x36 Reserved 0x37 112 cycles 0x38 114 cycles 0x5F 192 cycles (Default, JEDEC value, tZQCL, for LPDDR2, 360ns @ clock frequency 533MHz) 0xFE 510 cycles 0xFF 512 cycles
15–9 Reserved	This read-only field is reserved and always has the value 0.
ZQ_LP2_HW_ZQINIT	This register defines the period in cycles that it takes the memory device to perform a Init ZQ calibration. This is the period of time that the MMDC has to wait after sending a init ZQ calibration and before sending other commands. 0x0-0x36 Reserved 0x37 112 cycles 0x38 114 cycles 0x109 532 cycles (Default, JEDEC value, tZQINIT, for LPDDR2, 1us @ clock frequency 533MHz) 0x1FE 1022 cycles 0x1FF 1024 cycles

33.12.49 MMDC PHY Read Delay HW Calibration Control Register (MMDC_MPRDDLHWCTL)

Address: 21B_0000h base + 860h offset = 21B_0860h





MMDC_MPRDDLHWCTL field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 HW_RD_DL_ CMP_CYC	Automatic (HW) read sample cycle. If this bit is asserted then the MMDC will compare the read data 32 cycles after the MMDC sent the read command enable pulse else it compares the data after 16 cycles.
4 HW_RD_DL_EN	Enable automatic (HW) read calibration. If this bit is asserted then the MMDC will perform an automatic read calibration. HW should negate this bit upon completion of the calibration. Negation of this bit also points that the read calibration results are valid Note: Before issuing the first read command MMDC counts 12 cycles.
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
1 HW_RD_DL_ ERR1	Automatic (HW) read calibration error of Byte1. If this bit is asserted then it indicates that an error was found during the HW calibration process of read delay-line 1. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPRDDLHWST0 register. This bit is valid only after HW_RD_DL_EN is de-asserted.

Table continues on the next page...

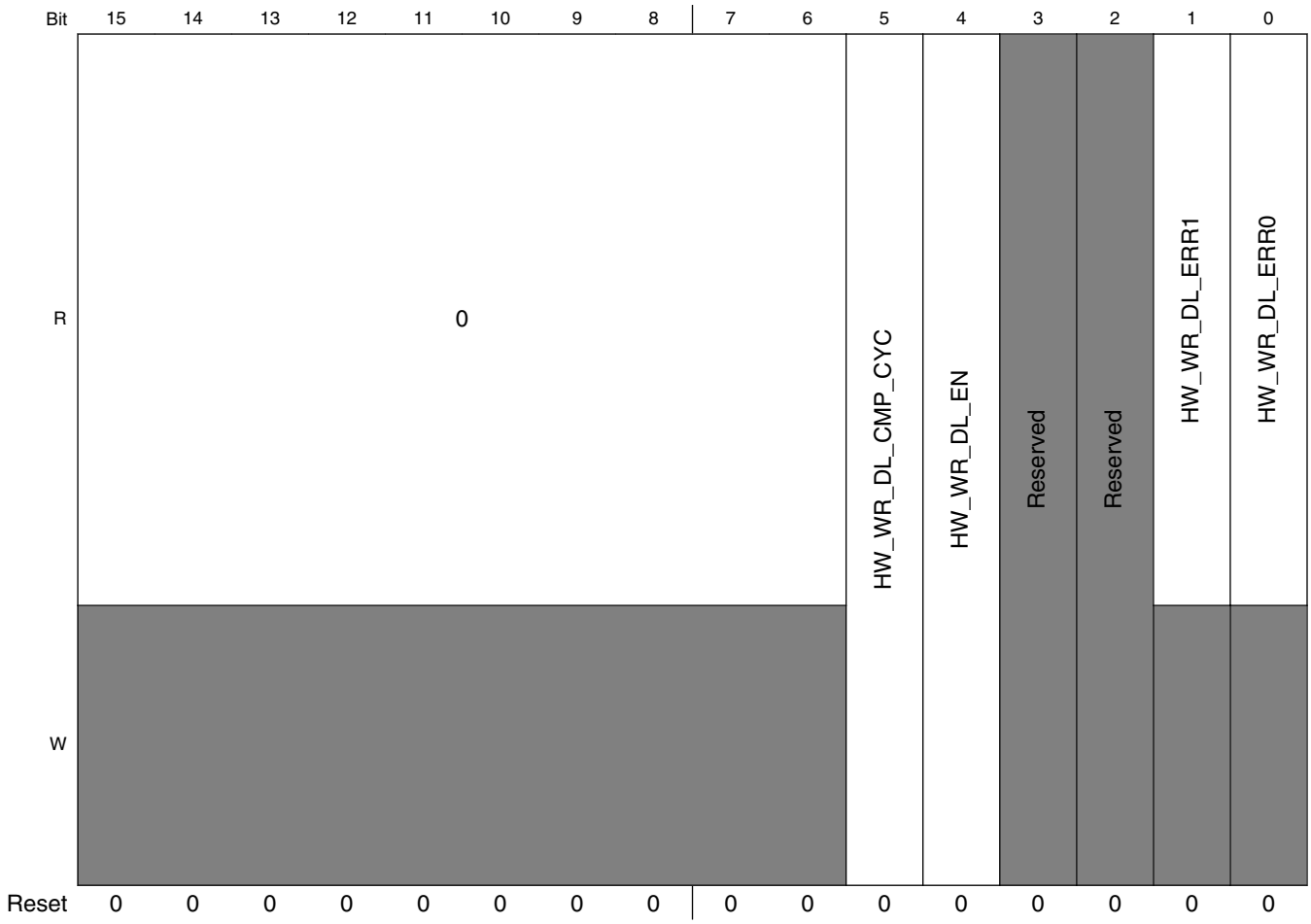
MMDC_MPRDDLHWCTL field descriptions (continued)

Field	Description
	0 No error was found in read delay-line 1 during the automatic (HW) read calibration process of read delay-line 1. 1 An error was found in read delay-line 1 during the automatic (HW) read calibration process of read delay-line 1.
0 HW_RD_DL_ERR0	Automatic (HW) read calibration error of Byte0. If this bit is asserted then it indicates that an error was found during the HW calibration process of read delay-line 0. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPRDDLHWST0 register. This bit is valid only after HW_RD_DL_EN is de-asserted. 0 No error was found in read delay-line 0 during the automatic (HW) read calibration process of read delay-line 0. 1 An error was found in read delay-line 0 during the automatic (HW) read calibration process of read delay-line 0.

33.12.50 MMDC PHY Write Delay HW Calibration Control Register (MMDC_MPWRDLHWCTL)

Address: 21B_0000h base + 864h offset = 21B_0864h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MMDC_MPWRDLHWCTL field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 HW_WR_DL_CMP_CYC	Write sample cycle. If this bit is asserted then the MMDC will compare the data 32 cycles after the MMDC sent the read command enable pulse else it compares the data after 16 cycles.
4 HW_WR_DL_EN	Enable automatic (HW) write calibration. If this bit is asserted then the MMDC will perform an automatic write calibration. HW should negate this bit upon completion of the calibration. Negation of this bit also indicates that the write calibration results are valid Note: Before issuing the first read command MMDC counts 12 cycles.
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
1 HW_WR_DL_ERR1	Automatic (HW) write calibration error of Byte1. If this bit is asserted then it indicates that an error was found during the HW calibration process of write delay-line 1. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPWRDLHWST0 register. This bit is valid only after HW_WR_DL_EN is de-asserted.

Table continues on the next page...

MMDC_MPWRDLHWCTL field descriptions (continued)

Field	Description
	0 No error was found during the automatic (HW) write calibration process of write delay-line 1. 1 An error was found during the automatic (HW) write calibration process of write delay-line 1.
0 HW_WR_DL_ERR0	Automatic (HW) write calibration error of Byte0. If this bit is asserted then it indicates that an error was found during the HW calibration process of write delay-line 0. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPWRDLHWST0 register. This bit is valid only after HW_WR_DL_EN is de-asserted. 0 No error was found during the automatic (HW) write calibration process of write delay-line 0. 1 An error was found during the automatic (HW) write calibration process of write delay-line 0.

33.12.51 MMDC PHY Read Delay HW Calibration Status Register 0 (MMDC_MPRDDLHWST0)

Address: 21B_0000h base + 868h offset = 21B_0868h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HW_RD_DL_UP1							0	HW_RD_DL_LOW1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HW_RD_DL_UP0							0	HW_RD_DL_LOW0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPRDDLHWST0 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 HW_RD_DL_UP1	Automatic (HW) read calibration result of the upper boundary of Byte1. This field holds the automatic (HW) read calibration result of the upper boundary of Byte1
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 HW_RD_DL_LOW1	Automatic (HW) read calibration result of the lower boundary of Byte1. This field holds the automatic (HW) read calibration result of the lower boundary of Byte1
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 HW_RD_DL_UP0	Automatic (HW) read calibration result of the upper boundary of Byte0. This field holds the automatic (HW) read calibration result of the upper boundary of Byte0.
7 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MPRDDLHWST0 field descriptions (continued)

Field	Description
HW_RD_DL_LOW0	Automatic (HW) read calibration result of the lower boundary of Byte0. This field holds the automatic (HW) read calibration result of the lower boundary of Byte0.

33.12.52 MMDC PHY Write Delay HW Calibration Status Register 0 (MMDC_MPWRDLHWST0)

Address: 21B_0000h base + 870h offset = 21B_0870h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HW_WR_DL_UP1							0	HW_WR_DL_LOW1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

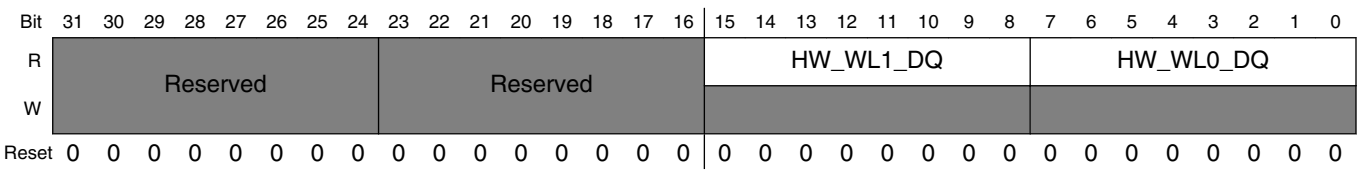
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HW_WR_DL_UP0							0	HW_WR_DL_LOW0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPWRDLHWST0 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 HW_WR_DL_UP1	Automatic (HW) write automatic (HW) write calibration result of the upper boundary of Byte1. This field holds the automatic (HW) write calibration result of the upper boundary of Byte1.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 HW_WR_DL_LOW1	Automatic (HW) write calibration result of the lower boundary of Byte1. This field holds the automatic (HW) write calibration result of the lower boundary of Byte1.
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 HW_WR_DL_UP0	Automatic (HW) write calibration result of the upper boundary of Byte0. This field holds the automatic (HW) write calibration result of the upper boundary of Byte0.
7 Reserved	This read-only field is reserved and always has the value 0.
HW_WR_DL_LOW0	Automatic (HW) write calibration result of the lower boundary of Byte0. This field holds the automatic (HW) write calibration result of the lower boundary of Byte0.

33.12.53 MMDC PHY Write Leveling HW Error Register (MMDC_MPWLHWERR)

Address: 21B_0000h base + 878h offset = 21B_0878h

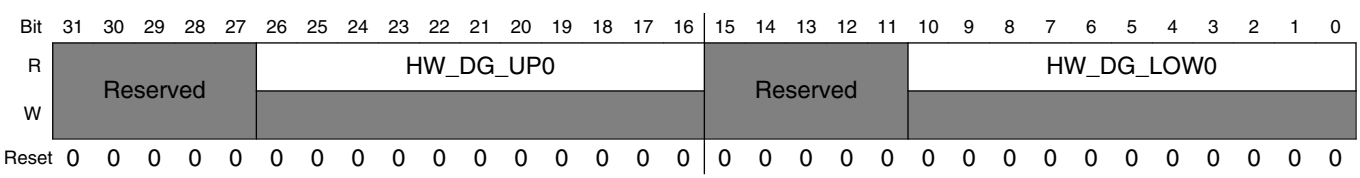


MMDC_MPWLHWERR field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 -	This field is reserved. Reserved
15–8 HW_WL1_DQ	HW write-leveling calibration result of Byte1. This field holds the results for all the 8 write-leveling steps of Byte1. i.e. bit 0 holds the result of the write-leveling calibration of 0 delay, bit 1holds the result of the write-leveling calibration of 1/8delay till bit 7 that holds the result of the write-leveling calibration of 7/8 delay
HW_WL0_DQ	HW write-leveling calibration result of Byte0. This field holds the results for all the 8 write-leveling steps of Byte0. i.e. bit 0 holds the result of the write-leveling calibration of 0 delay, bit 1holds the result of the write-leveling calibration of 1/8delay till bit 7 that holds the result of the write-leveling calibration of 7/8 delay

33.12.54 MMDC PHY Read DQS Gating HW Status Register 0 (MMDC_MPDGHWST0)

Address: 21B_0000h base + 87Ch offset = 21B_087Ch

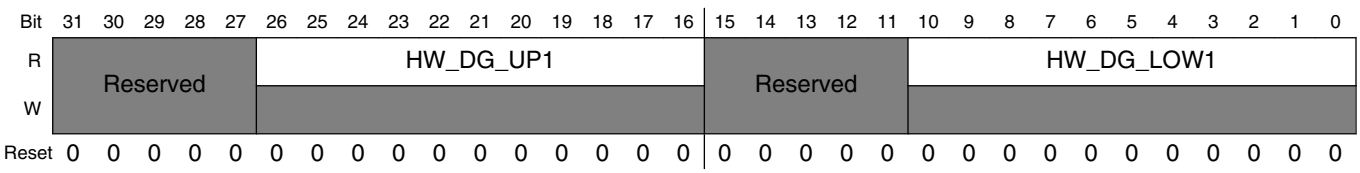


MMDC_MPDGHWST0 field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26–16 HW_DG_UP0	HW DQS gating calibration result of the upper boundary of Byte0. This field holds the HW DQS gating calibration result of the upper boundary of Byte0.
15–11 -	This field is reserved. Reserved
HW_DG_LOW0	HW DQS gating calibration result of the lower boundary of Byte0. This field holds the HW DQS gating calibration result of the lower boundary of Byte0.

33.12.55 MMDC PHY Read DQS Gating HW Status Register 1 (MMDC_MPDGHWST1)

Address: 21B_0000h base + 880h offset = 21B_0880h



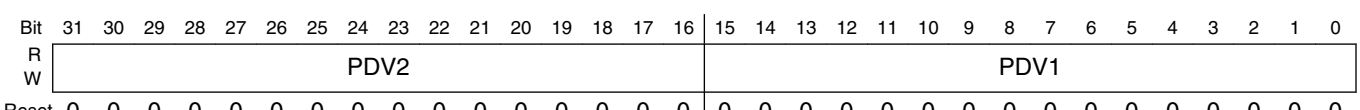
MMDC_MPDGHWST1 field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26–16 HW_DG_UP1	HW DQS gating calibration result of the upper boundary of Byte1. This field holds the HW DQS gating calibration result of the upper boundary of Byte1.
15–11 -	This field is reserved. Reserved
HW_DG_LOW1	HW DQS gating calibration result of the lower boundary of Byte1. This field holds the HW DQS gating calibration result of the lower boundary of Byte1.

33.12.56 MMDC PHY Pre-defined Compare Register 1 (MMDC_MPPDCMPR1)

This register holds the MMDC pre-defined compare value that will be used during automatic read, read DQS gating and write calibration process. The compare value can be the MPR value (as defined in the JEDEC) or can be programmed by the PDV1 and PDV2 fields.In case of DDR3 (BL=8) the MMDC will duplicate PDV1,PDV2 and drive that data on Beat4-7 of the same byte

Address: 21B_0000h base + 88Ch offset = 21B_088Ch



MMDC_MPPDCMPR1 field descriptions

Field	Description
31–16 PDV2	MMDC Pre defined compare value2. This field holds the 2 MSB of the data that will be driven to the DDR device during automatic read, read DQS gating and write calibrations in case MPR(DDR3)/ DQ calibration

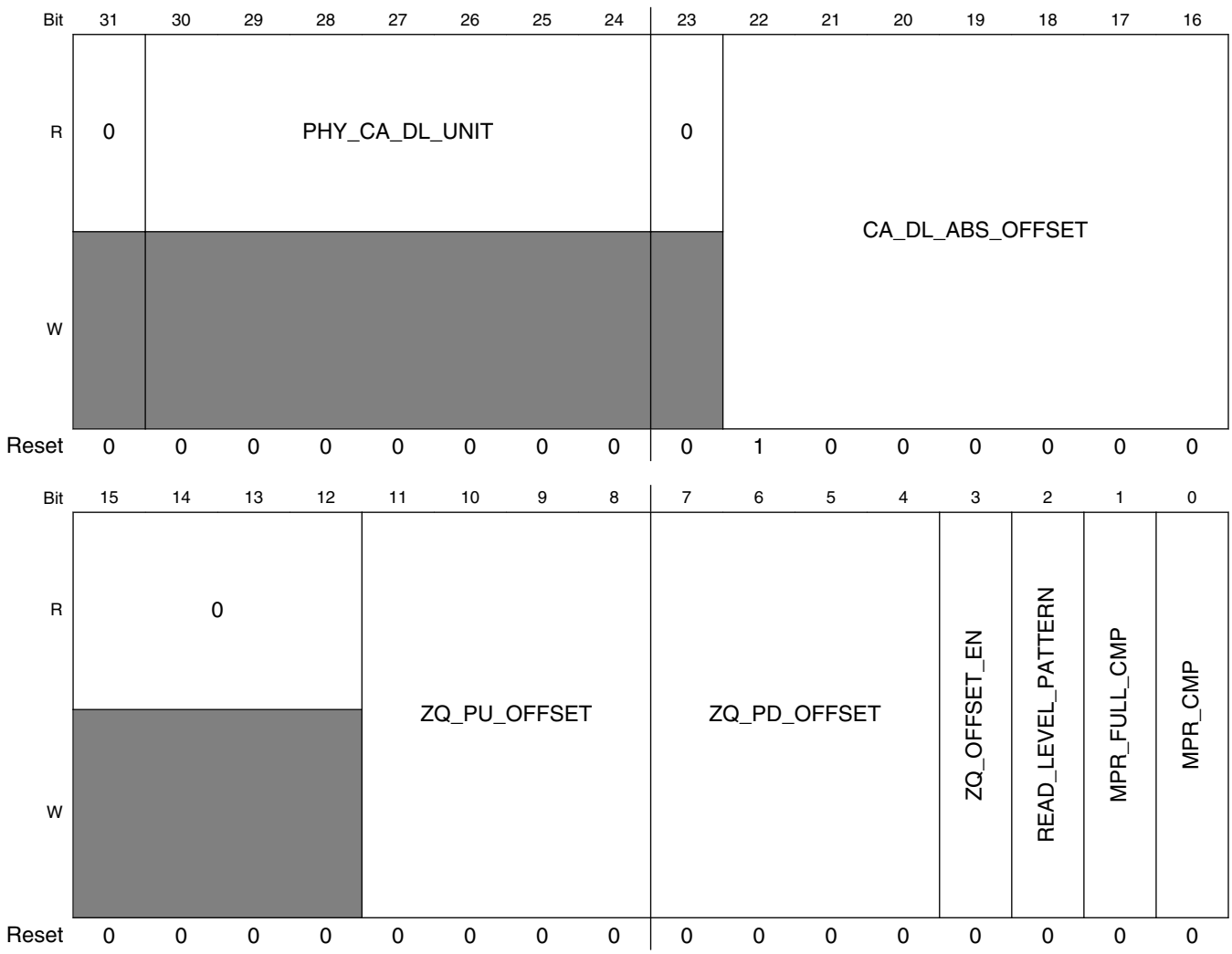
Table continues on the next page...

MMDC_MPPDCMPR1 field descriptions (continued)

Field	Description
	(LPDDR2/LPDDR3) mode are disabled (MPR_CMP is disabled). Upon read access during the calibration the MMDC will compare the read data with the data that is stored in this field. Note : Before issue the read access the MMDC will invert the value of this field and drive it to the associate entry in the read comparison FIFO. For further information see Section 19.14.3.1.2, "Calibration with pre-defined value , Section 19.14.4.1.2, "Calibration with pre-defined value and Section 19.14.5.1, "HW (automatic) Write Calibration
PDV1	MMDC Pre defined compare value2. This field holds the 2 LSB of the data that will be driven to the DDR device during automatic read, read DQS gating and write calibrations in case MPR(DDR3)/ DQ calibration (LPDDR2/LPDDR3) mode are disabled (MPR_CMP is disabled). Upon read access during the calibration the MMDC will compare the read data with the data that is stored in this field. **NOTE:** Before issuing the read access, the MMDC will invert the value of this field and drive it to the associated entry in the read comparison FIFO.

33.12.57 MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MMDC_MPPDCMPR2)

Address: 21B_0000h base + 890h offset = 21B_0890h



MMDC_MPPDCMPR2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 PHY_CA_DL_ UNIT	This field reflects the number of delay units that are actually used by CA(Command/Address of LPDDR2) delay-line
23 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDC_MPPDCMPR2 field descriptions (continued)

Field	Description
22–16 CA_DL_ABS_OFFSET	Absolute CA (Command/Address of LPDDRR2) offset. This field indicates the absolute delay between CA (Command/Address) bus and the DDR clock (CK) with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (CA_DL_ABS_OFFSET / 256) * MMDC AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay.
15–12 Reserved	This read-only field is reserved and always has the value 0.
11–8 ZQ_PU_OFFSET	Programmable offset from -7 to 7 added to the MMDC_MPZQHWCTRL[ZQ_HW_PU_RES] field when ZQ_OFFSET_EN is enabled. Bit[11] determines direction: 'b0 = Offset is added to ZQ_HW_PU_RES field 'b1 = Offset is subtracted from ZQ_HW_PU_RES field Bits[10:8] = Amount of change to apply. 0000 **0** — Offset is added to ZQ_HW_PU_RES field 0001 **1** — Offset is added to ZQ_HW_PU_RES field 0010 **2** — Offset is added to ZQ_HW_PU_RES field 0011 **3** — Offset is added to ZQ_HW_PU_RES field 0100 **4** — Offset is added to ZQ_HW_PU_RES field 0101 **5** — Offset is added to ZQ_HW_PU_RES field 0110 **6** — Offset is added to ZQ_HW_PU_RES field 0111 **7** — Offset is added to ZQ_HW_PU_RES field 1000 **0** — Offset is subtracted from ZQ_HW_PU_RES field 1001 **1** — Offset is subtracted from ZQ_HW_PU_RES field 1010 **2** — Offset is subtracted from ZQ_HW_PU_RES field 1011 **3** — Offset is subtracted from ZQ_HW_PU_RES field 1100 **4** — Offset is subtracted from ZQ_HW_PU_RES field 1101 **5** — Offset is subtracted from ZQ_HW_PU_RES field 1110 **6** — Offset is subtracted from ZQ_HW_PU_RES field 1111 **7** — Offset is subtracted from ZQ_HW_PU_RES field
7–4 ZQ_PD_OFFSET	Programmable offset from -7 to 7 added to the MMDC_MPZQHWCTRL[ZQ_HW_PD_RES] field when ZQ_OFFSET_EN is enabled. Bit[7] determines direction: 'b0 = Offset is added to ZQ_HW_PD_RES field 'b1 = Offset is subtracted from ZQ_HW_PD_RES field Bits[6:4] = Amount of change to apply. 0000 **0** — Offset is added to ZQ_HW_PD_RES field 0001 **1** — Offset is added to ZQ_HW_PD_RES field 0010 **2** — Offset is added to ZQ_HW_PD_RES field 0011 **3** — Offset is added to ZQ_HW_PD_RES field 0100 **4** — Offset is added to ZQ_HW_PD_RES field 0101 **5** — Offset is added to ZQ_HW_PD_RES field 0110 **6** — Offset is added to ZQ_HW_PD_RES field 0111 **7** — Offset is added to ZQ_HW_PD_RES field 1000 **0** — Offset is subtracted from ZQ_HW_PD_RES field

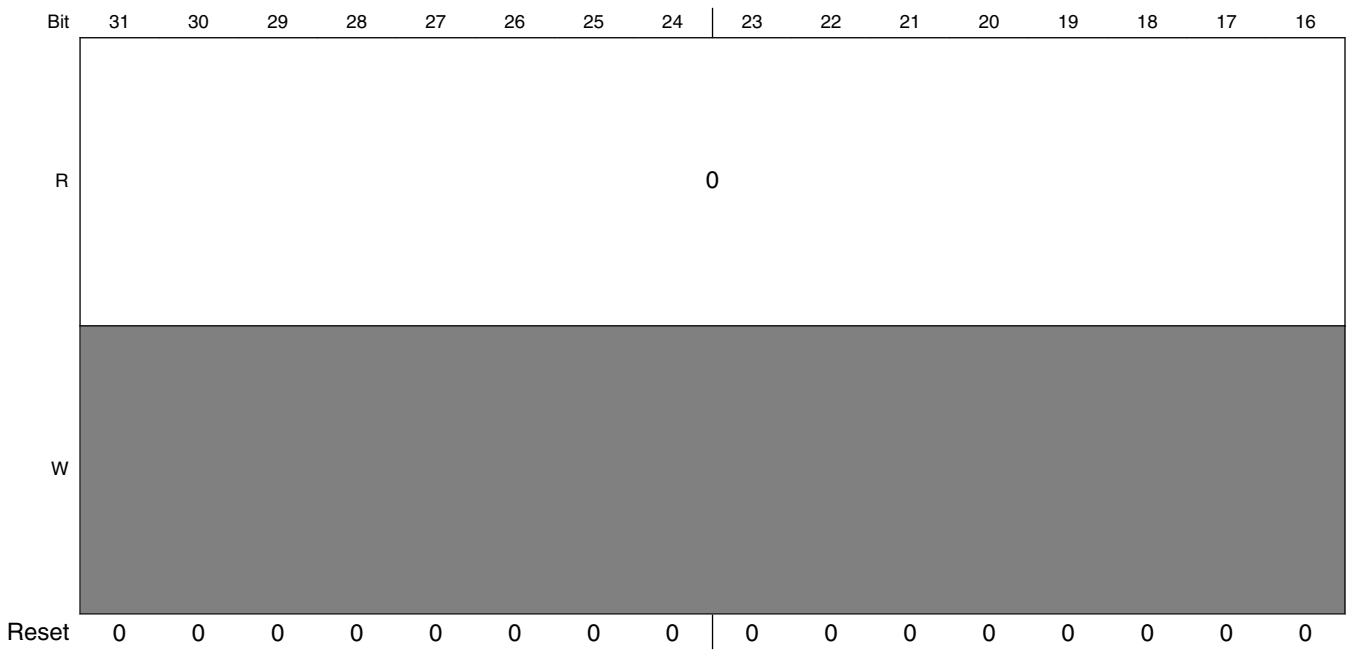
Table continues on the next page...

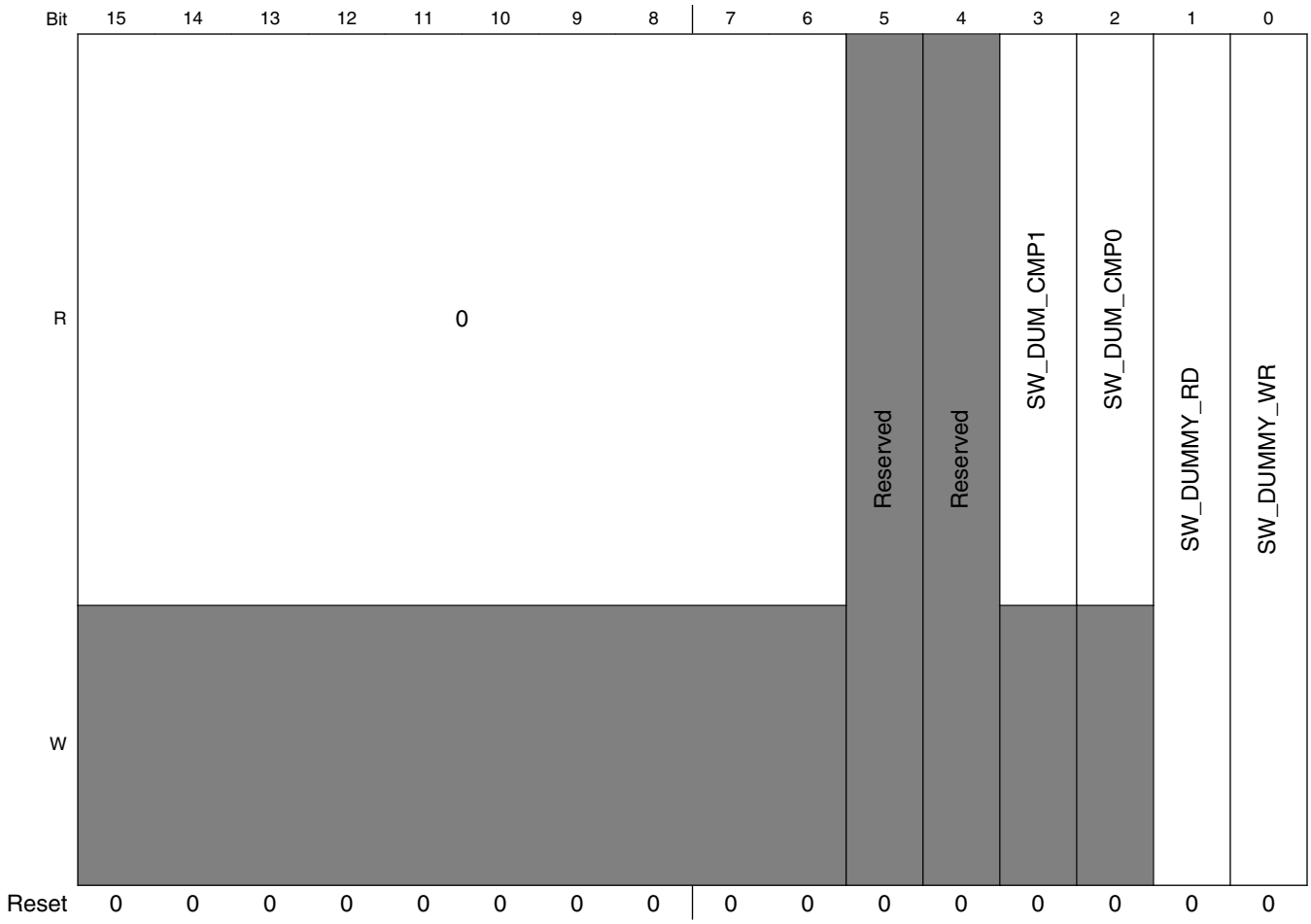
MMDC_MPPDCMPR2 field descriptions (continued)

Field	Description
	1001 **1** — Offset is subtracted from ZQ_HW_PD_RES field 1010 **2** — Offset is subtracted from ZQ_HW_PD_RES field 1011 **3** — Offset is subtracted from ZQ_HW_PD_RES field 1100 **4** — Offset is subtracted from ZQ_HW_PD_RES field 1101 **5** — Offset is subtracted from ZQ_HW_PD_RES field 1110 **6** — Offset is subtracted from ZQ_HW_PD_RES field 1111 **7** — Offset is subtracted from ZQ_HW_PD_RES field
3 ZQ_OFFSET_EN	0 Hardware ZQ offset disabled 1 Hardware ZQ offset enabled
2 READ_LEVEL_ PATTERN	MPR(DDR3)/ DQ calibration (LPDDR2/LPDDR3) read compare pattern. In case MPR(DDR3)/ DQ calibration (LPDDR2/LPDDR3) modes are used during the calibration process (MPR_CMP is asserted) then this field indicates the read pattern for the comparison. 0 Compare with read pattern 1010 1 Compare with read pattern 0011 (Used only in LPDDR2/LPDDR3 mode)
1 MPR_FULL_ CMP	MPR(DDR3)/ DQ calibration (LPDDR2/LPDDR3) full compare enable. In case MPR(DDR3)/ DQ calibration (LPDDR2/LPDDR3) modes are used during the calibration process (MPR_CMP is asserted) then this field indicates whether the MMDC will compare all the bits of the data that is read from the DDR device to the MPR pre-defined pattern. When this bit is de-asserted only LSB of each byte is compared.
0 MPR_CMP	MPR(DDR3)/ DQ calibration (LPDDR2/LPDDR3) compare enable. This bit indicates whether the MMDC will compare the read data during automatic read and read DQS calibration processes to the pre-defined patterns that are driven by the DDR device (READ_LEVEL_PATTERN as defined by JEDEC) or general pre-defined value that are stored in PDV1 and PDV2. When this bit is disabled data is compared to the data of the pre defined compare value field For further information see Read DQS Gating Calibration and Read Calibration .

33.12.58 MMDC PHY SW Dummy Access Register (MMDC_MPSWDAR0)

Address: 21B_0000h base + 894h offset = 21B_0894h





MMDC_MPSWDAR0 field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 -	This field is reserved. Reserved
4 -	This field is reserved. Reserved
3 SW_DUM_CMP1	SW dummy read byte1 compare results. This bit indicates the result of the read data comparison of Byte1 at the completion of SW_DUMMY_RD. This bit is valid only when SW_DUMMY_RD is de-assrted. 0 Dummy read fail 1 Dummy read pass
2 SW_DUM_CMP0	SW dummy read byte0 compare results. This bit indicates the result of the read data comparison of Byte0 at the completion of SW_DUMMY_RD. This bit is valid only when SW_DUMMY_RD is de-assrted. 0 Dummy read fail 1 Dummy read pass
1 SW_DUMMY_RD	SW dummy read. When this bit is asserted the MMDC will generate internally read access without intervention of the system toward bank 0, row 0, column 0. If MPR_CMP = 1then the read data will be compared to MPPDCMPR2[READ_LEVEL_PATTERN] . If MPR_CMP =0 then the read data will be

Table continues on the next page...

MMDC_MPSWDAR0 field descriptions (continued)

Field	Description
	compared to MPPDCMPR1[PDV1], MPPDCMPR1[PDV2]. Upon completion of the access this bit is de-asserted automatically and the read data and comparison results are valid at MPSWDAR0[SW_DUM_CMP#] and MPSWDRDR0-MPSWDRDR7 respectively.
0 SW_DUMMY_WR	SW dummy write. When this bit is asserted the MMDC will generate internally write access without intervention of the system toward bank 0, row 0, column 0, while the data is driven from MPPDCMPR1[PDV1] and MPPDCMPR1[PDV2]. The bit is de-asserted automatically upon completion of the access.

33.12.59 MMDC PHY SW Dummy Read Data Register 0 (MMDC_MPSWDRDR0)

Address: 21B_0000h base + 898h offset = 21B_0898h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD0																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDC_MPSWDRDR0 field descriptions

Field	Description
DUM_RD0	Dummy read data0. This field holds the first data that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted

33.12.60 MMDC PHY SW Dummy Read Data Register 1 (MMDC_MPSWDRDR1)

Address: 21B_0000h base + 89Ch offset = 21B_089Ch

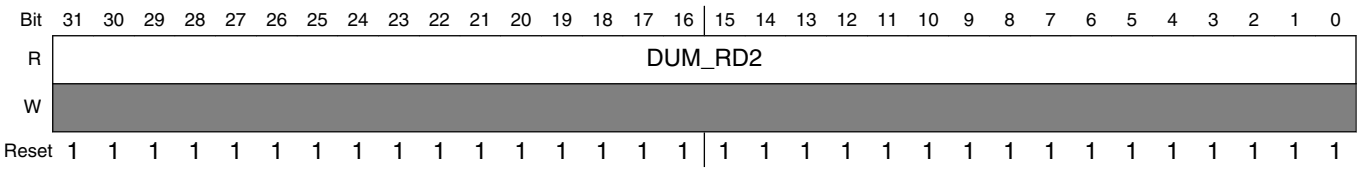
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD1																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDC_MPSWDRDR1 field descriptions

Field	Description
DUM_RD1	Dummy read data1. This field holds the second data that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted

33.12.61 MMDC PHY SW Dummy Read Data Register 2 (MMDC_MPSWDRDR2)

Address: 21B_0000h base + 8A0h offset = 21B_08A0h

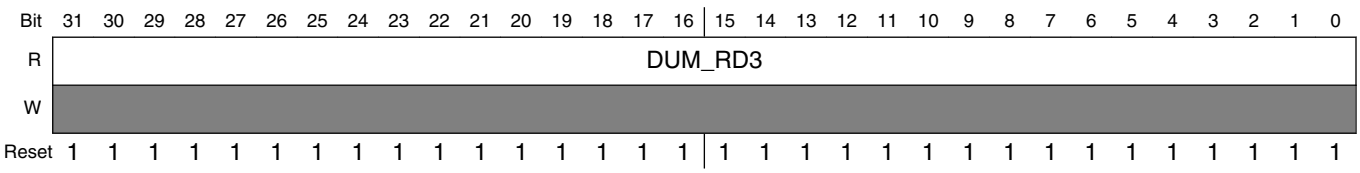


MMDC_MPSWDRDR2 field descriptions

Field	Description
DUM_RD2	Dummy read data2. This field holds the third data that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

33.12.62 MMDC PHY SW Dummy Read Data Register 3 (MMDC_MPSWDRDR3)

Address: 21B_0000h base + 8A4h offset = 21B_08A4h

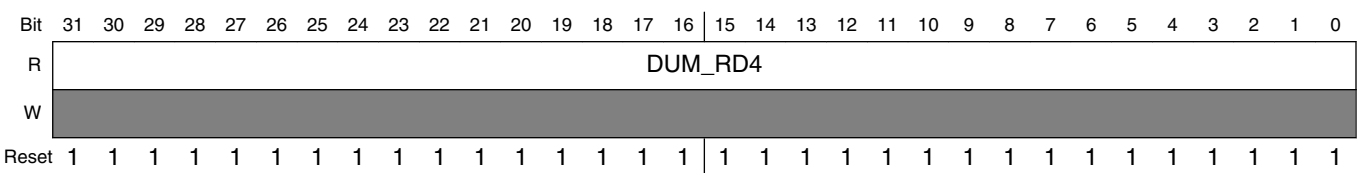


MMDC_MPSWDRDR3 field descriptions

Field	Description
DUM_RD3	Dummy read data3. This field holds the forth data that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

33.12.63 MMDC PHY SW Dummy Read Data Register 4 (MMDC_MPSWDRDR4)

Address: 21B_0000h base + 8A8h offset = 21B_08A8h



MMDC_MPSWDRDR4 field descriptions

Field	Description
DUM_RD4	Dummy read data4. This field holds the fifth data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

33.12.64 MMDC PHY SW Dummy Read Data Register 5 (MMDC_MPSWDRDR5)

Address: 21B_0000h base + 8ACh offset = 21B_08ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD5																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDC_MPSWDRDR5 field descriptions

Field	Description
DUM_RD5	Dummy read data5. This field holds the sixth data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

33.12.65 MMDC PHY SW Dummy Read Data Register 6 (MMDC_MPSWDRDR6)

Address: 21B_0000h base + 8B0h offset = 21B_08B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD6																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDC_MPSWDRDR6 field descriptions

Field	Description
DUM_RD6	Dummy read data6. This field holds the seventh data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

33.12.66 MMDC PHY SW Dummy Read Data Register 7 (MMDC_MPSWDRDR7)

Address: 21B_0000h base + 8B4h offset = 21B_08B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD7																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDC_MPSWDRDR7 field descriptions

Field	Description
DUM_RD7	Dummy read data7. This field holds the eight data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

33.12.67 MMDC PHY Measure Unit Register (MMDC_MPMUR0)

Address: 21B_0000h base + 8B8h offset = 21B_08B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						MU_UNIT_DEL_NUM									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

MMDC_MPMUR0 field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.
25–16 MU_UNIT_DEL_NUM	Number of delay units measured per cycle. This field is used in debug mode and holds the number of delay units that were measured by the measure unit per DDR clock cycle. The delay-lines that are used in every calibration process use that number for generating the desired delay.
15–12 Reserved	This read-only field is reserved and always has the value 0.
11 FRC_MSR	Force measurement on delay-lines. When this bit is asserted then a measurement process will be performed, where at the completion of the process the delay-lines will issue the desired delay. Upon completion of the measurement process the measure unit and the delay-lines will return to functional mode. This bit is self cleared.

Table continues on the next page...

MMDC_MPMUR0 field descriptions (continued)

Field	Description
	NOTE: This bit should be used only during manual (SW) calibration and not while the DDR is functional (being accessed). After initial calibration is done the hardware performs periodic measurements to track any operating conditions changes. Hence, force measurements (FRC_MSR) should not be used. See Calibration Process for more information. **NOTE:** User should make sure that there is no active accesses to/from DDR before asserting this bit. 0 No measurement is performed 1 Perform measurement process
10 MU_BYP_EN	Measure unit bypass enable. This field is used in debug mode and when it is asserted then the delay-lines will use the number of delay units that are indicated at MU_BYP_VAL, otherwise the delay-lines will use the number of delay units that was measured by the measurement unit and are indicated at MU_UNIT_DEL_NUM 0 The delay-lines use delay units as indicated at MU_UNIT_DEL_NUM. 1 The delay-lines use delay units as indicated at MU_BYPASS_VAL.
MU_BYP_VAL	Number of delay units for measurement bypass. This field is used in debug mode and holds the number of delay units that will be used by the delay-lines when MU_BYP_EN is asserted.

33.12.68 MMDC Write CA delay-line controller (MMDC_MPWRCADL)

This register is used to add fine-tuning adjustment to the CA (command/Address of LPDDR2 bus) relative to the DDR clock

Address: 21B_0000h base + 8BCh offset = 21B_08BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												WR_CA9_DEL		WR_CA8_DEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WR_CA7_DEL		WR_CA6_DEL		WR_CA5_DEL		WR_CA4_DEL		WR_CA3_DEL		WR_CA2_DEL		WR_CA1_DEL		WR_CA0_DEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDC_MPWRCADL field descriptions

Field	Description
31–20 Reserved	This read-only field is reserved and always has the value 0.
19–18 WR_CA9_DEL	CA (Command/Address LPDDR2 bus) bit 9 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 9 relative to the clock.

Table continues on the next page...

MMDC_MPWRCADL field descriptions (continued)

Field	Description
	00 No change in CA9 delay 01 Add CA9 delay of 1 delay unit 10 Add CA9 delay of 2 delay units. 11 Add CA9 delay of 3 delay units.
17–16 WR_CA8_DEL	CA (Command/Address LPDDR2 bus) bit 8 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 8 relative to the clock. 00 No change in CA8 delay 01 Add CA8 delay of 1 delay unit 10 Add CA8 delay of 2 delay units. 11 Add CA8 delay of 3 delay units.
15–14 WR_CA7_DEL	CA (Command/Address LPDDR2 bus) bit 7 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 7 relative to the clock. 00 No change in CA7 delay 01 Add CA7 delay of 1 delay unit 10 Add CA7 delay of 2 delay units. 11 Add CA7 delay of 3 delay units.
13–12 WR_CA6_DEL	CA (Command/Address LPDDR2 bus) bit 6 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 6 relative to the clock. 00 No change in CA6 delay 01 Add CA6 delay of 1 delay unit 10 Add CA6 delay of 2 delay units. 11 Add CA6 delay of 3 delay units.
11–10 WR_CA5_DEL	CA (Command/Address LPDDR2 bus) bit 5 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 5 relative to the clock. 00 No change in CA5 delay 01 Add CA5 delay of 1 delay unit 10 Add CA5 delay of 2 delay units. 11 Add CA5 delay of 3 delay units.
9–8 WR_CA4_DEL	CA (Command/Address LPDDR2 bus) bit 4 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 4 relative to the clock. 00 No change in CA4 delay 01 Add CA4 delay of 1 delay unit 10 Add CA4 delay of 2 delay units. 11 Add CA4 delay of 3 delay units.
7–6 WR_CA3_DEL	CA (Command/Address LPDDR2 bus) bit 3 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 3 relative to the clock. 00 No change in CA3 delay 01 Add CA3 delay of 1 delay unit 10 Add CA3 delay of 2 delay units. 11 Add CA3 delay of 3 delay units.
5–4 WR_CA2_DEL	CA (Command/Address LPDDR2 bus) bit 2 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 2 relative to the clock.

Table continues on the next page...

MMDC_MPWRCADL field descriptions (continued)

Field	Description
	00 No change in CA2 delay 01 Add CA2 delay of 1 delay unit 10 Add CA2 delay of 2 delay units. 11 Add CA2 delay of 3 delay units.
3–2 WR_CA1_DEL	CA (Command/Address LPDDR2 bus) bit 1 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 1 relative to the clock. 00 No change in CA1 delay 01 Add CA1 delay of 1 delay unit 10 Add CA1 delay of 2 delay units. 11 Add CA1 delay of 3 delay units.
WR_CA0_DEL	CA(Command/Address LPDDR2 bus) bit 0 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 0 relative to the clock. 00 No change in CA0 delay 01 Add CA0 delay of 1 delay unit 10 Add CA0 delay of 2 delay units. 11 Add CA0 delay of 3 delay units.

33.12.69 MMDC Duty Cycle Control Register (MMDC_MPDCCR)

This register is used to control the duty cycle of the DQS and the primary clock (CK0) . Programming of that register is permitted by entering the DDR device into self-refresh mode through LPMD/DVFS mechanism

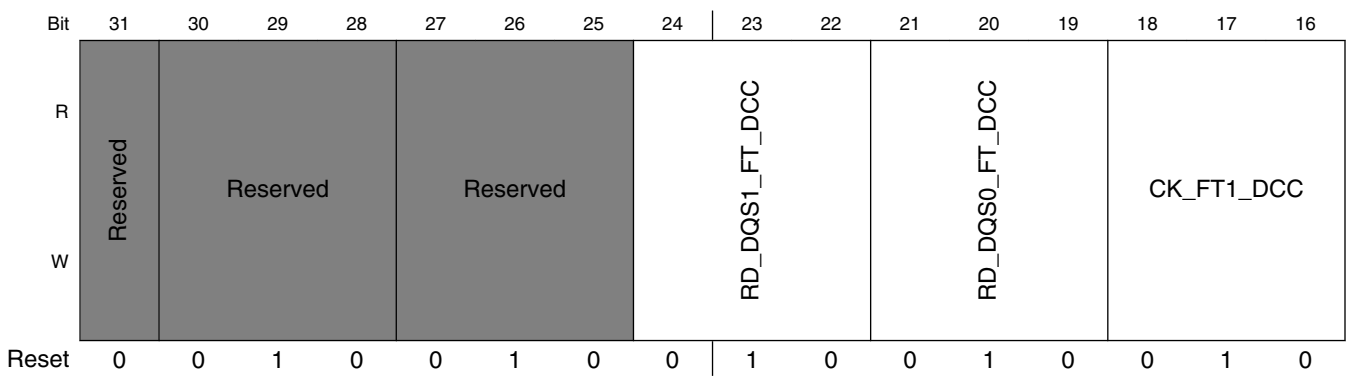
NOTE

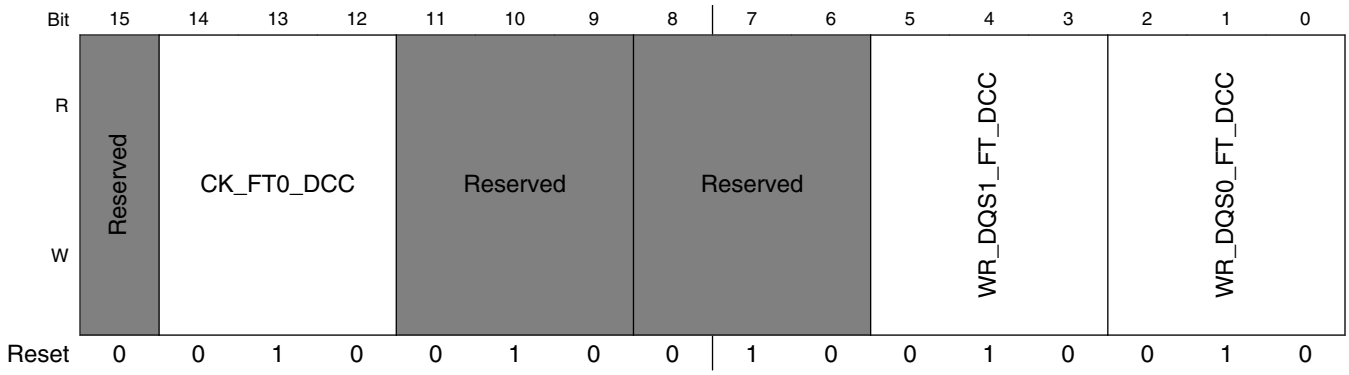
If the duty cycle is modified after DDR initialization, the DDR will have to be placed in self-refresh mode.

NOTE

The duty cycle may be changed during initial DDR initialization without having to be placed in self-refresh mode.

Address: 21B_0000h base + 8C0h offset = 21B_08C0h





MMDC_MPDCCR field descriptions

Field	Description
31 -	This field is reserved. reserved
30–28 -	This field is reserved. Reserved
27–25 -	This field is reserved. Reserved
24–22 RD_DQS1_FT_DCC	Read DQS duty cycle fine tuning control of Byte1. This field controls the duty cycle of read DQS of Byte1 Note all the other options are not allowed 001 51.5% low 48.5% high 010 50% duty cycle (default) 100 48.5% low 51.5% high
21–19 RD_DQS0_FT_DCC	Read DQS duty cycle fine tuning control of Byte0. This field controls the duty cycle of read DQS of Byte0 Note all the other options are not allowed 001 51.5% low 48.5% high 010 50% duty cycle (default) 100 48.5% low 51.5% high
18–16 CK_FT1_DCC	Secondary duty cycle fine tuning control of DDR clock. This field controls the duty cycle of the DDR clock and is cascaded to CK_FT0_DCC Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
15 -	This field is reserved. Reserved
14–12 CK_FT0_DCC	Primary duty cycle fine tuning control of DDR clock. This field controls the duty cycle of the DDR clock Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
11–9 -	This field is reserved. Reserved

Table continues on the next page...

MMDC_MPDCCR field descriptions (continued)

Field	Description
8–6 -	This field is reserved. Reserved
5–3 WR_DQS1_FT_DCC	Write DQS duty cycle fine tuning control of Byte1. This field controls the duty cycle of write DQS of Byte1 Note all the other options are not allowed 001 51.5% low 48.5% high 010 50% duty cycle (default) 100 48.5% low 51.5% high
WR_DQS0_FT_DCC	Write DQS duty cycle fine tuning control of Byte0. This field controls the duty cycle of write DQS of Byte0 Note all the other options are not allowed 001 51.5% low 48.5% high 010 50% duty cycle (default) 100 48.5% low 51.5% high

Chapter 34
Medium Quality Sound (MQS)

34.1 Overview

Medium quality sound (MQS) is used to generate medium quality audio via a standard GPIO in the pinmux, allowing the user to connect stereo speakers or headphones to a power amplifier without an additional DAC chip.

MQS accepts 2-channel, LSB-valid 16bit, MSB shift-out first; frame sync asserting with the first bit of the frame, data shifted with the posedge of bit clock, 44.1 kHz or 48 kHz signals from SAI1 in left_justified format; and it provides the SNR target as no more than 20dB for the signals below 10 kHz. The signals above 10 kHz will have worse THD+N values.

MQS provides only simple audio reproduction. No internal pop, click or distortion artifact reduction methods are provided.

34.2 Block Diagram

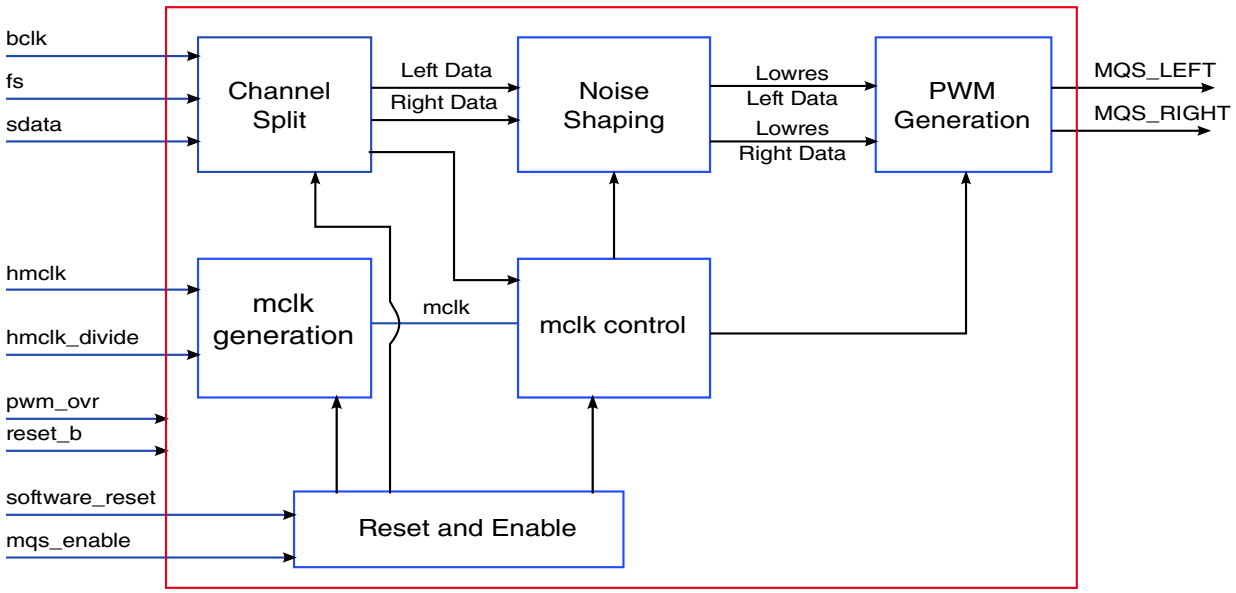


Figure 34-1. Block Diagram

MQS has the following sub-modules:

1. Channel Split: Splits the I2S signals into separate left channel and right channel audio data.
2. Noise Shaping: Uses the sigma-delta algorithm to generate low-resolution, very high sampling audio, while the audio sampling rate is increased.
3. PWM generation: Generates the bit stream to the GPIO, which is then used to drive the amplifier and then to drive the external speakers or headphones.
4. mclk generation: Used to generate the master clock (mclk). The frequency of mclk is determined by the final bit duration of PWM generation module.
5. mclk control:Used as a metronome to co-ordinate the different functional blocks working synchronously.
6. Reset and Enable: Used to generate the reset and enable logic to different clock domains.

34.3 External Signals

The following table describes the external signals of MQS:

Table 34-1. MQS External Signals

Signal	Description	Pad	Mode	Direction
MQS_LEFT	Left signal output	GPIO1_IO01	ALT4	O
		JTAG_TDI	ALT6	

Table continues on the next page...

Table 34-1. MQS External Signals (continued)

Signal	Description	Pad	Mode	Direction
		LCD_DATA23	ALT1	
MQS_RIGHT	Right signal output	GPIO1_IO00	ALT4	O
		JTAG_TDO	ALT6	
		LCD_DATA22	ALT1	

34.4 Programmability

MQS has no internal programmable registers. But it does have some programmability from IOMUXC_GPR2.

Register Bits	Name	Description
IOMUXC_GPR2[26]	MQS_OVERSAMPLE	Used to control the PWM oversampling rate compared with mclk. 1—64, 0—32.
IOMUXC_GPR2[25]	MQS_EN	MQS enable. 1—Enable MQS, 0—Disable MQS
IOMUXC_GPR2[24]	MQS_SW_RST	MQS software reset. 1—Enable software reset for MQS 0—Exit software reset for MQS
IOMUXC_GPR2[23:16]	MQS_CLK_DIV[7:0]	Divider ration control for mclk from hmclk. 0—mclk frequency = hmclk frequency; 1—mclk frequency = ½*hmclk frequency; 2—mclk frequency = 1/3*hmclk frequency; …; n—mclk frequency = 1/(n+1)*hmclk frequency

34.5 Usage Model

The user needs to program SAI1 to output 2-channel MSB-16bit active I2S signal, and then to program the related IOMUXC_GPR2 bits.

Due to the different devices connected to MQS, and different high frequency behaviors of the connected analog circuits, customer need choose the appropriate MQS_CLK_DIV and MQS_OVERSAMPLE values for the best audible effects.

Chapter 35
On-Chip OTP Controller (OCOTP_CTRL)

35.1 Overview

This section contains information describing the requirements for the on-chip eFuse OTP controller along with details about the block functionality and implementation.

In this document, the words "eFuse" and "OTP" are interchangeable. OCOTP refers to the hardware block itself.

35.1.1 Features

The OCOTP provides the following features :

- Loading and housing of fuse content into shadow registers.
- Generation of HWV_FUSE (hardware visible fuse bus) and the HWV_REG bus which is made of up of volatile PIO register based "fuses". The HWV_REG bits come from the SCS (Software Controllable Signals) register.
- Generation of STICKY_REG which is consist of sticky register bits.
- Provide program-protect and read-protect eFuse.
- Provide override and read protection of shadow register.
- CRC32 test for read-lock fuse content.

35.2 Clocks

The table found here describes the clock sources for OCOTP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 35-1. OCOTP Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

35.3 Top-Level Symbol and Functional Overview

The figure found here shows the OCOTP system level diagram.

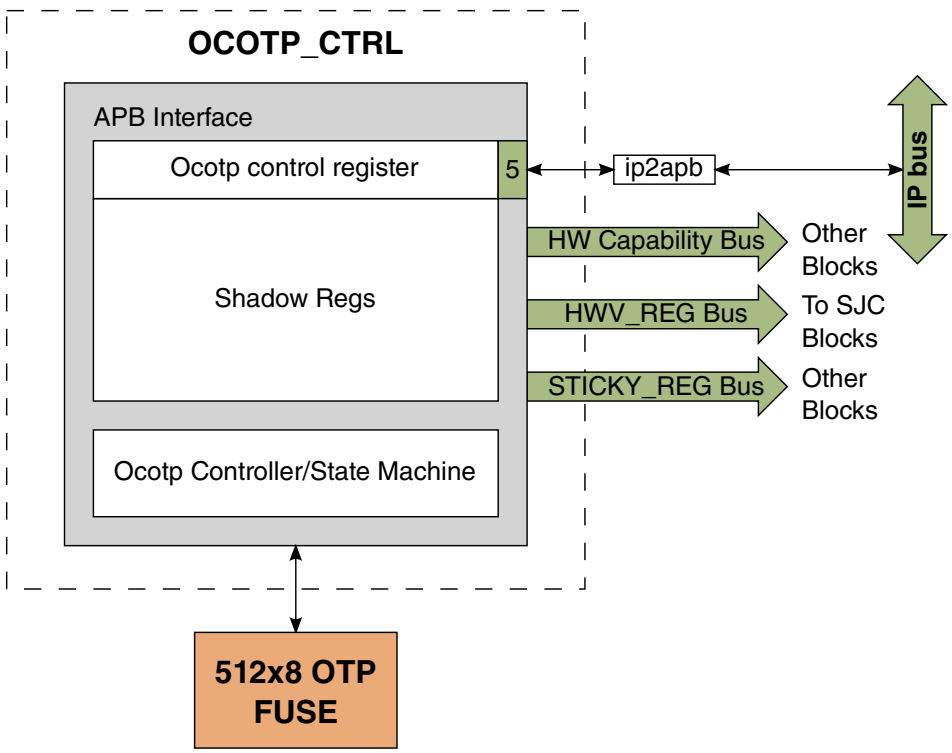


Figure 35-1. OCOTP System Level Diagram

35.3.1 Operation

The IP bus interface of the OCOTP provides two functions.

- Configure control registers for programming and reading fuse .
- Override and read shadow registers.

35.3.1.1 Shadow Register Reload

All fuse words in are shadowed. Therefore, fuse information is available through memory mapped shadow registers. If fuses are subsequently programmed, the shadow registers should be reloaded to keep them coherent with the fuse bank arrays.

The "reload shadows" feature allows the user to force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device. To force a reload, complete the following steps:

1. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write , read or reload must be completed before a new access can be requested.
2. Set the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit. OCOTP will read all the fuse one by one and put it into corresponding shadow register.
3. Wait for HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[RELOAD_SHADOWS] to be cleared by the controller.

The controller will automatically clear the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit after the successful completion of the operation.

35.3.1.2 Fuse and Shadow register read

All shadow registers are always readable through the APB bus except some secret keys regions. When their corresponding fuse lock bits are set, the shadow registers also become read locked. After read locking, reading from these registers will return 0xBADABADA.

In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write , read or reload access can be issued. Subsequent reads to unlocked shadow locations will still work successfully however.

To read fuse words directly from correctly complete the following steps:

1. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write, read or reload must be completed before a read access can be requested.
2. Write the requested to HW_OCOTP_CTRL[ADDR].

35.3.1.3 Fuse and Shadow Register Writes

Shadow register bits can be overridden by software until the corresponding fuse lock bit for the region is set. When the lock shadow bit is set, the shadow registers for that lock region become write locked. The LOCK shadow register also has no shadow or fuse lock bits but it is always read only.

In order to avoid "rogue" code performing erroneous writes to OTP, a special unlocking sequence is required for writes to the fuse banks. To program fuse bank correctly complete the following steps:

1. Program HW_OCOTP_TIMING[STROBE_PROG] and HW_OCOTP_TIMING[RELAX] fields with timing values to match the current frequency of the ipg_clk. OTP writes will work at maximum bus frequencies as long as the parameters are set correctly.
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write or reload must be completed before a write access can be requested.
3. Write the requested to HW_OCOTP_CTRL[ADDR] and program the unlock code into HW_OCOTP_CTRL[WR_UNLOCK]. This must be programmed for each write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation.
4. Write the data to the HW_OCOTP_DATA register. This will automatically set HW_OCOTP_CTRL[BUSY] and clear HW_OCOTP_CTRL[WR_UNLOCK]. To protect programming same OTP bit twice, before program OCOTP will automatically read fuse value in OTP and use read value to mask program data. The controller will use masked program data to program a 32-bit word in the OTP per the address in HW_OCOTP_CTRL[ADDR]. Bit fields with 1's will result in that OTP bit being programmed. Bit fields with 0's will be ignored. At the same time that the write is accepted, the controller makes an internal copy of HW_OCOTP_CTRL[ADDR] which cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to HW_OCOTP_CTRL[ADDR] will not affect an active write operation. It should also be noted that during the programming HW_OCOTP_DATA will shift right (with zero fill). This shifting is required to program the OTP serially. During the write operation, HW_OCOTP_DATA cannot be modified.
5. Once complete, the controller will clear BUSY. A write request to a protected or locked region will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are numbers of 10 micro-seconds. Write latencies is based on amount of bit filed which is 1. For example : program half fuse bits in one word need 10 us x 16.

For further details of OTP read/write operations see [eFUSE].

HW_OCOTP_CTRL[ERROR] will be set under the following conditions:

- A write is performed to a shadow register during a shadow reload (essentially, while HW_OCOTP_CTRL[RELOAD_SHADOWS] is set. In addition, the contents of the shadow register shall not be updated.
- A write is performed to a shadow register which has been locked.
- A read is performed to from a shadow register which has been read locked.
- A program is performed to a fuse which has been .
- A read is performed to from a fuse which has been read locked.

35.3.1.4 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 us after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes.

A recommended software sequence to meet the postamble requirements is as follows:

- Issue the write and poll for BUSY (as per Fuse Shadow Memory Footprint).
- Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 us.
- Perform the next OTP operation.

35.3.2 Fuse Shadow Memory Footprint

The OTP memory footprint shows in the following figure. The registers are grouped by lock region. Their names correspond to the PIO register and fusemap names.

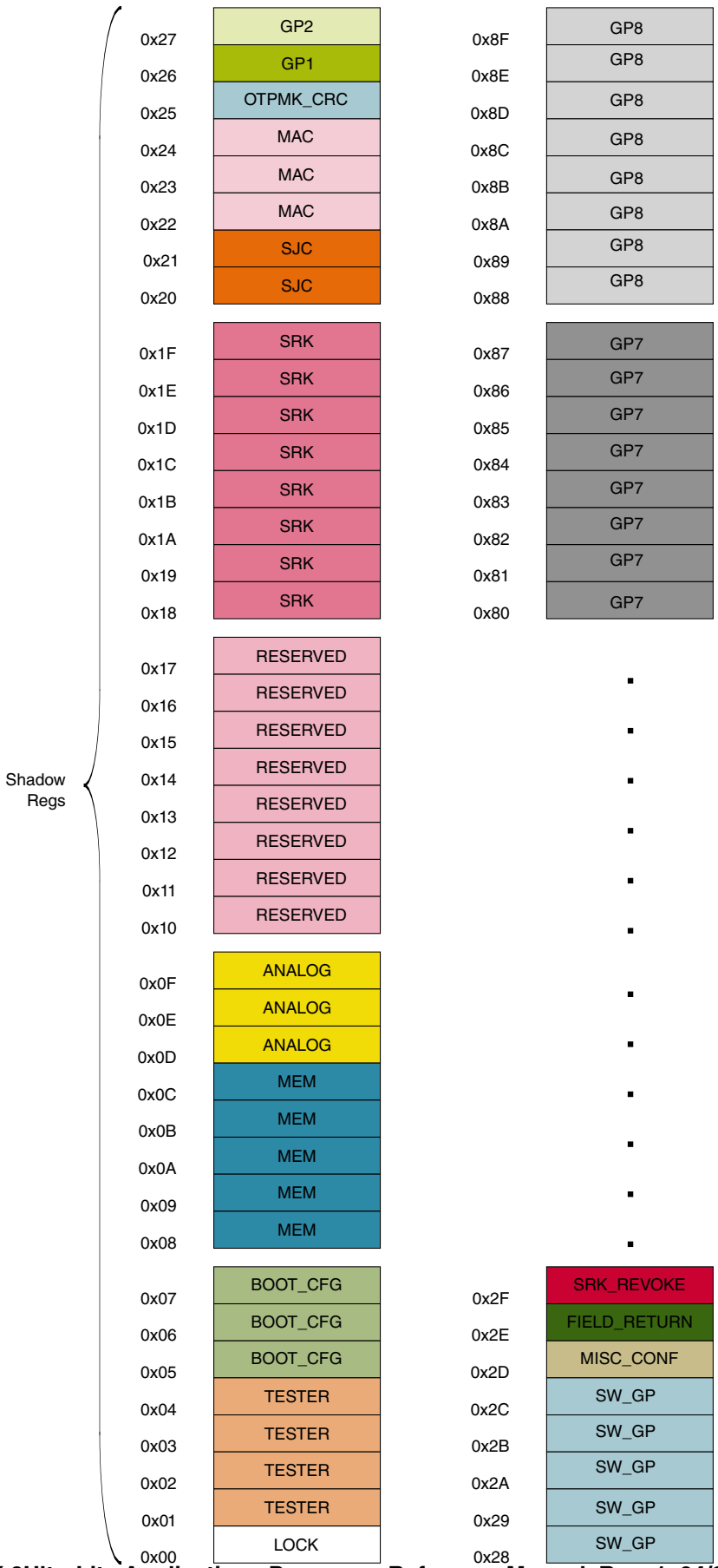


Figure 35-2. OTP Memory Footprint

35.3.3 OTP Read/Write Timing Parameters

There are three timing fields contained in the HW_OCOTP_TIMING register that specify counter limit values, which are used to specify the signal timing.

Both two timing parameters are specified in ipg_clk cycles. Since the ipg_clk frequency can be set to a range of values, these parameters must be adjusted with the clock to yield the appropriate delay.

35.3.4 Hardware Visible Fuses

The hwv_fuse bus emanates from the OCOTP block and goes to various other blocks inside the chip. This bus is made up of the shadow register bits for .

Only a subset of these fuse bits are currently used by the hardware. The fuse bits are initially copied from the banks after reset is deasserted. When all fuse bits are loaded into their shadow registers, the OCOTP asserts the fuse_latched output signal.

The hwv_reg bus also comes from the OCOTP. Its source is the HW_OCOTP_SCS register. This register has 1 defined bit, the HAB_JDE bit, that is connected to the SJC block. The SCS bits are intended to be used as volatile fuse bits under software control. Additional bits will be defined as needed in future implementations.

The system-wide reset sequence must be coordinated by the system reset controller, so that the hwv_fuse and hwv_reg buses are stable and reflect the values of the fuses before they are used by the rest of the system.

35.3.5 Behavior During Reset

The OCOTP is always active. The shadow registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is similar to that of a "reload shadow" operation.

35.3.6 Secure JTAG control

The JTAG control fuses are used to allow or disallow JTAG access to secured resources.

Three JTAG security levels are envisioned, as shown in the table below.

Table 35-2. JTAG Security Level Control Bits

Security Mode	JTAG_SMODE	Description
No Debug	2'b11	The highest security level.
Secure JTAG	2'b01	Limit the JTAG access by using key based authentication mechanism.
JTAG Enable	2'b00	Low Security, all JTAG features are enabled.

35.4 Fuse Map

See the Fusemap chapter of this reference manual for more information.

35.5 OCOTP Memory Map/Register Definition

OCOTP Hardware Register Format Summary

OCOTP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C000	OTP Controller Control Register (OCOTP_CTRL)	32	R/W	0000_0000h	35.5.1/2190
21B_C004	OTP Controller Control Register (OCOTP_CTRL_SET)	32	R/W	0000_0000h	35.5.1/2190
21B_C008	OTP Controller Control Register (OCOTP_CTRL_CLR)	32	R/W	0000_0000h	35.5.1/2190
21B_C00C	OTP Controller Control Register (OCOTP_CTRL_TOG)	32	R/W	0000_0000h	35.5.1/2190
21B_C010	OTP Controller Timing Register (OCOTP_TIMING)	32	R/W	0146_1299h	35.5.2/2192
21B_C020	OTP Controller Write Data Register (OCOTP_DATA)	32	R/W	0000_0000h	35.5.3/2193
21B_C030	OTP Controller Write Data Register (OCOTP_READ_CTRL)	32	R/W	0000_0000h	35.5.4/2193
21B_C040	OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA)	32	R/W	0000_0000h	35.5.5/2194
21B_C050	Sticky bit Register (OCOTP_SW_STICKY)	32	R/W	0000_0000h	35.5.6/2195
21B_C060	Software Controllable Signals Register (OCOTP_SCS)	32	R/W	0000_0000h	35.5.7/2196
21B_C064	Software Controllable Signals Register (OCOTP_SCS_SET)	32	R/W	0000_0000h	35.5.7/2196
21B_C068	Software Controllable Signals Register (OCOTP_SCS_CLR)	32	R/W	0000_0000h	35.5.7/2196
21B_C06C	Software Controllable Signals Register (OCOTP_SCS_TOG)	32	R/W	0000_0000h	35.5.7/2196
21B_C070	OTP Controller CRC test address (OCOTP_CRC_ADDR)	32	R/W	0000_0000h	35.5.8/2197
21B_C080	OTP Controller CRC Value Register (OCOTP_CRC_VALUE)	32	R/W	0000_0000h	35.5.9/2198

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C090	OTP Controller Version Register (OCOTP_VERSION)	32	R/W	0300_0000h	35.5.10/ 2198
21B_C400	Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)	32	R	0000_0000h	35.5.11/ 2199
21B_C410	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)	32	R/W	0000_0000h	35.5.12/ 2201
21B_C420	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)	32	R/W	0000_0000h	35.5.13/ 2202
21B_C430	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)	32	R/W	0000_0000h	35.5.14/ 2202
21B_C440	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)	32	R/W	0000_0000h	35.5.15/ 2203
21B_C450	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)	32	R/W	0000_0000h	35.5.16/ 2203
21B_C460	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)	32	R/W	0000_0000h	35.5.17/ 2204
21B_C470	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)	32	R/W	0000_0000h	35.5.18/ 2204
21B_C480	Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)	32	R/W	0000_0000h	35.5.19/ 2205
21B_C490	Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)	32	R/W	0000_0000h	35.5.20/ 2205
21B_C4A0	Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)	32	R/W	0000_0000h	35.5.21/ 2206
21B_C4B0	Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3)	32	R/W	0000_0000h	35.5.22/ 2206
21B_C4C0	Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4)	32	R/W	0000_0000h	35.5.23/ 2207
21B_C4D0	Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_ANA0)	32	R/W	0000_0000h	35.5.24/ 2207
21B_C4E0	Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_ANA1)	32	R/W	0000_0000h	35.5.25/ 2208
21B_C4F0	Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_ANA2)	32	R/W	0000_0000h	35.5.26/ 2208
21B_C500	Value of OTP Bank2 Word0 (OTPMK Key) (OCOTP_OTPMK0)	32	R/W	0000_0000h	35.5.27/ 2209
21B_C510	Value of OTP Bank2 Word1 (OTPMK Key) (OCOTP_OTPMK1)	32	R/W	0000_0000h	35.5.28/ 2209
21B_C520	Value of OTP Bank2 Word2 (OTPMK Key) (OCOTP_OTPMK2)	32	R/W	0000_0000h	35.5.29/ 2210
21B_C530	Value of OTP Bank2 Word3 (OTPMK Key) (OCOTP_OTPMK3)	32	R/W	0000_0000h	35.5.30/ 2210
21B_C540	Value of OTP Bank2 Word4 (OTPMK Key) (OCOTP_OTPMK4)	32	R/W	0000_0000h	35.5.31/ 2211

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C550	Value of OTP Bank2 Word5 (OTPMK Key) (OCOTP_OTPMK5)	32	R/W	0000_0000h	35.5.32/ 2211
21B_C560	Value of OTP Bank2 Word6 (OTPMK Key) (OCOTP_OTPMK6)	32	R/W	0000_0000h	35.5.33/ 2212
21B_C570	Value of OTP Bank2 Word7 (OTPMK Key) (OCOTP_OTPMK7)	32	R/W	0000_0000h	35.5.34/ 2212
21B_C580	Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0)	32	R/W	0000_0000h	35.5.35/ 2213
21B_C590	Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1)	32	R/W	0000_0000h	35.5.36/ 2213
21B_C5A0	Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2)	32	R/W	0000_0000h	35.5.37/ 2214
21B_C5B0	Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3)	32	R/W	0000_0000h	35.5.38/ 2214
21B_C5C0	Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4)	32	R/W	0000_0000h	35.5.39/ 2215
21B_C5D0	Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5)	32	R/W	0000_0000h	35.5.40/ 2215
21B_C5E0	Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6)	32	R/W	0000_0000h	35.5.41/ 2216
21B_C5F0	Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7)	32	R/W	0000_0000h	35.5.42/ 2216
21B_C600	Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0)	32	R/W	0000_0000h	35.5.43/ 2217
21B_C610	Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1)	32	R/W	0000_0000h	35.5.44/ 2217
21B_C620	Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0)	32	R/W	0000_0000h	35.5.45/ 2218
21B_C630	Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1)	32	R/W	0000_0000h	35.5.46/ 2218
21B_C640	Value of OTP Bank4 Word4 (MAC Address) (OCOTP_RESERVED) (OCOTP_MAC)	32	R/W	0000_0000h	35.5.47/ 2218
21B_C650	Value of OTP Bank4 Word5 (CRC Key) (OCOTP_CRC)	32	R/W	0000_0000h	35.5.48/ 2219
21B_C660	Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (OCOTP_GP1)	32	R/W	0000_0000h	35.5.49/ 2219
21B_C670	Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (OCOTP_GP2)	32	R/W	0000_0000h	35.5.50/ 2220
21B_C680	Value of OTP Bank5 Word0 (SW GP) (OCOTP_SW_GP0)	32	R/W	0000_0000h	35.5.51/ 2220
21B_C690	Value of OTP Bank5 Word1 (SW GP) (OCOTP_SW_GP1)	32	R/W	0000_0000h	35.5.52/ 2221
21B_C6A0	Value of OTP Bank5 Word2 (SW GP) (OCOTP_SW_GP2)	32	R/W	0000_0000h	35.5.53/ 2221

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C6B0	Value of OTP Bank5 Word3 (SW GP) (OCOTP_SW_GP3)	32	R/W	0000_0000h	35.5.54/ 2221
21B_C6C0	Value of OTP Bank5 Word4 (SW GP) (OCOTP_SW_GP4)	32	R/W	0000_0000h	35.5.55/ 2222
21B_C6D0	Value of OTP Bank5 Word5 (Misc Conf) (OCOTP_MISC_CONF)	32	R/W	0000_0000h	35.5.56/ 2222
21B_C6E0	Value of OTP Bank5 Word6 (Field Return) (OCOTP_FIELD_RETURN)	32	R/W	0000_0000h	35.5.57/ 2223
21B_C6F0	Value of OTP Bank5 Word7 (SRK Revoke) (OCOTP_SRK_REVOKE)	32	R/W	0000_0000h	35.5.58/ 2223
21B_C800	Value of OTP Bank6 Word0 (ROM Patch) (OCOTP_ROM_PATCH0)	32	R/W	0000_0000h	35.5.59/ 2224
21B_C810	Value of OTP Bank6 Word1 (ROM Patch) (OCOTP_ROM_PATCH1)	32	R/W	0000_0000h	35.5.60/ 2224
21B_C820	Value of OTP Bank6 Word2 (ROM Patch) (OCOTP_ROM_PATCH2)	32	R/W	0000_0000h	35.5.61/ 2224
21B_C830	Value of OTP Bank6 Word3 (ROM Patch) (OCOTP_ROM_PATCH3)	32	R/W	0000_0000h	35.5.62/ 2225
21B_C840	Value of OTP Bank6 Word4 (ROM Patch) (OCOTP_ROM_PATCH4)	32	R/W	0000_0000h	35.5.63/ 2225
21B_C850	Value of OTP Bank6 Word5 (ROM Patch) (OCOTP_ROM_PATCH5)	32	R/W	0000_0000h	35.5.64/ 2226
21B_C860	Value of OTP Bank6 Word6 (ROM Patch) (OCOTP_ROM_PATCH6)	32	R/W	0000_0000h	35.5.65/ 2226
21B_C870	Value of OTP Bank6 Word7 (ROM Patch) (OCOTP_ROM_PATCH7)	32	R/W	0000_0000h	35.5.66/ 2227
21B_C880	Value of OTP Bank7 Word0 (ROM Patch) (OCOTP_ROM_PATCH8)	32	R/W	0000_0000h	35.5.67/ 2227
21B_C890	Value of OTP Bank7 Word1 (ROM Patch) (OCOTP_ROM_PATCH9)	32	R/W	0000_0000h	35.5.68/ 2227
21B_C8A0	Value of OTP Bank7 Word2 (ROM Patch) (OCOTP_ROM_PATCH10)	32	R/W	0000_0000h	35.5.69/ 2228
21B_C8B0	Value of OTP Bank7 Word3 (ROM Patch) (OCOTP_ROM_PATCH11)	32	R/W	0000_0000h	35.5.70/ 2228
21B_C8C0	Value of OTP Bank7 Word4 (ROM Patch) (OCOTP_ROM_PATCH12)	32	R/W	0000_0000h	35.5.71/ 2229
21B_C8D0	Value of OTP Bank7 Word5 (ROM Patch) (OCOTP_ROM_PATCH13)	32	R/W	0000_0000h	35.5.72/ 2229
21B_C8E0	Value of OTP Bank7 Word6 (ROM Patch) (OCOTP_ROM_PATCH14)	32	R/W	0000_0000h	35.5.73/ 2230
21B_C8F0	Value of OTP Bank7 Word7 (ROM Patch) (OCOTP_ROM_PATCH15)	32	R/W	0000_0000h	35.5.74/ 2230
21B_C900	Value of OTP Bank8 Word0 (General Purpose Customer Defined Info) (OCOTP_GP30)	32	R/W	0000_0000h	35.5.75/ 2230

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C910	Value of OTP Bank8 Word1 (General Purpose Customer Defined Info) (OCOTP_GP31)	32	R/W	0000_0000h	35.5.76/ 2231
21B_C920	Value of OTP Bank8 Word2 (General Purpose Customer Defined Info) (OCOTP_GP32)	32	R/W	0000_0000h	35.5.77/ 2231
21B_C930	Value of OTP Bank8 Word3 (General Purpose Customer Defined Info) (OCOTP_GP33)	32	R/W	0000_0000h	35.5.78/ 2232
21B_C940	Value of OTP Bank8 Word4 (General Purpose Customer Defined Info) (OCOTP_GP34)	32	R/W	0000_0000h	35.5.79/ 2232
21B_C950	Value of OTP Bank8 Word5 (General Purpose Customer Defined Info) (OCOTP_GP35)	32	R/W	0000_0000h	35.5.80/ 2233
21B_C960	Value of OTP Bank8 Word6 (General Purpose Customer Defined Info) (OCOTP_GP36)	32	R/W	0000_0000h	35.5.81/ 2233
21B_C970	Value of OTP Bank8 Word7 (General Purpose Customer Defined Info) (OCOTP_GP37)	32	R/W	0000_0000h	35.5.82/ 2233
21B_C980	Value of OTP Bank9 Word0 (General Purpose Customer Defined Info) (OCOTP_GP38)	32	R/W	0000_0000h	35.5.83/ 2234
21B_C990	Value of OTP Bank9 Word1 (General Purpose Customer Defined Info) (OCOTP_GP39)	32	R/W	0000_0000h	35.5.84/ 2234
21B_C9A0	Value of OTP Bank9 Word2 (General Purpose Customer Defined Info) (OCOTP_GP310)	32	R/W	0000_0000h	35.5.85/ 2235
21B_C9B0	Value of OTP Bank9 Word3 (General Purpose Customer Defined Info) (OCOTP_GP311)	32	R/W	0000_0000h	35.5.86/ 2235
21B_C9C0	Value of OTP Bank9 Word4 (General Purpose Customer Defined Info) (OCOTP_GP312)	32	R/W	0000_0000h	35.5.87/ 2236
21B_C9D0	Value of OTP Bank9 Word5 (General Purpose Customer Defined Info) (OCOTP_GP313)	32	R/W	0000_0000h	35.5.88/ 2236
21B_C9E0	Value of OTP Bank9 Word6 (General Purpose Customer Defined Info) (OCOTP_GP314)	32	R/W	0000_0000h	35.5.89/ 2236
21B_C9F0	Value of OTP Bank9 Word7 (General Purpose Customer Defined Info) (OCOTP_GP315)	32	R/W	0000_0000h	35.5.90/ 2237
21B_CA00	Value of OTP Bank10 Word0 (General Purpose Customer Defined Info) (OCOTP_GP40)	32	R/W	0000_0000h	35.5.91/ 2237
21B_CA10	Value of OTP Bank10 Word1 (General Purpose Customer Defined Info) (OCOTP_GP41)	32	R/W	0000_0000h	35.5.92/ 2238
21B_CA20	Value of OTP Bank10 Word2 (General Purpose Customer Defined Info) (OCOTP_GP42)	32	R/W	0000_0000h	35.5.93/ 2238
21B_CA30	Value of OTP Bank10 Word3 (General Purpose Customer Defined Info) (OCOTP_GP43)	32	R/W	0000_0000h	35.5.94/ 2239
21B_CA40	Value of OTP Bank10 Word4 (General Purpose Customer Defined Info) (OCOTP_GP44)	32	R/W	0000_0000h	35.5.95/ 2239
21B_CA50	Value of OTP Bank10 Word5 (General Purpose Customer Defined Info) (OCOTP_GP45)	32	R/W	0000_0000h	35.5.96/ 2239
21B_CA60	Value of OTP Bank10 Word6 (General Purpose Customer Defined Info) (OCOTP_GP46)	32	R/W	0000_0000h	35.5.97/ 2240

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_CA70	Value of OTP Bank10 Word7 (General Purpose Customer Defined Info) (OCOTP_GP47)	32	R/W	0000_0000h	35.5.98/ 2240
21B_CA80	Value of OTP Bank11 Word0 (General Purpose Customer Defined Info) (OCOTP_GP48)	32	R/W	0000_0000h	35.5.99/ 2241
21B_CA90	Value of OTP Bank11 Word1 (General Purpose Customer Defined Info) (OCOTP_GP49)	32	R/W	0000_0000h	35.5.100/ 2241
21B_CAA0	Value of OTP Bank11 Word2 (General Purpose Customer Defined Info) (OCOTP_GP410)	32	R/W	0000_0000h	35.5.101/ 2242
21B_CAB0	Value of OTP Bank11 Word3 (General Purpose Customer Defined Info) (OCOTP_GP411)	32	R/W	0000_0000h	35.5.102/ 2242
21B_CAC0	Value of OTP Bank11 Word4 (General Purpose Customer Defined Info) (OCOTP_GP412)	32	R/W	0000_0000h	35.5.103/ 2242
21B_CAD0	Value of OTP Bank11 Word5 (General Purpose Customer Defined Info) (OCOTP_GP413)	32	R/W	0000_0000h	35.5.104/ 2243
21B_CAE0	Value of OTP Bank11 Word6 (General Purpose Customer Defined Info) (OCOTP_GP414)	32	R/W	0000_0000h	35.5.105/ 2243
21B_CAF0	Value of OTP Bank11 Word7 (General Purpose Customer Defined Info) (OCOTP_GP415)	32	R/W	0000_0000h	35.5.106/ 2244
21B_CB00	Value of OTP Bank12 Word0 (General Purpose Customer Defined Info) (OCOTP_GP50)	32	R/W	0000_0000h	35.5.107/ 2244
21B_CB10	Value of OTP Bank12 Word1 (General Purpose Customer Defined Info) (OCOTP_GP51)	32	R/W	0000_0000h	35.5.108/ 2245
21B_CB20	Value of OTP Bank12 Word2 (General Purpose Customer Defined Info) (OCOTP_GP52)	32	R/W	0000_0000h	35.5.109/ 2245
21B_CB30	Value of OTP Bank12 Word3 (General Purpose Customer Defined Info) (OCOTP_GP53)	32	R/W	0000_0000h	35.5.110/ 2245
21B_CB40	Value of OTP Bank12 Word4 (General Purpose Customer Defined Info) (OCOTP_GP54)	32	R/W	0000_0000h	35.5.111/ 2246
21B_CB50	Value of OTP Bank12 Word5 (General Purpose Customer Defined Info) (OCOTP_GP55)	32	R/W	0000_0000h	35.5.112/ 2246
21B_CB60	Value of OTP Bank12 Word6 (General Purpose Customer Defined Info) (OCOTP_GP56)	32	R/W	0000_0000h	35.5.113/ 2247
21B_CB70	Value of OTP Bank12 Word7 (General Purpose Customer Defined Info) (OCOTP_GP57)	32	R/W	0000_0000h	35.5.114/ 2247
21B_CB80	Value of OTP Bank13 Word0 (General Purpose Customer Defined Info) (OCOTP_GP58)	32	R/W	0000_0000h	35.5.115/ 2248
21B_CB90	Value of OTP Bank13 Word1 (General Purpose Customer Defined Info) (OCOTP_GP59)	32	R/W	0000_0000h	35.5.116/ 2248
21B_CBA0	Value of OTP Bank13 Word2 (General Purpose Customer Defined Info) (OCOTP_GP510)	32	R/W	0000_0000h	35.5.117/ 2248
21B_CBB0	Value of OTP Bank13 Word3 (General Purpose Customer Defined Info) (OCOTP_GP511)	32	R/W	0000_0000h	35.5.118/ 2249
21B_CBC0	Value of OTP Bank13 Word4 (General Purpose Customer Defined Info) (OCOTP_GP512)	32	R/W	0000_0000h	35.5.119/ 2249

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_CBD0	Value of OTP Bank13 Word5 (General Purpose Customer Defined Info) (OCOTP_GP513)	32	R/W	0000_0000h	35.5.120/ 2250
21B_CBE0	Value of OTP Bank13 Word6 (General Purpose Customer Defined Info) (OCOTP_GP514)	32	R/W	0000_0000h	35.5.121/ 2250
21B_CBF0	Value of OTP Bank13 Word7 (General Purpose Customer Defined Info) (OCOTP_GP515)	32	R/W	0000_0000h	35.5.122/ 2251
21B_CC00	Value of OTP Bank14 Word0 (General Purpose Customer Defined Info) (OCOTP_GP60)	32	R/W	0000_0000h	35.5.123/ 2251
21B_CC10	Value of OTP Bank14 Word1 (General Purpose Customer Defined Info) (OCOTP_GP61)	32	R/W	0000_0000h	35.5.124/ 2251
21B_CC20	Value of OTP Bank14 Word2 (General Purpose Customer Defined Info) (OCOTP_GP62)	32	R/W	0000_0000h	35.5.125/ 2252
21B_CC30	Value of OTP Bank14 Word3 (General Purpose Customer Defined Info) (OCOTP_GP63)	32	R/W	0000_0000h	35.5.126/ 2252
21B_CC40	Value of OTP Bank14 Word4 (General Purpose Customer Defined Info) (OCOTP_GP64)	32	R/W	0000_0000h	35.5.127/ 2253
21B_CC50	Value of OTP Bank14 Word5 (General Purpose Customer Defined Info) (OCOTP_GP65)	32	R/W	0000_0000h	35.5.128/ 2253
21B_CC60	Value of OTP Bank14 Word6 (General Purpose Customer Defined Info) (OCOTP_GP66)	32	R/W	0000_0000h	35.5.129/ 2254
21B_CC70	Value of OTP Bank14 Word7 (General Purpose Customer Defined Info) (OCOTP_GP67)	32	R/W	0000_0000h	35.5.130/ 2254
21B_CC80	Value of OTP Bank15 Word0 (General Purpose Customer Defined Info) (OCOTP_GP70)	32	R/W	0000_0000h	35.5.131/ 2254
21B_CC90	Value of OTP Bank15 Word1 (General Purpose Customer Defined Info) (OCOTP_GP71)	32	R/W	0000_0000h	35.5.132/ 2255
21B_CCA0	Value of OTP Bank15 Word2 (General Purpose Customer Defined Info) (OCOTP_GP72)	32	R/W	0000_0000h	35.5.133/ 2255
21B_CCB0	Value of OTP Bank15 Word3 (General Purpose Customer Defined Info) (OCOTP_GP73)	32	R/W	0000_0000h	35.5.134/ 2256
21B_CCC0	Value of OTP Bank15 Word4 (General Purpose Customer Defined Info) (OCOTP_GP80)	32	R/W	0000_0000h	35.5.135/ 2256
21B_CCD0	Value of OTP Bank15 Word5 (General Purpose Customer Defined Info) (OCOTP_GP81)	32	R/W	0000_0000h	35.5.136/ 2257
21B_CCE0	Value of OTP Bank15 Word6 (General Purpose Customer Defined Info) (OCOTP_GP82)	32	R/W	0000_0000h	35.5.137/ 2257
21B_CCF0	Value of OTP Bank15 Word7 (General Purpose Customer Defined Info) (OCOTP_GP83)	32	R/W	0000_0000h	35.5.138/ 2257

35.5.1 OTP Controller Control Register (OCOTP_CTRL*n*)

The OCOTP Control and Status Register specifies the copy state, as well as the control required for random access of the OTP memory

OCOTP_CTRL: 0x000

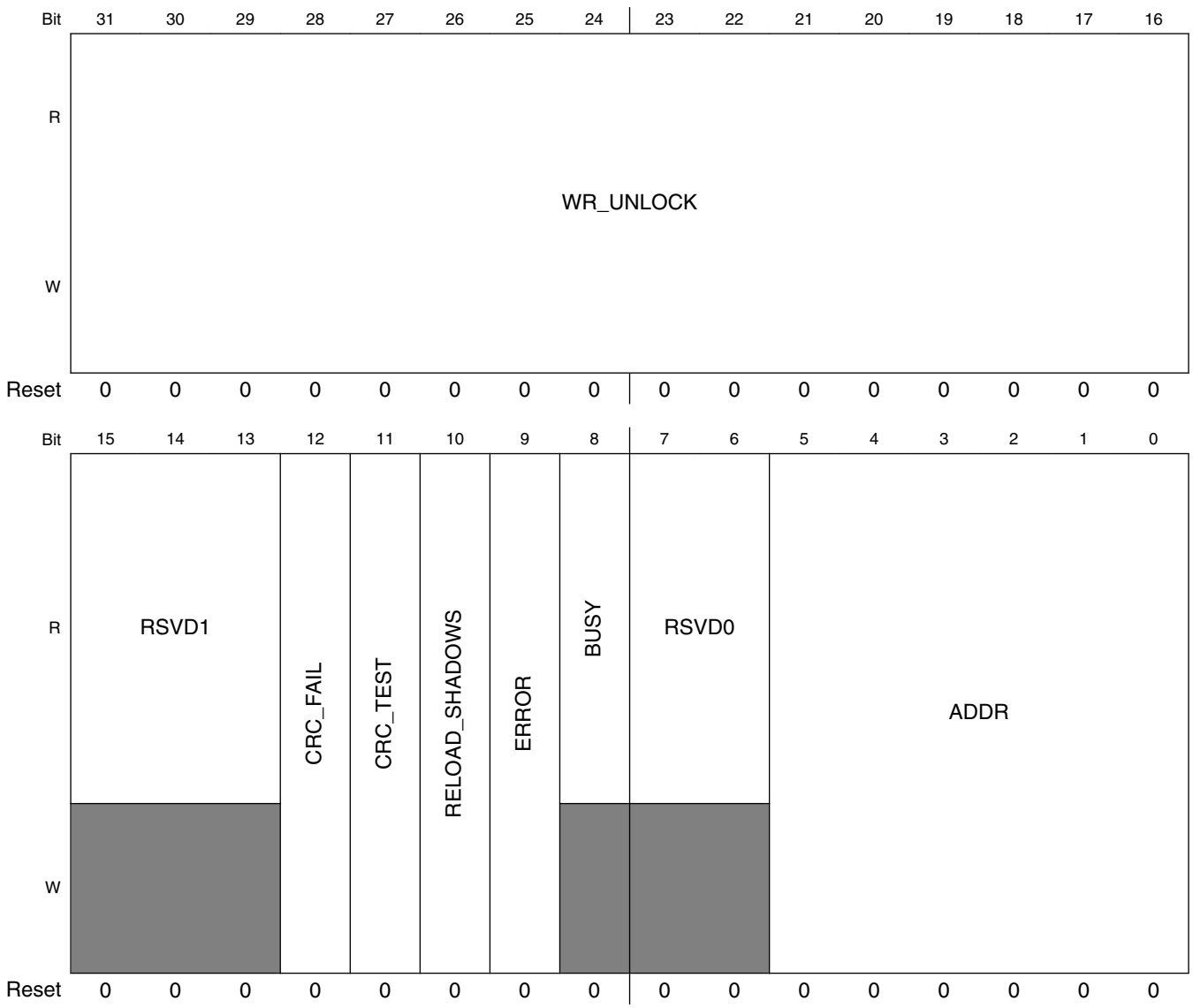
OCOTP_CTRL_SET: 0x004

OCOTP_CTRL_CLR: 0x008

OCOTP_CTRL_TOG: 0x00C

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjuction with the OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and OCOTP_READ_CTRL register. Read value is saved in OCOTP_READ_FUSE_DATA register.

Address: 21B_C000h base + 0h offset + (4d × i), where i=0d to 3d



OCOTP_CTRL*n* field descriptions

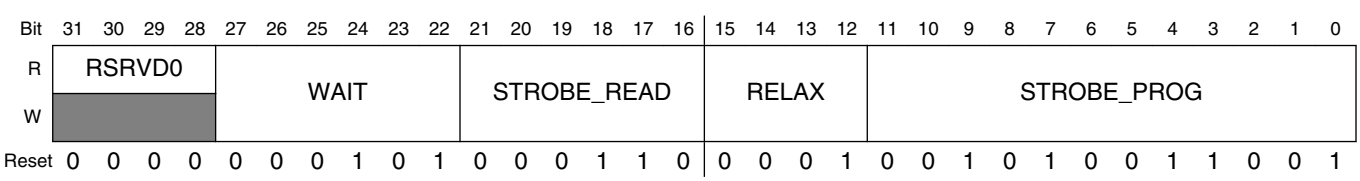
Field	Description
31–16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY). 0x3E77 **KEY** — Key needed to unlock HW_OCOTP_DATA register.
15–13 RSVD1	Reserved
12 CRC_FAIL	Set by controller when calculated CRC value is not equal to appointed CRC fuse word
11 CRC_TEST	Set to calculate CRC according to start address and end address in CRC_ADDR register.And compare with CRC fuse word according CRC address in CRC_ADDR register to generate CRC_FAIL flag
10 RELOAD_ SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7–6 RSVD0	Reserved
ADDR	OTP write and read access address register. Specifies one of 64 word address locations (0x00 - 0x3f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

35.5.2 OTP Controller Timing Register (OCOTP_TIMING)

The OCOTP Data Register is used for OTP Programming

This register specifies timing parameters for programming and reading the OCOTP fuse array.

Address: 21B_C000h base + 10h offset = 21B_C010h



OCOTP_TIMING field descriptions

Field	Description
31–28 RSRVD0	These bits always read back zero.
27–22 WAIT	This count value specifies time interval between auto read and write access in one time program. It is given in number of ipg_clk periods.
21–16 STROBE_READ	This count value specifies the strobe period in one time read OTP. Trd = ((STROBE_READ+1)- 2 * (RELAX + 1)) / ipg_clk_freq. It is given in number of IPG_CLK periods.
15–12 RELAX	This count value specifies the time to add to all default timing parameters other than the Tpgm and Trd. It is given in number of ipg_clk periods.
STROBE_PROG	This count value specifies the strobe period in one time write OTP. Tpgm = ((STROBE_PROG+1)- 2 * (RELAX + 1)) / ipg_clk_freq. It is given in number of ipg_clk periods.

35.5.3 OTP Controller Write Data Register (OCOTP_DATA)

The OCOTP Data Register is used for OTP Programming

This register is used in conjuction with OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

Address: 21B_C000h base + 20h offset = 21B_C020h

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

OCOTP_DATA field descriptions

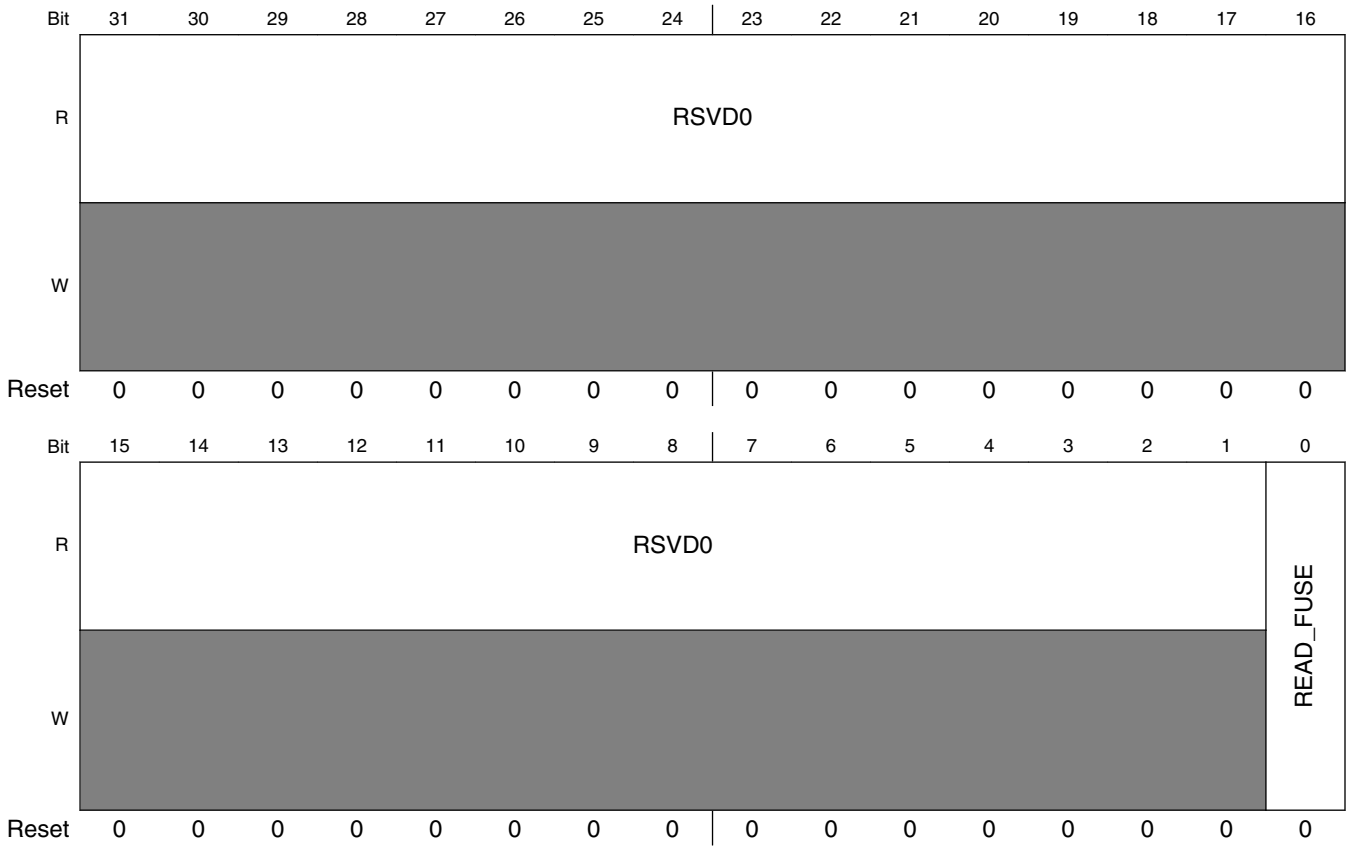
Field	Description
DATA	Used to initiate a write to OTP. Please see the "Software Write Sequence" section for operating details.

35.5.4 OTP Controller Write Data Register (OCOTP_READ_CTRL)

The OCOTP Register is used for OTP Read

This register is used in conjuction with OCOTP_CTRL to perform one time read to the OTP. Please see the "Software read Sequence" section for operating details.

Address: 21B_C000h base + 30h offset = 21B_C030h



OCOTP_READ_CTRL field descriptions

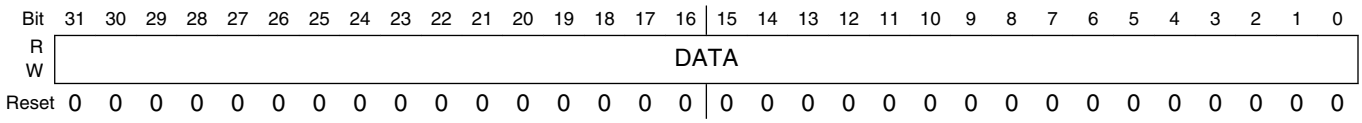
Field	Description
31–1 RSVD0	Reserved
0 READ_FUSE	Used to initiate a read to OTP. Please see the "Software read Sequence" section for operating details.

35.5.5 OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA)

The OCOTP Data Register is used for OTP Read

The data read from OTP

Address: 21B_C000h base + 40h offset = 21B_C040h



OCOTP_READ_FUSE_DATA field descriptions

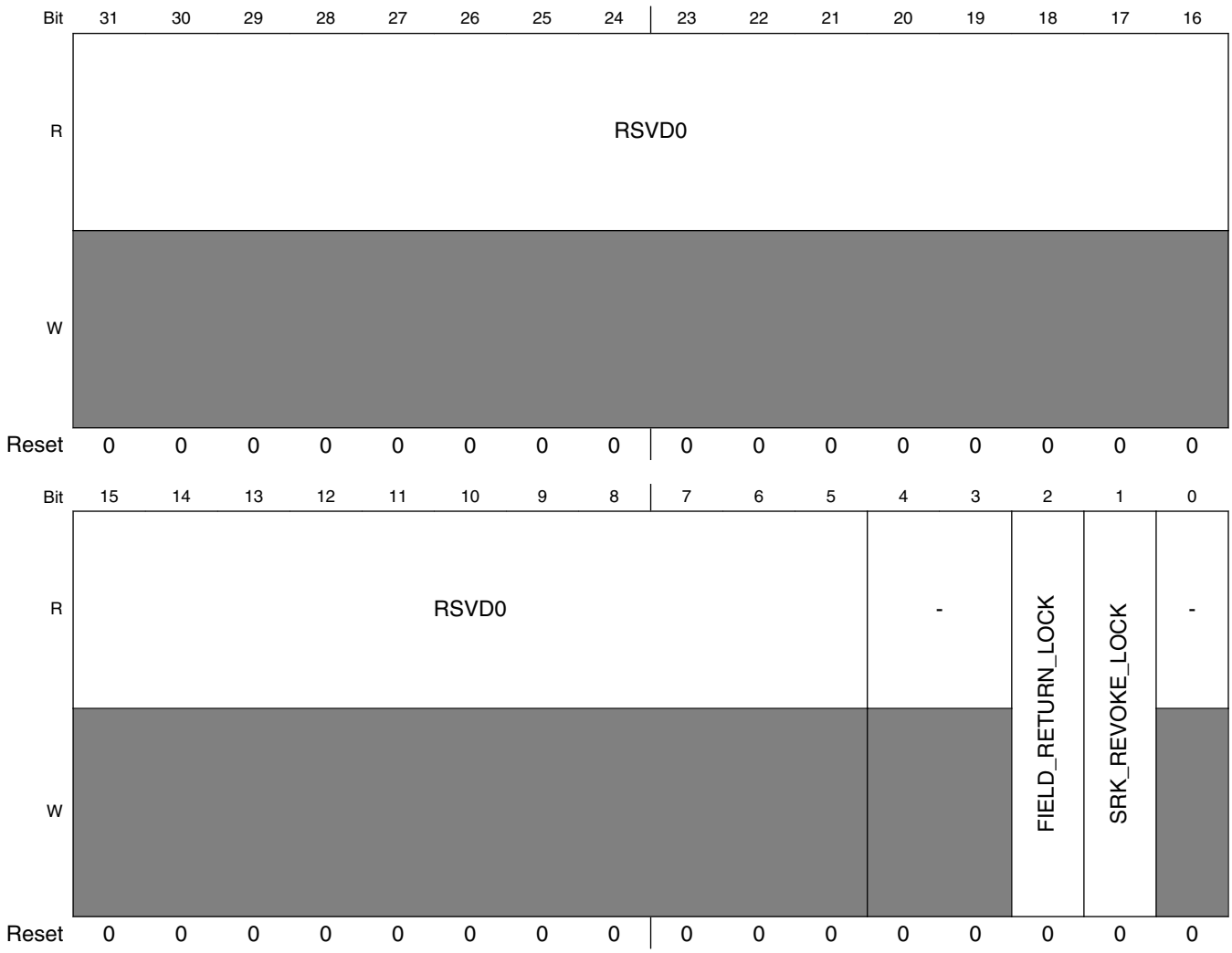
Field	Description
DATA	The data read from OTP

35.5.6 Sticky bit Register (OCOTP_SW_STICKY)

Some SW sticky bits .

Some sticky bits are used by SW to lock some fuse area, shadow registers and other features.

Address: 21B_C000h base + 50h offset = 21B_C050h



OCOTP_SW_STICKY field descriptions

Field	Description
31–5 RSVD0	Reserved
4–3 -	Reserved
2 FIELD_ RETURN_LOCK	Shadow register write and OTP write lock for FIELD_RETURN region. When set, the writing of this region's shadow register and OTP fuse word are blocked.Once this bit is set, it is always high unless a POR is issued.
1 SRK_REVOKE_ LOCK	Shadow register write and OTP write lock for SRK_REVOKE region. When set, the writing of this region's shadow register and OTP fuse word are blocked. Once this bit is set, it is always high unless a POR is issued.
0 -	Reserved

35.5.7 Software Controllable Signals Register (OCOTP_SCS*n*)

OCOTP_SCS: 0x060

OCOTP_SCS_SET: 0x064

OCOTP_SCS_CLR: 0x068

OCOTP_SCS_TOG: 0x06C

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

Address: 21B_C000h base + 60h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LOCK	SPARE														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

OCOTP_SCS*n* field descriptions

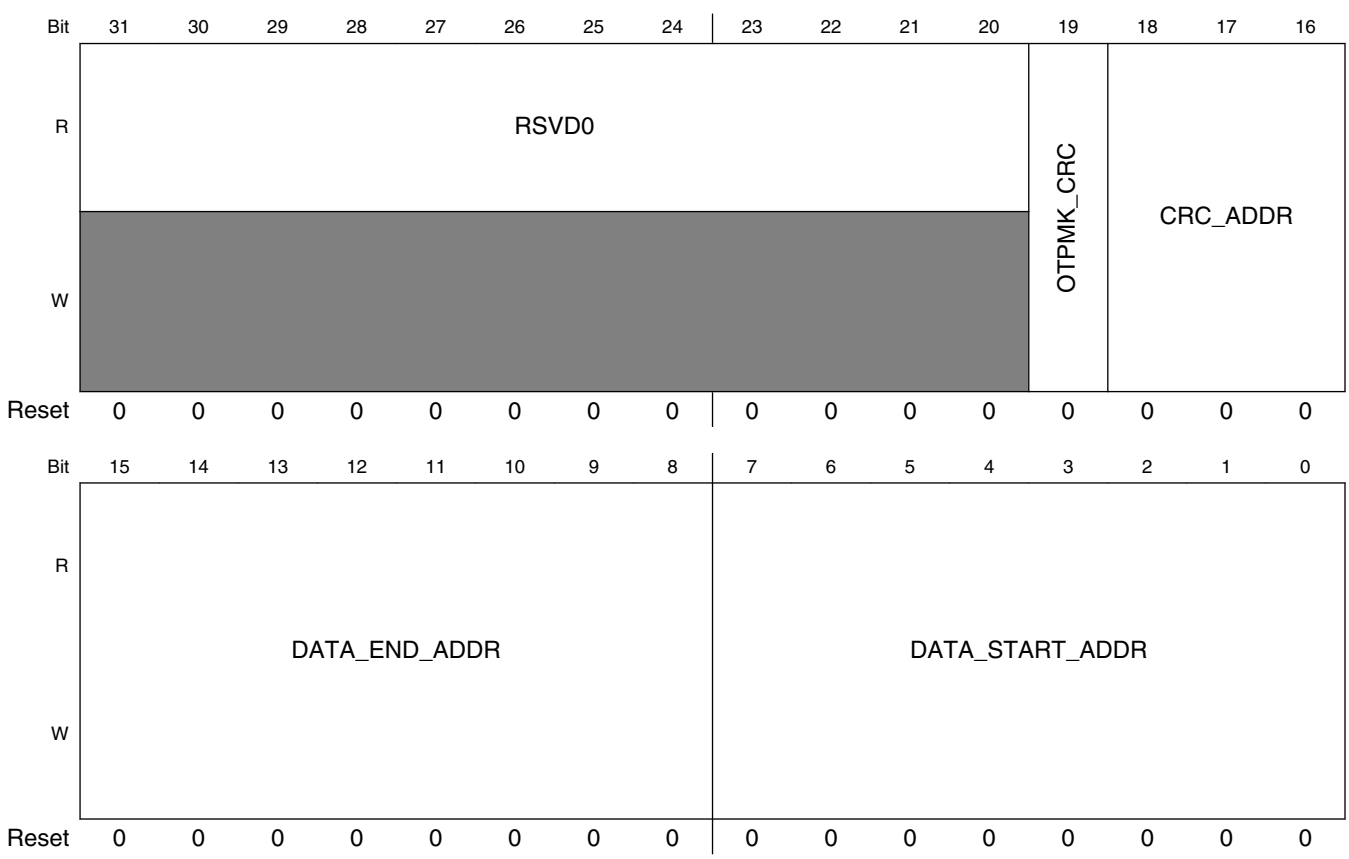
Field	Description
31 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30–1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

35.5.8 OTP Controller CRC test address (OCOTP_CRC_ADDR)

The OCOTP Data Register is used for OTP Read

The address for CRC calculation

Address: 21B_C000h base + 70h offset = 21B_C070h



OCOTP_CRC_ADDR field descriptions

Field	Description
31–20 RSVD0	Reserved
19 OTPMK_CRC	Enable bit for CRC32 calculation address When OTPMK_CRC_ADDR_OTPMK_CRC bit sets to 1, calculation address sets to OTPMK_CRC (recommend). Otherwise address sets to GP7 or GP8.
18–16 CRC_ADDR	Address of 32-bit CRC result for comparing
15–8 DATA_END_ADDR	Start address of fuse location for CRC calculation
DATA_START_ADDR	End address of fuse location for CRC calculation

35.5.9 OTP Controller CRC Value Register (OCOTP_CRC_VALUE)

The OCOTP Data Register is used for OTP Read

The CRC32 value is based on CRC_ADDR.

Address: 21B_C000h base + 80h offset = 21B_C080h

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

OCOTP_CRC_VALUE field descriptions

Field	Description
DATA	The crc32 value based on CRC_ADDR

35.5.10 OTP Controller Version Register (OCOTP_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

Address: 21B_C000h base + 90h offset = 21B_C090h

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

OCOTP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

35.5.11 Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

Address: 21B_C000h base + 400h offset = 21B_C400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GP6		GP8		GP7		PIN	GP5	GP4	MISC_CONF	ROM_PATCH	OTPMK_CRC	ANALOG		OTPMK	SW_GP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GP3	SRK	GP2		GP1		MAC_ADDR		RSVD0	SJC_RESP	MEM_TRIM		BOOT_CFG		TESTER	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_LOCK field descriptions

Field	Description
31–30 GP6	Status of shadow register and OTP write lock for GP6 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
29–28 GP8	Status of shadow register and OTP write lock for gp8 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked. 27-26 GP7 Status of shadow register and OTP write lock for GP7 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
27–26 GP7	Status of shadow register and OTP write lock for GP7 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
25 PIN	Status of Pin access lock bit. When set, pin access is disabled.
24 GP5	Status of shadow register and OTP write lock for GP5 region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
23 GP4	Status of shadow register and OTP write lock for GP4 region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
22 MISC_CONF	Status of shadow register and OTP write lock for misc_conf region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
21 ROM_PATCH	Status of shadow register and OTP write lock for rom_patch region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
20 OTPMK_CRC	Status of shadow register and OTP write lock for otpmk crc region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
19–18 ANALOG	Status of shadow register and OTP write lock for analog region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
17 OTPMK	Status of shadow register and OTP write lock for OTPMK region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
16 SW_GP	Status of shadow register and OTP write lock for SW_GP region. When set, the writing of this region's shadow register and OTP fuse word are blocked.

Table continues on the next page...

OCOTP_LOCK field descriptions (continued)

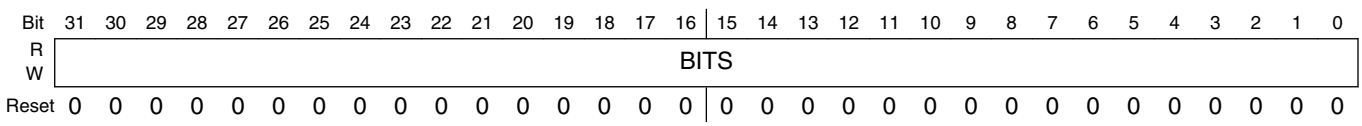
Field	Description
15 GP3	Status of shadow register and OTP write lock for GP3 region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
14 SRK	Status of shadow register and OTP write lock for srk region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
13–12 GP2	Status of shadow register and OTP write lock for gp2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
11–10 GP1	Status of shadow register and OTP write lock for gp2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
9–8 MAC_ADDR	Status of shadow register and OTP write lock for mac_addr region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
7 RSVD0	Reserved
6 SJC_RESP	Status of shadow register read and write, OTP read and write lock for sjc_resp region. When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
5–4 MEM_TRIM	Status of shadow register and OTP write lock for mem_trim region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
3–2 BOOT_CFG	Status of shadow register and OTP write lock for boot_cfg region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
TESTER	Status of shadow register and OTP write lock for tester region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.

35.5.12 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

Address: 21B_C000h base + 410h offset = 21B_C410h



OCOTP_CFG0 field descriptions

Field	Description
BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 1 (ADDR = 0x01). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

35.5.13 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

Address: 21B_C000h base + 420h offset = 21B_C420h

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

OCOTP_CFG1 field descriptions

Field	Description
BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 2 (ADDR = 0x02). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

35.5.14 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

Address: 21B_C000h base + 430h offset = 21B_C430h

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

OCOTP_CFG2 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 3 (ADDR = 0x03). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

35.5.15 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

Address: 21B_C000h base + 440h offset = 21B_C440h

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

OCOTP_CFG3 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 4 (ADDR = 0x04). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

35.5.16 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

Address: 21B_C000h base + 450h offset = 21B_C450h

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

OCOTP_CFG4 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 5 (ADDR = 0x05). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

35.5.17 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

Address: 21B_C000h base + 460h offset = 21B_C460h

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

OCOTP_CFG5 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 6 (ADDR = 0x06). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

35.5.18 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

Address: 21B_C000h base + 470h offset = 21B_C470h

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

OCOTP_CFG6 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

35.5.19 Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 0 (ADDR = 0x08).

Address: 21B_C000h base + 480h offset = 21B_C480h

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

OCOTP_MEM0 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 0 (ADDR = 0x08). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

35.5.20 Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 1 (ADDR = 0x09).

Address: 21B_C000h base + 490h offset = 21B_C490h

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

OCOTP_MEM1 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 1 (ADDR = 0x09). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

35.5.21 Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 2 (ADDR = 0x0A).

Address: 21B_C000h base + 4A0h offset = 21B_C4A0h

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

OCOTP_MEM2 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 2 (ADDR = 0x0A). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

35.5.22 Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 3 (ADDR = 0x0B).

Address: 21B_C000h base + 4B0h offset = 21B_C4B0h

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

OCOTP_MEM3 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 3 (ADDR = 0x0B). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

35.5.23 Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 4 (ADDR = 0x0C).

Address: 21B_C000h base + 4C0h offset = 21B_C4C0h

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

OCOTP_MEM4 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 4 (ADDR = 0x0C). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

35.5.24 Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_ANA0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 5 (ADDR = 0x0D).

Address: 21B_C000h base + 4D0h offset = 21B_C4D0h

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

OCOTP_ANA0 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 5 (ADDR = 0x0D). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

35.5.25 Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_ANA1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 6 (ADDR = 0x0E).

Address: 21B_C000h base + 4E0h offset = 21B_C4E0h

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

OCOTP_ANA1 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 6 (ADDR = 0x0E). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

35.5.26 Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_ANA2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 7 (ADDR = 0x0F).

Address: 21B_C000h base + 4F0h offset = 21B_C4F0h

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

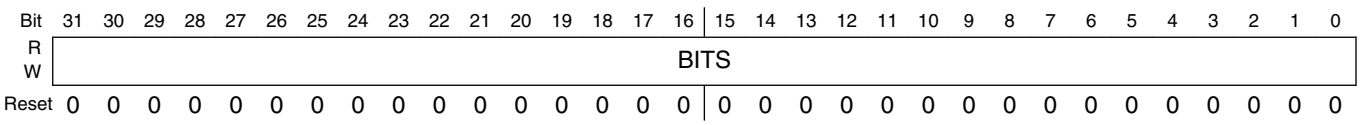
OCOTP_ANA2 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 7 (ADDR = 0x0F). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

35.5.27 Value of OTP Bank2 Word0 (OTPMK Key) (OCOTP_OTPMK0)

Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 0 (ADDR = 0x10)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

Address: 21B_C000h base + 500h offset = 21B_C500h



OCOTP_OTPMK0 field descriptions

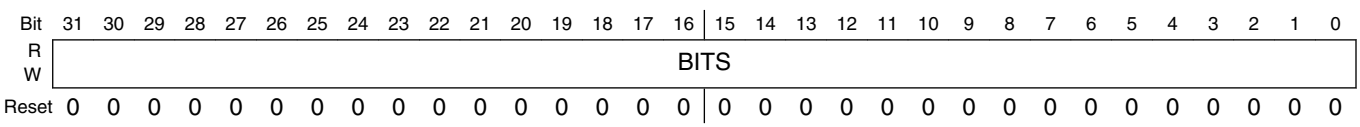
Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 0 (ADDR = 0x10)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.28 Value of OTP Bank2 Word1 (OTPMK Key) (OCOTP_OTPMK1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 1 (ADDR = 0x11).

Address: 21B_C000h base + 510h offset = 21B_C510h



OCOTP_OTPMK1 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 1 (ADDR = 0x11)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.29 Value of OTP Bank2 Word2 (OTPMK Key) (OCOTP_OTPMK2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 2 (ADDR = 0x12).

Address: 21B_C000h base + 520h offset = 21B_C520h

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

OCOTP_OTPMK2 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 2 (ADDR = 0x12)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.30 Value of OTP Bank2 Word3 (OTPMK Key) (OCOTP_OTPMK3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 3 (ADDR = 0x13).

Address: 21B_C000h base + 530h offset = 21B_C530h

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

OCOTP_OTPMK3 field descriptions

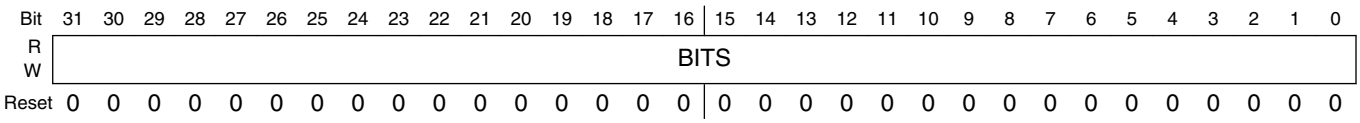
Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 3 (ADDR = 0x13)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.31 Value of OTP Bank2 Word4 (OTPMK Key) (OCOTP_OTPMK4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 4 (ADDR = 0x14).

Address: 21B_C000h base + 540h offset = 21B_C540h



OCOTP_OTPMK4 field descriptions

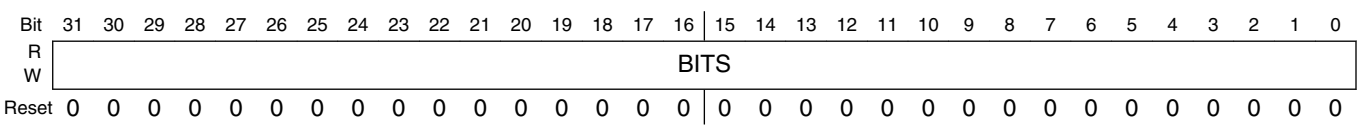
Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 4 (ADDR = 0x14)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.32 Value of OTP Bank2 Word5 (OTPMK Key) (OCOTP_OTPMK5)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 5 (ADDR = 0x15).

Address: 21B_C000h base + 550h offset = 21B_C550h



OCOTP_OTPMK5 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 4 (ADDR = 0x14)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.33 Value of OTP Bank2 Word6 (OTPMK Key) (OCOTP_OTPMK6)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 6 (ADDR = 0x16).

Address: 21B_C000h base + 560h offset = 21B_C560h

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

OCOTP_OTPMK6 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 6 (ADDR = 0x16)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.34 Value of OTP Bank2 Word7 (OTPMK Key) (OCOTP_OTPMK7)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 7 (ADDR = 0x17).

Address: 21B_C000h base + 570h offset = 21B_C570h

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

OCOTP_OTPMK7 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 2, word 7 (ADDR = 0x17)). These bits can not be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.35 Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 0 (ADDR = 0x18).

Address: 21B_C000h base + 580h offset = 21B_C580h

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

OCOTP_SRK0 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word0 (Copy of OTP Bank 3, word 0 (ADDR = 0x18)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.36 Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 1 (ADDR = 0x19).

Address: 21B_C000h base + 590h offset = 21B_C590h

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

OCOTP_SRK1 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word1 (Copy of OTP Bank 3, word 1 (ADDR = 0x19)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.37 Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 2 (ADDR = 0x1A).

Address: 21B_C000h base + 5A0h offset = 21B_C5A0h

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

OCOTP_SRK2 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word2 (Copy of OTP Bank 3, word 2 (ADDR = 0x1A)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.38 Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 3 (ADDR = 0x1B).

Address: 21B_C000h base + 5B0h offset = 21B_C5B0h

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

OCOTP_SRK3 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word3 (Copy of OTP Bank 3, word 3 (ADDR = 0x1B)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.39 Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 4 (ADDR = 0x1C).

Address: 21B_C000h base + 5C0h offset = 21B_C5C0h

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

OCOTP_SRK4 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word4 (Copy of OTP Bank 3, word 4 (ADDR = 0x1C)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.40 Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 5 (ADDR = 0x1D).

Address: 21B_C000h base + 5D0h offset = 21B_C5D0h

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

OCOTP_SRK5 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word5 (Copy of OTP Bank 3, word 5 (ADDR = 0x1D)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.41 Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 6 (ADDR = 0x1E).

Address: 21B_C000h base + 5E0h offset = 21B_C5E0h

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

OCOTP_SRK6 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word6 (Copy of OTP Bank 3, word 6 (ADDR = 0x1E)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.42 Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 7 (ADDR = 0x1F).

Address: 21B_C000h base + 5F0h offset = 21B_C5F0h

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

OCOTP_SRK7 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word7 (Copy of OTP Bank 3, word 7 (ADDR = 0x1F)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

35.5.43 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 0 (ADDR = 0x20).

Address: 21B_C000h base + 600h offset = 21B_C600h

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

OCOTP_SJC_RESP0 field descriptions

Field	Description
BITS	Shadow register for the SJC_RESP Key word0 (Copy of OTP Bank 4, word 0 (ADDR = 0x20)). These bits can be not read and wrotten after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.44 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 1 (ADDR = 0x21).

Address: 21B_C000h base + 610h offset = 21B_C610h

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

OCOTP_SJC_RESP1 field descriptions

Field	Description
BITS	Shadow register for the SJC_RESP Key word1 (Copy of OTP Bank 4, word 1 (ADDR = 0x21)). These bits can be not read and wrotten after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

35.5.45 Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 2 (ADDR = 0x22).

Address: 21B_C000h base + 620h offset = 21B_C620h

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

OCOTP_MAC0 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 2 (ADDR = 0x22).

35.5.46 Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 3 (ADDR = 0x23).

Address: 21B_C000h base + 630h offset = 21B_C630h

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

OCOTP_MAC1 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 3 (ADDR = 0x23).

35.5.47 Value of OTP Bank4 Word4 (MAC Address) (OCOTP_RESERVED) (OCOTP_MAC)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 4 (ADDR = 0x24).

Address: 21B_C000h base + 640h offset = 21B_C640h

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

OCOTP_MAC field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 4 (ADDR = 0x24).

35.5.48 Value of OTP Bank4 Word5 (CRC Key) (OCOTP_CRC)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 5 (ADDR = 0x25).

Address: 21B_C000h base + 650h offset = 21B_C650h

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

OCOTP_CRC field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 5 (ADDR = 0x25).

35.5.49 Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (OCOTP_GP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 6 (ADDR = 0x26).

Address: 21B_C000h base + 660h offset = 21B_C660h

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

OCOTP_GP1 field descriptions

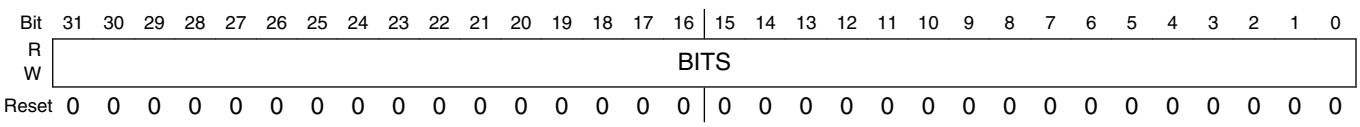
Field	Description
BITS	Reflects value of OTP Bank 4, word 6 (ADDR = 0x26).

35.5.50 Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (OCOTP_GP2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 7 (ADDR = 0x27).

Address: 21B_C000h base + 670h offset = 21B_C670h



OCOTP_GP2 field descriptions

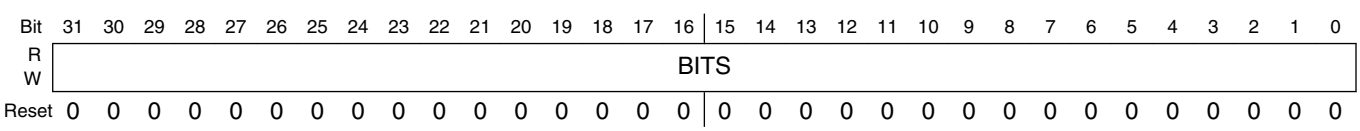
Field	Description
BITS	Reflects value of OTP Bank 4, word 7 (ADDR = 0x27).

35.5.51 Value of OTP Bank5 Word0 (SW GP) (OCOTP_SW_GP0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 0 (ADDR = 0x28).

Address: 21B_C000h base + 680h offset = 21B_C680h



OCOTP_SW_GP0 field descriptions

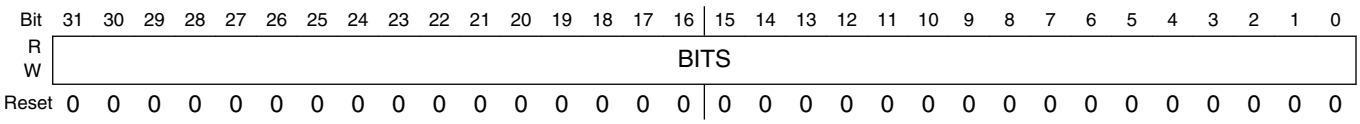
Field	Description
BITS	Reflects value of OTP Bank 5, word 0 (ADDR = 0x28).

35.5.52 Value of OTP Bank5 Word1 (SW GP) (OCOTP_SW_GP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 1 (ADDR = 0x29).

Address: 21B_C000h base + 690h offset = 21B_C690h



OCOTP_SW_GP1 field descriptions

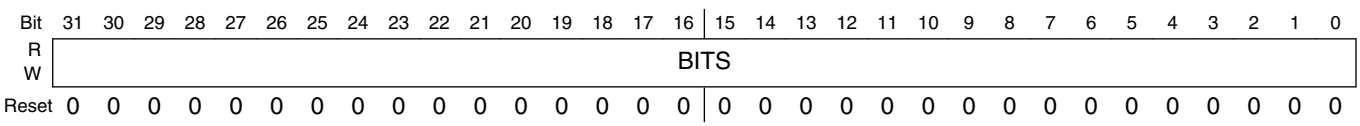
Field	Description
BITS	Reflects value of OTP Bank 5, word 1 (ADDR = 0x29).

35.5.53 Value of OTP Bank5 Word2 (SW GP) (OCOTP_SW_GP2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 2 (ADDR = 0x2a).

Address: 21B_C000h base + 6A0h offset = 21B_C6A0h



OCOTP_SW_GP2 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 2 (ADDR = 0x2a).

35.5.54 Value of OTP Bank5 Word3 (SW GP) (OCOTP_SW_GP3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 3 (ADDR = 0x2b).

Address: 21B_C000h base + 6B0h offset = 21B_C6B0h

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

OCOTP_SW_GP3 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 3 (ADDR = 0x2b).

35.5.55 Value of OTP Bank5 Word4 (SW GP) (OCOTP_SW_GP4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 4 (ADDR = 0x2c).

Address: 21B_C000h base + 6C0h offset = 21B_C6C0h

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

OCOTP_SW_GP4 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 4 (ADDR = 0x2c).

35.5.56 Value of OTP Bank5 Word5 (Misc Conf) (OCOTP_MISC_CONF)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 5 (ADDR = 0x2d).

Address: 21B_C000h base + 6D0h offset = 21B_C6D0h

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

OCOTP_MISC_CONF field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 5 (ADDR = 0x2d).

35.5.57 Value of OTP Bank5 Word6 (Field Return) (OCOTP_FIELD_RETURN)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 6 (ADDR = 0x2e).

Address: 21B_C000h base + 6E0h offset = 21B_C6E0h

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

OCOTP_FIELD_RETURN field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 6 (ADDR = 0x2e).

35.5.58 Value of OTP Bank5 Word7 (SRK Revoke) (OCOTP_SRK_REVOKE)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 7 (ADDR = 0x2f).

Address: 21B_C000h base + 6F0h offset = 21B_C6F0h

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

OCOTP_SRK_REVOKE field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 7 (ADDR = 0x2f).

35.5.59 Value of OTP Bank6 Word0 (ROM Patch) (OCOTP_ROM_PATCH0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 0 (ADDR = 0x30).

Address: 21B_C000h base + 800h offset = 21B_C800h

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

OCOTP_ROM_PATCH0 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 0 (ADDR = 0x30).

35.5.60 Value of OTP Bank6 Word1 (ROM Patch) (OCOTP_ROM_PATCH1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 1 (ADDR = 0x31).

Address: 21B_C000h base + 810h offset = 21B_C810h

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

OCOTP_ROM_PATCH1 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 1 (ADDR = 0x31).

35.5.61 Value of OTP Bank6 Word2 (ROM Patch) (OCOTP_ROM_PATCH2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 2 (ADDR = 0x32).

Address: 21B_C000h base + 820h offset = 21B_C820h

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

OCOTP_ROM_PATCH2 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 2 (ADDR = 0x32).

35.5.62 Value of OTP Bank6 Word3 (ROM Patch) (OCOTP_ROM_PATCH3)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 3 (ADDR = 0x33).

Address: 21B_C000h base + 830h offset = 21B_C830h

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

OCOTP_ROM_PATCH3 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 3 (ADDR = 0x33).

35.5.63 Value of OTP Bank6 Word4 (ROM Patch) (OCOTP_ROM_PATCH4)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 4 (ADDR = 0x34).

Address: 21B_C000h base + 840h offset = 21B_C840h

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

OCOTP_ROM_PATCH4 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 4 (ADDR = 0x34).

35.5.64 Value of OTP Bank6 Word5 (ROM Patch) (OCOTP_ROM_PATCH5)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 5 (ADDR = 0x35).

Address: 21B_C000h base + 850h offset = 21B_C850h

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

OCOTP_ROM_PATCH5 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 5 (ADDR = 0x35).

35.5.65 Value of OTP Bank6 Word6 (ROM Patch) (OCOTP_ROM_PATCH6)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 6 (ADDR = 0x36).

Address: 21B_C000h base + 860h offset = 21B_C860h

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

OCOTP_ROM_PATCH6 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 6 (ADDR = 0x36).

35.5.66 Value of OTP Bank6 Word7 (ROM Patch) (OCOTP_ROM_PATCH7)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 6, word 7 (ADDR = 0x37).

Address: 21B_C000h base + 870h offset = 21B_C870h

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

OCOTP_ROM_PATCH7 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 6, word 7 (ADDR = 0x37).

35.5.67 Value of OTP Bank7 Word0 (ROM Patch) (OCOTP_ROM_PATCH8)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 0 (ADDR = 0x38).

Address: 21B_C000h base + 880h offset = 21B_C880h

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

OCOTP_ROM_PATCH8 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 7, word 0 (ADDR = 0x38).

35.5.68 Value of OTP Bank7 Word1 (ROM Patch) (OCOTP_ROM_PATCH9)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 1 (ADDR = 0x39).

Address: 21B_C000h base + 890h offset = 21B_C890h

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

OCOTP_ROM_PATCH9 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 7, word 1 (ADDR = 0x39).

35.5.69 Value of OTP Bank7 Word2 (ROM Patch) (OCOTP_ROM_PATCH10)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 2 (ADDR = 0x3a).

Address: 21B_C000h base + 8A0h offset = 21B_C8A0h

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

OCOTP_ROM_PATCH10 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 7, word 2 (ADDR = 0x3a).

35.5.70 Value of OTP Bank7 Word3 (ROM Patch) (OCOTP_ROM_PATCH11)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 3 (ADDR = 0x3b).

Address: 21B_C000h base + 8B0h offset = 21B_C8B0h

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

OCOTP_ROM_PATCH11 field descriptions

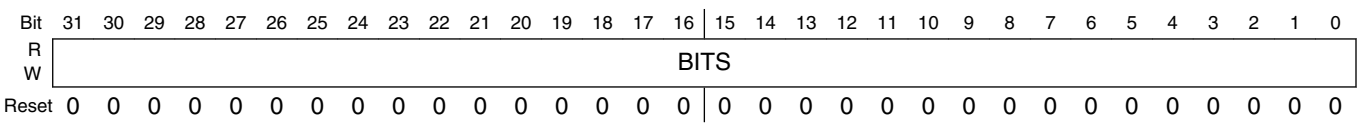
Field	Description
BITS	Reflects value of OTP Bank 7, word 3 (ADDR = 0x3b).

35.5.71 Value of OTP Bank7 Word4 (ROM Patch) (OCOTP_ROM_PATCH12)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 4 (ADDR = 0x3c).

Address: 21B_C000h base + 8C0h offset = 21B_C8C0h



OCOTP_ROM_PATCH12 field descriptions

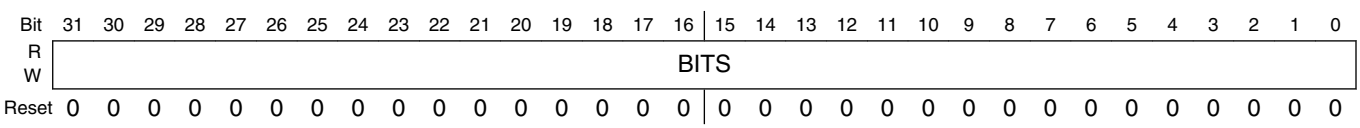
Field	Description
BITS	Reflects value of OTP Bank 7, word 4 (ADDR = 0x3c).

35.5.72 Value of OTP Bank7 Word5 (ROM Patch) (OCOTP_ROM_PATCH13)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 5 (ADDR = 0x3d).

Address: 21B_C000h base + 8D0h offset = 21B_C8D0h



OCOTP_ROM_PATCH13 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 7, word 5 (ADDR = 0x3d).

35.5.73 Value of OTP Bank7 Word6 (ROM Patch) (OCOTP_ROM_PATCH14)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 6 (ADDR = 0x3e).

Address: 21B_C000h base + 8E0h offset = 21B_C8E0h

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

OCOTP_ROM_PATCH14 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 7, word 6 (ADDR = 0x3e).

35.5.74 Value of OTP Bank7 Word7 (ROM Patch) (OCOTP_ROM_PATCH15)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 7, word 7 (ADDR = 0x3f).

Address: 21B_C000h base + 8F0h offset = 21B_C8F0h

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

OCOTP_ROM_PATCH15 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 7, word 7 (ADDR = 0x3f).

35.5.75 Value of OTP Bank8 Word0 (General Purpose Customer Defined Info) (OCOTP_GP30)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 0 (ADDR = 0x40).

Address: 21B_C000h base + 900h offset = 21B_C900h

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

OCOTP_GP30 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 8, word 0 (ADDR = 0x40).

35.5.76 Value of OTP Bank8 Word1 (General Purpose Customer Defined Info) (OCOTP_GP31)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 1 (ADDR = 0x41).

Address: 21B_C000h base + 910h offset = 21B_C910h

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

OCOTP_GP31 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 8, word 1 (ADDR = 0x41).

35.5.77 Value of OTP Bank8 Word2 (General Purpose Customer Defined Info) (OCOTP_GP32)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 2 (ADDR = 0x42).

Address: 21B_C000h base + 920h offset = 21B_C920h

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

OCOTP_GP32 field descriptions

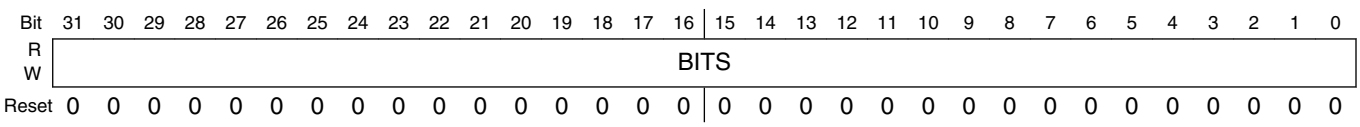
Field	Description
BITS	Reflects value of OTP Bank 8, word 2 (ADDR = 0x42).

35.5.78 Value of OTP Bank8 Word3 (General Purpose Customer Defined Info) (OCOTP_GP33)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 3 (ADDR = 0x43).

Address: 21B_C000h base + 930h offset = 21B_C930h



OCOTP_GP33 field descriptions

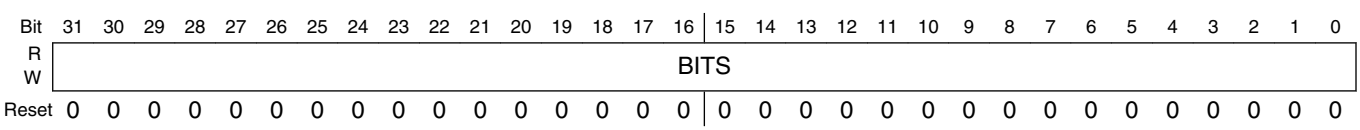
Field	Description
BITS	Reflects value of OTP Bank 8, word 3 (ADDR = 0x43).

35.5.79 Value of OTP Bank8 Word4 (General Purpose Customer Defined Info) (OCOTP_GP34)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 4 (ADDR = 0x44).

Address: 21B_C000h base + 940h offset = 21B_C940h



OCOTP_GP34 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 8, word 4 (ADDR = 0x44).

35.5.80 Value of OTP Bank8 Word5 (General Purpose Customer Defined Info) (OCOTP_GP35)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 5 (ADDR = 0x45).

Address: 21B_C000h base + 950h offset = 21B_C950h

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

OCOTP_GP35 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 8, word 5 (ADDR = 0x45).

35.5.81 Value of OTP Bank8 Word6 (General Purpose Customer Defined Info) (OCOTP_GP36)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 6 (ADDR = 0x46).

Address: 21B_C000h base + 960h offset = 21B_C960h

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

OCOTP_GP36 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 8, word 6 (ADDR = 0x46).

35.5.82 Value of OTP Bank8 Word7 (General Purpose Customer Defined Info) (OCOTP_GP37)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 8, word 7 (ADDR = 0x47).

Address: 21B_C000h base + 970h offset = 21B_C970h

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

OCOTP_GP37 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 8, word 7 (ADDR = 0x47).

35.5.83 Value of OTP Bank9 Word0 (General Purpose Customer Defined Info) (OCOTP_GP38)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 0 (ADDR = 0x48).

Address: 21B_C000h base + 980h offset = 21B_C980h

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

OCOTP_GP38 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 0 (ADDR = 0x48).

35.5.84 Value of OTP Bank9 Word1 (General Purpose Customer Defined Info) (OCOTP_GP39)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 1 (ADDR = 0x49).

Address: 21B_C000h base + 990h offset = 21B_C990h

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

OCOTP_GP39 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 1 (ADDR = 0x49).

35.5.85 Value of OTP Bank9 Word2 (General Purpose Customer Defined Info) (OCOTP_GP310)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 2 (ADDR = 0x4a).

Address: 21B_C000h base + 9A0h offset = 21B_C9A0h

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

OCOTP_GP310 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 2 (ADDR = 0x4a).

35.5.86 Value of OTP Bank9 Word3 (General Purpose Customer Defined Info) (OCOTP_GP311)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 3 (ADDR = 0x4b).

Address: 21B_C000h base + 9B0h offset = 21B_C9B0h

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

OCOTP_GP311 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 3 (ADDR = 0x4b).

35.5.87 Value of OTP Bank9 Word4 (General Purpose Customer Defined Info) (OCOTP_GP312)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 4 (ADDR = 0x4c).

Address: 21B_C000h base + 9C0h offset = 21B_C9C0h

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

OCOTP_GP312 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 4 (ADDR = 0x4c).

35.5.88 Value of OTP Bank9 Word5 (General Purpose Customer Defined Info) (OCOTP_GP313)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 5 (ADDR = 0x4d).

Address: 21B_C000h base + 9D0h offset = 21B_C9D0h

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

OCOTP_GP313 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 5 (ADDR = 0x4d).

35.5.89 Value of OTP Bank9 Word6 (General Purpose Customer Defined Info) (OCOTP_GP314)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 6 (ADDR = 0x4e).

Address: 21B_C000h base + 9E0h offset = 21B_C9E0h

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

OCOTP_GP314 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 6 (ADDR = 0x4e).

35.5.90 Value of OTP Bank9 Word7 (General Purpose Customer Defined Info) (OCOTP_GP315)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 9, word 7 (ADDR = 0x4f).

Address: 21B_C000h base + 9F0h offset = 21B_C9F0h

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

OCOTP_GP315 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 9, word 7 (ADDR = 0x4f).

35.5.91 Value of OTP Bank10 Word0 (General Purpose Customer Defined Info) (OCOTP_GP40)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 0 (ADDR = 0x50).

Address: 21B_C000h base + A00h offset = 21B_CA00h

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

OCOTP_GP40 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 0 (ADDR = 0x50).

35.5.92 Value of OTP Bank10 Word1 (General Purpose Customer Defined Info) (OCOTP_GP41)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 1 (ADDR = 0x51).

Address: 21B_C000h base + A10h offset = 21B_CA10h

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

OCOTP_GP41 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 1 (ADDR = 0x51).

35.5.93 Value of OTP Bank10 Word2 (General Purpose Customer Defined Info) (OCOTP_GP42)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 2 (ADDR = 0x52).

Address: 21B_C000h base + A20h offset = 21B_CA20h

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

OCOTP_GP42 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 2 (ADDR = 0x52).

35.5.94 Value of OTP Bank10 Word3 (General Purpose Customer Defined Info) (OCOTP_GP43)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 3 (ADDR = 0x53).

Address: 21B_C000h base + A30h offset = 21B_CA30h

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

OCOTP_GP43 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 3 (ADDR = 0x53).

35.5.95 Value of OTP Bank10 Word4 (General Purpose Customer Defined Info) (OCOTP_GP44)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 4 (ADDR = 0x54).

Address: 21B_C000h base + A40h offset = 21B_CA40h

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

OCOTP_GP44 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 4 (ADDR = 0x54).

35.5.96 Value of OTP Bank10 Word5 (General Purpose Customer Defined Info) (OCOTP_GP45)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 5 (ADDR = 0x55).

Address: 21B_C000h base + A50h offset = 21B_CA50h

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

OCOTP_GP45 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 5 (ADDR = 0x55).

35.5.97 Value of OTP Bank10 Word6 (General Purpose Customer Defined Info) (OCOTP_GP46)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 6 (ADDR = 0x56).

Address: 21B_C000h base + A60h offset = 21B_CA60h

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

OCOTP_GP46 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 6 (ADDR = 0x56).

35.5.98 Value of OTP Bank10 Word7 (General Purpose Customer Defined Info) (OCOTP_GP47)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 10, word 7 (ADDR = 0x57).

Address: 21B_C000h base + A70h offset = 21B_CA70h

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

OCOTP_GP47 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 10, word 7 (ADDR = 0x57).

35.5.99 Value of OTP Bank11 Word0 (General Purpose Customer Defined Info) (OCOTP_GP48)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 0 (ADDR = 0x58).

Address: 21B_C000h base + A80h offset = 21B_CA80h

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

OCOTP_GP48 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 0 (ADDR = 0x58).

35.5.100 Value of OTP Bank11 Word1 (General Purpose Customer Defined Info) (OCOTP_GP49)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 1 (ADDR = 0x59).

Address: 21B_C000h base + A90h offset = 21B_CA90h

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

OCOTP_GP49 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 1 (ADDR = 0x59).

35.5.101 Value of OTP Bank11 Word2 (General Purpose Customer Defined Info) (OCOTP_GP410)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 2 (ADDR = 0x5A).

Address: 21B_C000h base + AA0h offset = 21B_CAA0h

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

OCOTP_GP410 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 2 (ADDR = 0x5A).

35.5.102 Value of OTP Bank11 Word3 (General Purpose Customer Defined Info) (OCOTP_GP411)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 3 (ADDR = 0x5B).

Address: 21B_C000h base + AB0h offset = 21B_CAB0h

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

OCOTP_GP411 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 3 (ADDR = 0x5B).

35.5.103 Value of OTP Bank11 Word4 (General Purpose Customer Defined Info) (OCOTP_GP412)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 4 (ADDR = 0x5C).

Address: 21B_C000h base + AC0h offset = 21B_CAC0h

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

OCOTP_GP412 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 4 (ADDR = 0x5C).

35.5.104 Value of OTP Bank11 Word5 (General Purpose Customer Defined Info) (OCOTP_GP413)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 5 (ADDR = 0x5D).

Address: 21B_C000h base + AD0h offset = 21B_CAD0h

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

OCOTP_GP413 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 5 (ADDR = 0x5D).

35.5.105 Value of OTP Bank11 Word6 (General Purpose Customer Defined Info) (OCOTP_GP414)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 6 (ADDR = 0x5E).

Address: 21B_C000h base + AE0h offset = 21B_CAE0h

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

OCOTP_GP414 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 6 (ADDR = 0x5E).

35.5.106 Value of OTP Bank11 Word7 (General Purpose Customer Defined Info) (OCOTP_GP415)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 11, word 7 (ADDR = 0x5F).

Address: 21B_C000h base + AF0h offset = 21B_CAF0h

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

OCOTP_GP415 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 11, word 7 (ADDR = 0x5F).

35.5.107 Value of OTP Bank12 Word0 (General Purpose Customer Defined Info) (OCOTP_GP50)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 0 (ADDR = 0x60).

Address: 21B_C000h base + B00h offset = 21B_CB00h

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

OCOTP_GP50 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 0 (ADDR = 0x60).

35.5.108 Value of OTP Bank12 Word1 (General Purpose Customer Defined Info) (OCOTP_GP51)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 1 (ADDR = 0x61).

Address: 21B_C000h base + B10h offset = 21B_CB10h

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

OCOTP_GP51 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 1 (ADDR = 0x61).

35.5.109 Value of OTP Bank12 Word2 (General Purpose Customer Defined Info) (OCOTP_GP52)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 2 (ADDR = 0x62).

Address: 21B_C000h base + B20h offset = 21B_CB20h

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

OCOTP_GP52 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 2 (ADDR = 0x62).

35.5.110 Value of OTP Bank12 Word3 (General Purpose Customer Defined Info) (OCOTP_GP53)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 3 (ADDR = 0x63).

Address: 21B_C000h base + B30h offset = 21B_CB30h

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

OCOTP_GP53 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 3 (ADDR = 0x63).

35.5.111 Value of OTP Bank12 Word4 (General Purpose Customer Defined Info) (OCOTP_GP54)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 4 (ADDR = 0x64).

Address: 21B_C000h base + B40h offset = 21B_CB40h

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

OCOTP_GP54 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 4 (ADDR = 0x64).

35.5.112 Value of OTP Bank12 Word5 (General Purpose Customer Defined Info) (OCOTP_GP55)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 5 (ADDR = 0x65).

Address: 21B_C000h base + B50h offset = 21B_CB50h

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

OCOTP_GP55 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 5 (ADDR = 0x65).

35.5.113 Value of OTP Bank12 Word6 (General Purpose Customer Defined Info) (OCOTP_GP56)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 6 (ADDR = 0x66).

Address: 21B_C000h base + B60h offset = 21B_CB60h

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

OCOTP_GP56 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 6 (ADDR = 0x66).

35.5.114 Value of OTP Bank12 Word7 (General Purpose Customer Defined Info) (OCOTP_GP57)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 12, word 7 (ADDR = 0x67).

Address: 21B_C000h base + B70h offset = 21B_CB70h

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

OCOTP_GP57 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 12, word 7 (ADDR = 0x67).

35.5.115 Value of OTP Bank13 Word0 (General Purpose Customer Defined Info) (OCOTP_GP58)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 0 (ADDR = 0x68).

Address: 21B_C000h base + B80h offset = 21B_CB80h

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

OCOTP_GP58 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 0 (ADDR = 0x68).

35.5.116 Value of OTP Bank13 Word1 (General Purpose Customer Defined Info) (OCOTP_GP59)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 1 (ADDR = 0x69).

Address: 21B_C000h base + B90h offset = 21B_CB90h

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

OCOTP_GP59 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 1 (ADDR = 0x69).

35.5.117 Value of OTP Bank13 Word2 (General Purpose Customer Defined Info) (OCOTP_GP510)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 2 (ADDR = 0x6A).

Address: 21B_C000h base + BA0h offset = 21B_CBA0h

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

OCOTP_GP510 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 2 (ADDR = 0x6A).

35.5.118 Value of OTP Bank13 Word3 (General Purpose Customer Defined Info) (OCOTP_GP511)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 3 (ADDR = 0x6B).

Address: 21B_C000h base + BB0h offset = 21B_CBB0h

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

OCOTP_GP511 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 3 (ADDR = 0x6B).

35.5.119 Value of OTP Bank13 Word4 (General Purpose Customer Defined Info) (OCOTP_GP512)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 4 (ADDR = 0x6C).

Address: 21B_C000h base + BC0h offset = 21B_CBC0h

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

OCOTP_GP512 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 4 (ADDR = 0x6C).

35.5.120 Value of OTP Bank13 Word5 (General Purpose Customer Defined Info) (OCOTP_GP513)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 5 (ADDR = 0x6D).

Address: 21B_C000h base + BD0h offset = 21B_CBD0h

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

OCOTP_GP513 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 5 (ADDR = 0x6D).

35.5.121 Value of OTP Bank13 Word6 (General Purpose Customer Defined Info) (OCOTP_GP514)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 6 (ADDR = 0x6E).

Address: 21B_C000h base + BE0h offset = 21B_CBE0h

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

OCOTP_GP514 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 6 (ADDR = 0x6E).

35.5.122 Value of OTP Bank13 Word7 (General Purpose Customer Defined Info) (OCOTP_GP515)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 13, word 7 (ADDR = 0x6F).

Address: 21B_C000h base + BF0h offset = 21B_CBF0h

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

OCOTP_GP515 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 13, word 7 (ADDR = 0x6F).

35.5.123 Value of OTP Bank14 Word0 (General Purpose Customer Defined Info) (OCOTP_GP60)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 0 (ADDR = 0x70).

Address: 21B_C000h base + C00h offset = 21B_CC00h

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

OCOTP_GP60 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 0 (ADDR = 0x70).

35.5.124 Value of OTP Bank14 Word1 (General Purpose Customer Defined Info) (OCOTP_GP61)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 1 (ADDR = 0x71).

Address: 21B_C000h base + C10h offset = 21B_CC10h

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

OCOTP_GP61 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 1 (ADDR = 0x71).

35.5.125 Value of OTP Bank14 Word2 (General Purpose Customer Defined Info) (OCOTP_GP62)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 2 (ADDR = 0x72).

Address: 21B_C000h base + C20h offset = 21B_CC20h

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

OCOTP_GP62 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 2 (ADDR = 0x72).

35.5.126 Value of OTP Bank14 Word3 (General Purpose Customer Defined Info) (OCOTP_GP63)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 3 (ADDR = 0x73).

Address: 21B_C000h base + C30h offset = 21B_CC30h

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

OCOTP_GP63 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 3 (ADDR = 0x73).

35.5.127 Value of OTP Bank14 Word4 (General Purpose Customer Defined Info) (OCOTP_GP64)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 4 (ADDR = 0x74).

Address: 21B_C000h base + C40h offset = 21B_CC40h

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

OCOTP_GP64 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 4 (ADDR = 0x74).

35.5.128 Value of OTP Bank14 Word5 (General Purpose Customer Defined Info) (OCOTP_GP65)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 5 (ADDR = 0x75).

Address: 21B_C000h base + C50h offset = 21B_CC50h

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

OCOTP_GP65 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 5 (ADDR = 0x75).

35.5.129 Value of OTP Bank14 Word6 (General Purpose Customer Defined Info) (OCOTP_GP66)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 6 (ADDR = 0x76).

Address: 21B_C000h base + C60h offset = 21B_CC60h

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

OCOTP_GP66 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 6 (ADDR = 0x76).

35.5.130 Value of OTP Bank14 Word7 (General Purpose Customer Defined Info) (OCOTP_GP67)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 14, word 7 (ADDR = 0x77).

Address: 21B_C000h base + C70h offset = 21B_CC70h

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

OCOTP_GP67 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 14, word 7 (ADDR = 0x77).

35.5.131 Value of OTP Bank15 Word0 (General Purpose Customer Defined Info) (OCOTP_GP70)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 0 (ADDR = 0x78).

Address: 21B_C000h base + C80h offset = 21B_CC80h

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

OCOTP_GP70 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 0 (ADDR = 0x78).

35.5.132 Value of OTP Bank15 Word1 (General Purpose Customer Defined Info) (OCOTP_GP71)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 1 (ADDR = 0x79).

Address: 21B_C000h base + C90h offset = 21B_CC90h

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

OCOTP_GP71 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 1 (ADDR = 0x79).

35.5.133 Value of OTP Bank15 Word2 (General Purpose Customer Defined Info) (OCOTP_GP72)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 2 (ADDR = 0x7A).

Address: 21B_C000h base + CA0h offset = 21B_CCA0h

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

OCOTP_GP72 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 2 (ADDR = 0x7A).

35.5.134 Value of OTP Bank15 Word3 (General Purpose Customer Defined Info) (OCOTP_GP73)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 3 (ADDR = 0x7B).

Address: 21B_C000h base + CB0h offset = 21B_CCB0h

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

OCOTP_GP73 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 3 (ADDR = 0x7B).

35.5.135 Value of OTP Bank15 Word4 (General Purpose Customer Defined Info) (OCOTP_GP80)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 4 (ADDR = 0x7C).

Address: 21B_C000h base + CC0h offset = 21B_CCC0h

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

OCOTP_GP80 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 4 (ADDR = 0x7C).

35.5.136 Value of OTP Bank15 Word5 (General Purpose Customer Defined Info) (OCOTP_GP81)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 5 (ADDR = 0x7D).

Address: 21B_C000h base + CD0h offset = 21B_CCD0h

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

OCOTP_GP81 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 5 (ADDR = 0x7D).

35.5.137 Value of OTP Bank15 Word6 (General Purpose Customer Defined Info) (OCOTP_GP82)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 6 (ADDR = 0x7E).

Address: 21B_C000h base + CE0h offset = 21B_CCE0h

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

OCOTP_GP82 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 6 (ADDR = 0x7E).

35.5.138 Value of OTP Bank15 Word7 (General Purpose Customer Defined Info) (OCOTP_GP83)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 15, word 7 (ADDR = 0x7F).

Address: 21B_C000h base + CF0h offset = 21B_CCF0h

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

OCOTP_GP83 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 15, word 7 (ADDR = 0x7F).

Chapter 36
On-Chip RAM Memory Controller (OCRAM)

36.1 Overview

There is 1 OCRAM controller implemented in i.MX 6UltraLite. One controller is for the 128KB on-chip RAM.

The on-chip RAM block is implemented as a slave module on the 64-bit system AXI bus. Designed as a simple on-chip memory controller, it supports only one AXI port with memory banks. For the AXI port, the read and write transactions are handled by two independent modules. As it is possible to have simultaneous read and write requests from the AXI bus, each memory bank has an arbiter with a round-robin scheme. After arbitration, the granted read or write access command can then be issued to the memory cell through a read/write MUX.

The 4 memory banks are organized with the lower 2 bits of the address which is the AXI bus address and is 64 bits aligned interleaved. This allows a read access and a write access to be processed at the same time if they are targeted to different memory banks.

Various options are provided for adding a pipeline or wait-states in a read/write access, in order to ensure flexible timing control at both high and low frequencies.

The internal block diagram is shown in the figure below.

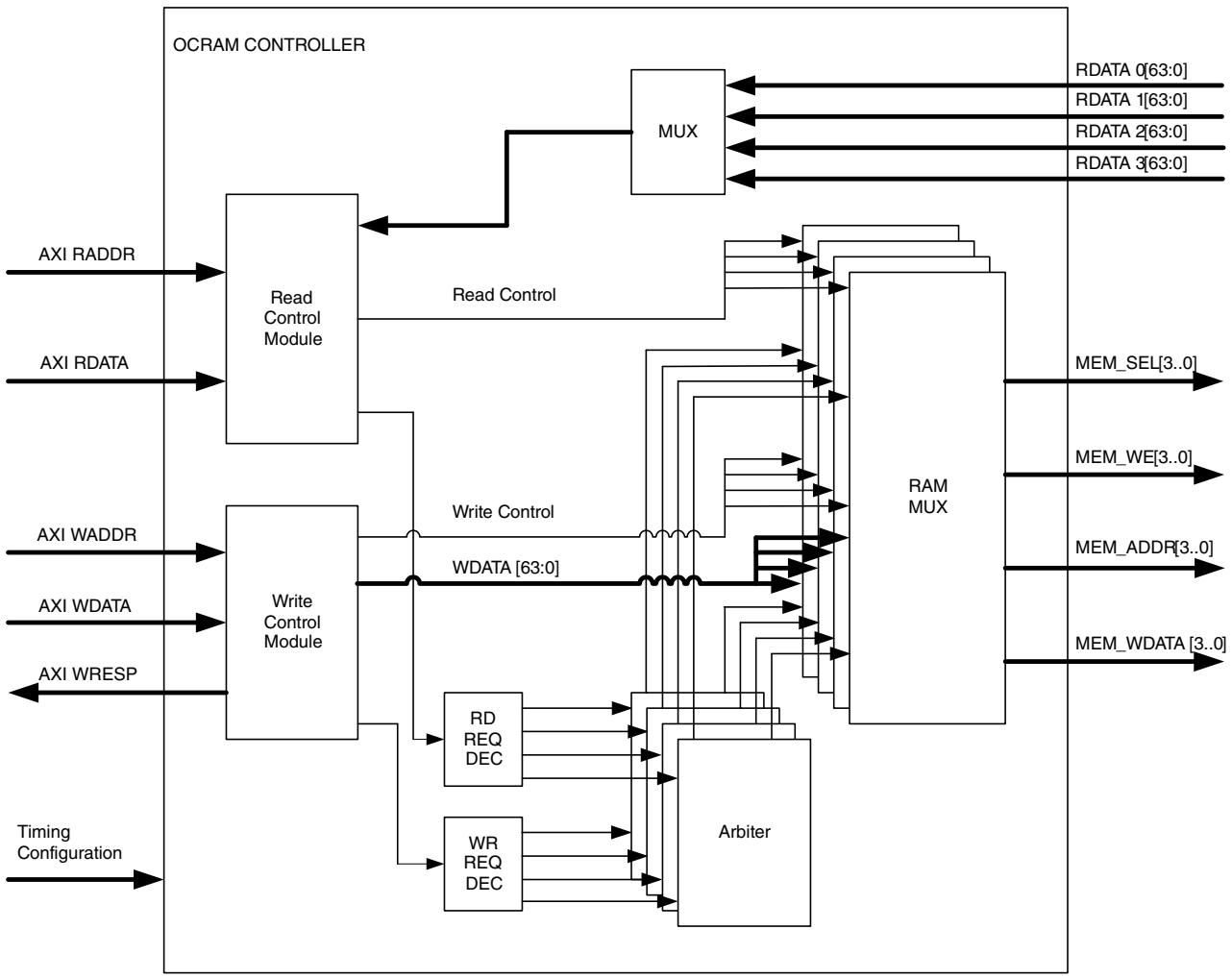


Figure 36-1. On-chip RAM Block Diagram

36.2 Basic Functions

36.2.1 Read/Write Arbitration

The detailed rules used in arbitration are as follows:

- If there is no granted read or write in the last cycle, and there is only a read request or a write request, the request will be granted.
- If there is no granted read or write in the last cycle, and there are both read or write requests coming in at the same time, the read request will be granted first.

- If a granted read/write transaction has just finished, the write/read request will have the higher priority in the next cycle.
- If the first read/write access request in a transaction is granted, all the data transfer in this burst will be finished before the next arbitration begins, that is, the round-robin arbitration mechanism is based on AXI transaction, not data access.

36.3 Advanced Features

This section describes some advanced features designed to avoid timing issues when the on-chip RAM is working at high frequency.

All of the features can be disabled/enabled by programming the corresponding fields of the General Purpose Register (IOMUXC.GPR3) bits [24:21] and bits [3:0] in the IOMUX chapter.

36.3.1 Read Data Wait State

When the wait state is enabled, it will take 2 cycles for each read access (each beat of a read burst).

This can avoid the potential timing problem caused by the longer memory access time at higher frequency.

When this feature is disabled, it only takes 1 clock cycle to finish a read transaction. That is, read data is available in the next cycle of read request becomes valid on the bus.

For the normal OCRAM, the read data wait state is configurable via IOMUXC.GPR3[21]. For the L2 cache as OCRAM, the read data wait state is configurable via IOMUXC.GPR3[0].

36.3.2 Read Address Pipeline

When this feature is enabled, the read address from the AXI master is delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issues for the read access on the memory cell at high frequency. Enabling this feature can cost, at most, 1 more clock cycle for each AXI read transaction, that is, at most 1 more clock cycle for each read burst with multiple beats of data.

When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait).

For the normal OCRAM, the read address pipeline is configurable via IOMUXC.GPR3[22]. For the L2 cache as OCRAM, the read address pipeline is configurable via IOMUXC.GPR3[1].

36.3.3 Write Data Pipeline

When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle).

For the normal OCRAM, the write data pipeline is configurable via IOMUXC.GPR3[23]. For the L2 cache as OCRAM, the write data pipeline is configurable via IOMUXC.GPR3[2].

36.3.4 Write Address Pipeline

When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would take at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle).

For the normal OCRAM, the write address pipeline is configurable via IOMUXC.GPR3[24]. For the L2 cache as OCRAM, the write address pipeline is configurable via IOMUXC.GPR3[3]

36.4 Programmable Registers

There are no programmable registers in this block; however, OCRAM configurable bits can be found in the IOMUX Controller (IOMUXC) general purpose registers found here.

- TrustZone bits: IOMUXC_GPR10
- WAIT state / Pipeline bits: IOMUXC_GPR3

Chapter 37
Power Management Unit (PMU)

37.1 Overview

The power management unit (PMU) is designed to simplify the external power interface. The power system can be split into the input power sources and their characteristics, the integrated power transforming and controlling elements, and the final load interconnection and requirements.

A typical power system utilizing the PMU is depicted below.

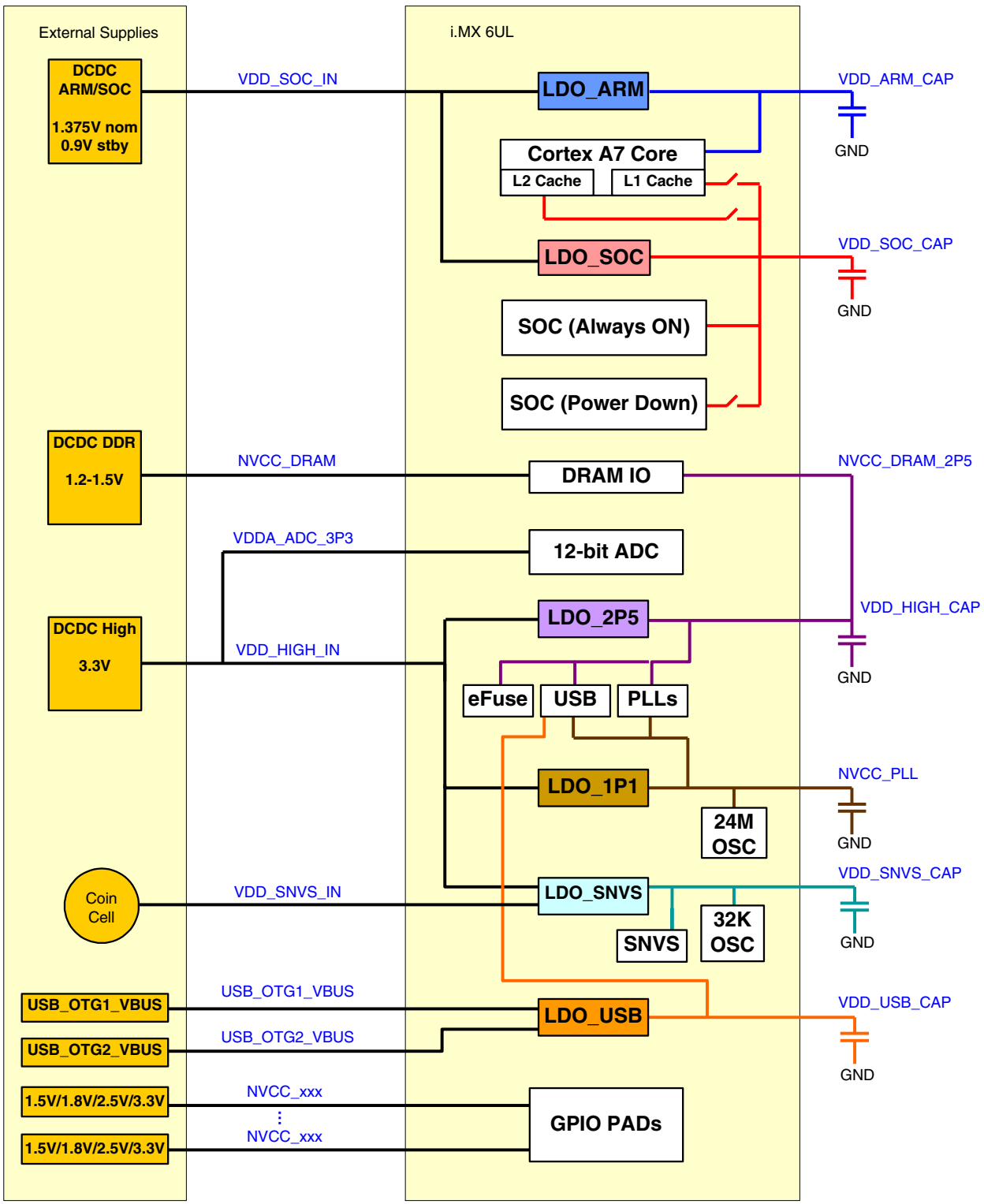


Figure 37-1. Power system overview

Utilizing seven LDO regulators, the number of external supplies is greatly reduced. Not counting the backup coin and USB inputs, the number of external supplies is reduced to two. Missing from this external supply total is the number of necessary external supplies to power the desired memory interface; that number varies depending on the type of external memory selected. Other supplies may also be necessary to supply the voltage to the different I/O power segments if their I/O voltages have to be different from what is provided above.

37.2 Digital LDO Regulators

The PMU has three digital LDO regulators. They are referred to as "digital" because of the logic loads they drive, not because of their construction. These regulators have three basic modes that are unique to the digital regulators.

- Power Gate—The regulation FET is switched off fully, limiting the current draw from the supply. The analog part of the regulator is powered down, limiting the power consumption. The output voltage falls to a level at which the residual leakage of the power FET balances with the leakage of the load. (TARG = 0x00)
- Analog regulation mode—The regulation FET is controlled such that the output voltage of the regulator equals the programmed target voltage. The target voltage is fully programmable in 25mV steps.

These modes allow the regulators to implement voltage scaling and power gating and allow bypass. With the bypass feature, all of the accuracy and control requirements can be shifted to the external supply source if capable and desired.

These digital regulators also feature brownout detection which is helpful when supplies are starting to collapse. The voltage value where brownout is signaled is programmable as an offset from the programmed target voltage. The controls are located in the PMU_MISC2 register. The core is interrupted on a brownout.

The two digital regulators are known as LDO_ARM, , and LDO_SOC. As shown in the power system overview figure, the ARM regulator powers the ARM cores. All regulators support generous programming ranges in 25mV steps. It is possible to program voltages above the process limit for the chip, thus causing permanent damage. Likewise, it is possible to program the voltage so low that the chip cannot continue to operate or even retain state without clocks. Care should be taken with these settings.

Care must be taken when raising the output voltage of the regulator rapidly. This can cause large currents to flow into the output cap of the regulator up to the limits of the input supply. When the input supply capability is exceeded, this can cause an input supply dip that may affect other regulators on the same supply. Therefore, the rate of

voltage change on the output of the regulator should be limited. When powering up the regulator, the integrated current limiter controls the ramp rate. This limiter is only effective when transitioning from the off state of the regulator (bypassed or power gated).

However, in a DVFS situation, the same high rate of change can occur if the target voltage is raised rapidly by software. To limit the rate of change, the hardware controlling the regulator effects a piecewise linear ramp by stepping the output voltage in 25mV steps until the desired output voltage is reached. The slope of the ramp is controlled by the time spent at each 25mV step and is controlled by the step time field in the PMU_MISC2 register. The same situation is not a problem when the output voltage is dropped as the load pulls down the output cap. As a result, any reduction in the programmed regulator target voltage is immediately effective with the actual supply voltage falling at a rate controlled by the load on the regulator.

37.3 Analog LDO Regulators

There are two analog regulators described here.

37.3.1 LDO 1P1

The LDO_1P1 module on the chip implements a programmable linear-regulator function from a higher analog supply voltage (2.8 V–3.3 V) to produce a nominal 1.1 V output voltage.

The output of the regulator can be programmed in 25 mV steps from 0.8 V to 1.4 V. The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitance, though the actual capacitance required should be determined by the application. A programmable brownout detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded, so the necessary steps can be taken.

Current limiting can be enabled by setting the PMU_REG_1P1[ENABLE_ILIMIT] bit to allow for in-rush current requirements during startup if needed. Active pulldown can also be enabled by setting the PMU_REG_1P1[ENABLE_PULLDOWN] bit for systems requiring this feature.

37.3.2 LDO 2P5

The LDO_2P5 module on the chip implements a programmable linear-regulator function from a higher analog supply voltage (2.8V-3.3V) to produce a nominal 2.5V output voltage.

The output of the regulator can be programmed in 25mV steps from 2.0V to 2.75V. The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitance, though the actual capacitance required should be determined by the application. A programmable brown-out detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded to take the necessary steps.

Current-limiting can be enabled by setting the REG_PMU_2P5[ENABLE_ILIMIT] bit to allow for in-rush current requirements during start-up if needed. Active-pulldown can also be enabled by setting the REG_PMU_2P5[ENABLE_PULLDOWN] bit for systems requiring this feature.

37.3.3 Low Power Operation

The 1.1 V and 2.5 V LDO includes an alternate, self-biased, low-precision, weak regulator which can be enabled for applications needing to keep the 1.1 V and 2.5 V output voltage alive during low-power modes where the main regulator and its associated global bandgap reference module are disabled.

The output of this weak regulator is not programmable and is a function of its input power supply as well as load current. The low-power mode is enabled by setting high the PMU_REG_1P1[ENABLE_WEAK_LINREG] and PMU_REG_2P5[ENABLE_WEAK_LINREG] bit of the regulator. It is recommended that the following sequence be followed to enable this mode:

1. Throttle down the 1.1 V / 2.5 V attached load to its low-power maintain state.
2. Disable the main 1.1 V / 2.5 V regulator driver by clearing the PMU_REG_1P1[ENABLE_LINREG] / PMU_REG_2P5[ENABLE_LINREG] bit.
3. Enable the weak 1.1 V / 2.5 V regulator by setting the PMU_REG_1P1[ENABLE_WEAK_LINREG] / PMU_REG_2P5[ENABLE_WEAK_LINREG] bit.

To go back to full-power operation, reverse the steps outlined above. Note that the external decoupling cap is supporting the power supply between steps 2 and 3. Therefore step 3 should happen appropriately in time relative to the discharge of the supporting capacitor.

37.4 USB LDO Regulator

The USB_LDO module on the chip implements a programmable linear-regulator function from the USB VBUS voltages (typically 5 V) to produce a nominal 3.0 V output voltage.

The output of the regulator can be programmed in 25 mV steps, from 2.625V to 3.4 V . The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitor of 4.7 μF, though the actual capacitance required should be determined by the application. A programmable brownout detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded, so the necessary steps can be taken. This regulator has a built-in power mux which allows the user to choose to run the regulator from either VBUS supply when both are present. If only one of the VBUS voltages is present, then the regulator automatically selects this supply. Current limit is also included to help the system meet in-rush current targets.

Upon attachment of VBUS, this regulator starts up in a low-power, self-preservation mode to prevent over-voltage conditions on the chip. It is expected that the user transition to full regulation by enabling the regulator and disabling the in-rush current limits via its control registers. Upon VBUS removal, it is further expected that the regulator controls are returned to their reset state.

37.5 SNVS Regulator

The SNVS regulator takes the SNVS_IN supply and generates the SNVS_CAP supply, which powers the real time clock and SNVS blocks.

If VDDHIGH_IN is present, then the SNVS_IN supply is internally shorted to the VDDHIGH_IN supply to allow coin cell recharging if necessary. The output voltage is roughly one third of SNVS_IN.

37.6 PMU Memory Map/Register Definition

The register definitions that affect the behavior of the digital LDO regulators follow.

NOTE

Some of the registers are collections of bits that affect multiple components on the chip. Those that are not pertinent to this chapter have comments in the related register bitfields.

If a full description is desired, please consult the full register programming reference in the related block.

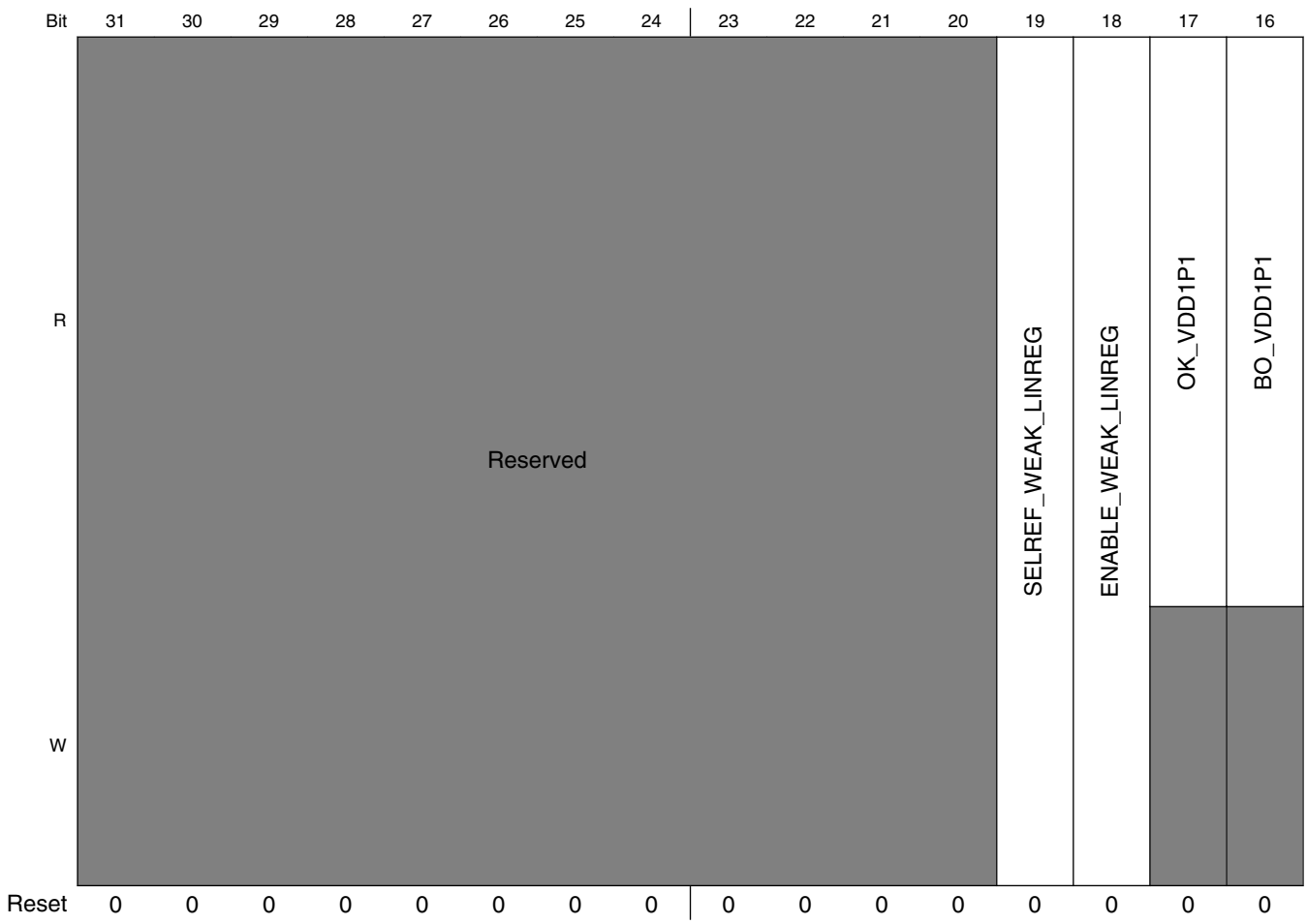
PMU memory map

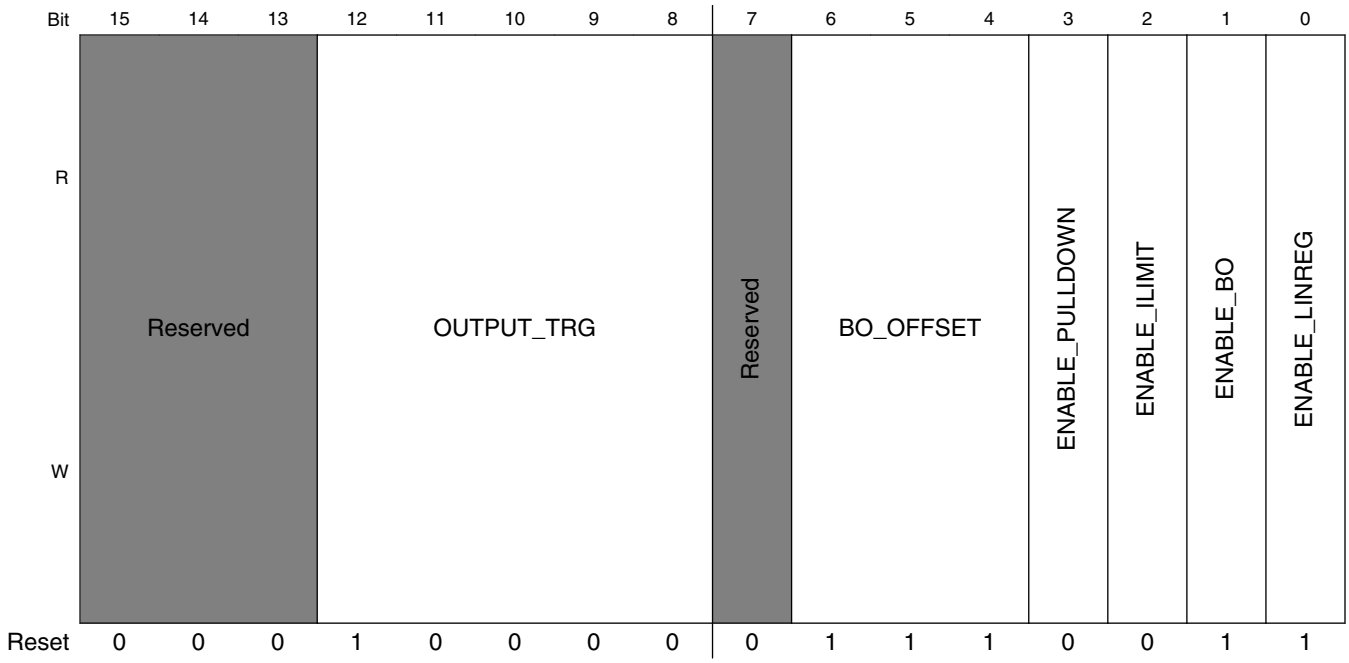
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8110	Regulator 1P1 Register (PMU_REG_1P1)	32	R/W	0000_1073h	37.6.1/2272
20C_8120	Regulator 3P0 Register (PMU_REG_3P0)	32	R/W	0000_0F74h	37.6.2/2275
20C_8130	Regulator 2P5 Register (PMU_REG_2P5)	32	R/W	0000_1073h	37.6.3/2277
20C_8140	Digital Regulator Core Register (PMU_REG_CORE)	32	R/W	0048_2012h	37.6.4/2279
20C_8150	Miscellaneous Register 0 (PMU_MISC0)	32	R/W	0400_0000h	37.6.5/2281
20C_8160	Miscellaneous Register 1 (PMU_MISC1)	32	R/W	0000_0000h	37.6.6/2284
20C_8164	Miscellaneous Register 1 (PMU_MISC1_SET)	32	R/W	0000_0000h	37.6.6/2284
20C_8168	Miscellaneous Register 1 (PMU_MISC1_CLR)	32	R/W	0000_0000h	37.6.6/2284
20C_816C	Miscellaneous Register 1 (PMU_MISC1_TOG)	32	R/W	0000_0000h	37.6.6/2284
20C_8170	Miscellaneous Control Register (PMU_MISC2)	32	R/W	0027_2727h	37.6.7/2287
20C_8174	Miscellaneous Control Register (PMU_MISC2_SET)	32	R/W	0027_2727h	37.6.7/2287
20C_8178	Miscellaneous Control Register (PMU_MISC2_CLR)	32	R/W	0027_2727h	37.6.7/2287
20C_817C	Miscellaneous Control Register (PMU_MISC2_TOG)	32	R/W	0027_2727h	37.6.7/2287
20C_8270	Low Power Control Register (PMU_LOWPWR_CTRL_SET)	32	R/W	0000_4009h	37.6.8/2291
20C_8274	Low Power Control Register (PMU_LOWPWR_CTRL_CLR)	32	R/W	0000_4009h	37.6.8/2291
20C_8278	Low Power Control Register (PMU_LOWPWR_CTRL_TOG)	32	R/W	0000_4009h	37.6.8/2291

37.6.1 Regulator 1P1 Register (PMU_REG_1P1)

This register defines the control and status bits for the 1.1V regulator. This regulator is designed to power the digital portions of the analog cells.

Address: 20C_8000h base + 110h offset = 20C_8110h





PMU_REG_1P1 field descriptions

Field	Description
31–20 -	This field is reserved.
19 SELREF_ WEAK_LINREG	Selects the source for the reference voltage of the weak 1p1 regulator. 0 Weak-linreg output tracks low-power-bandgap voltage 1 Weak-linreg output tracks VDD_SOC_CAP voltage
18 ENABLE_ WEAK_LINREG	Enables the weak 1p1 regulator. This regulator can be used when the main 1p1 regulator is disabled, under low-power conditions.
17 OK_VDD1P1	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD1P1	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. Choices must be in this range: 0x1b >= output_trg >= 0x04 **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x04 0.8V 0x10 1.1V 0x1b 1.375V
7 -	This field is reserved.

Table continues on the next page...

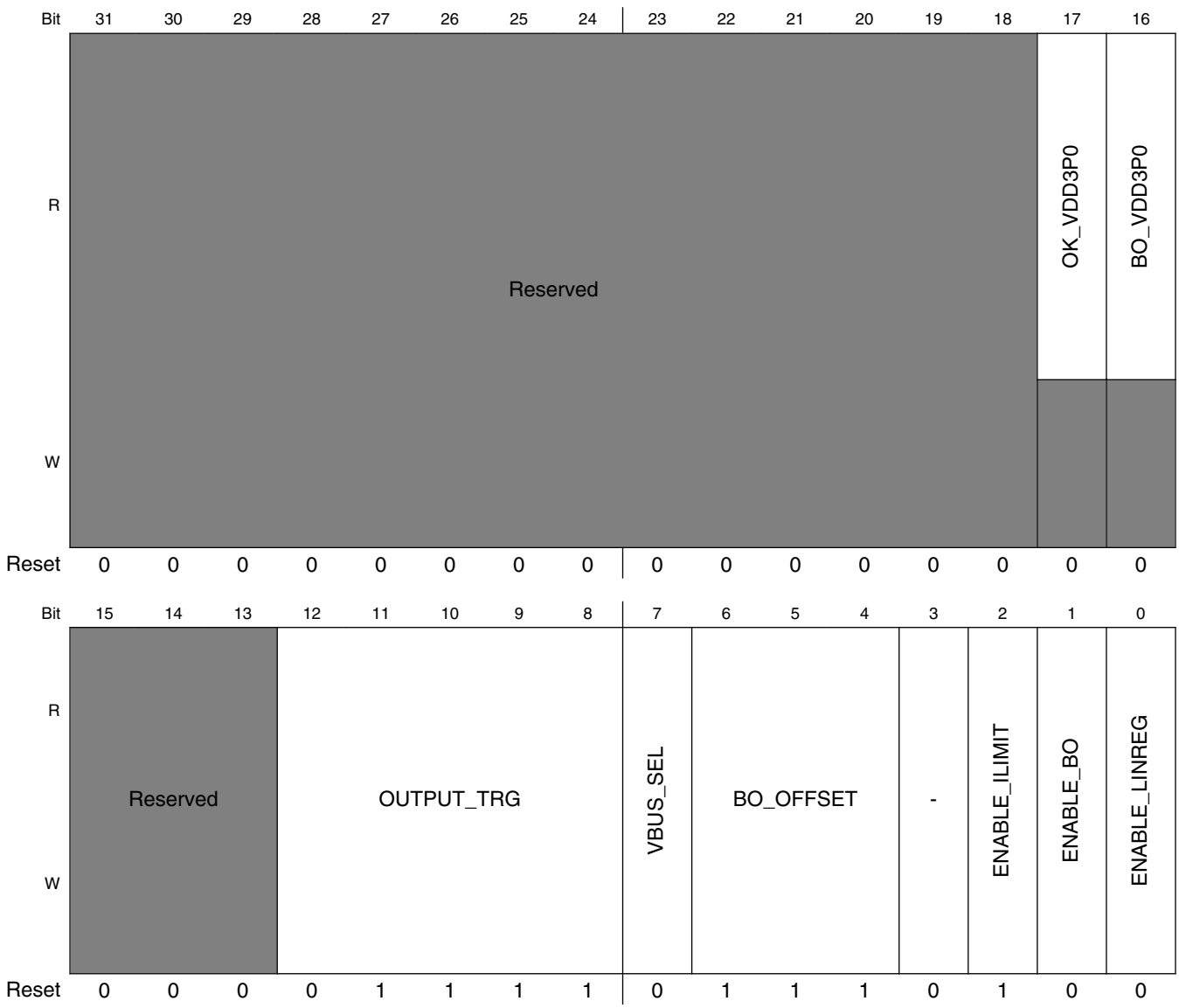
PMU_REG_1P1 field descriptions (continued)

Field	Description
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

37.6.2 Regulator 3P0 Register (PMU_REG_3P0)

This register defines the control and status bits for the 3.0V regulator powered by the host USB VBUS pin.

Address: 20C_8000h base + 120h offset = 20C_8120h



PMU_REG_3P0 field descriptions

Field	Description
31–18 -	This field is reserved.

Table continues on the next page...

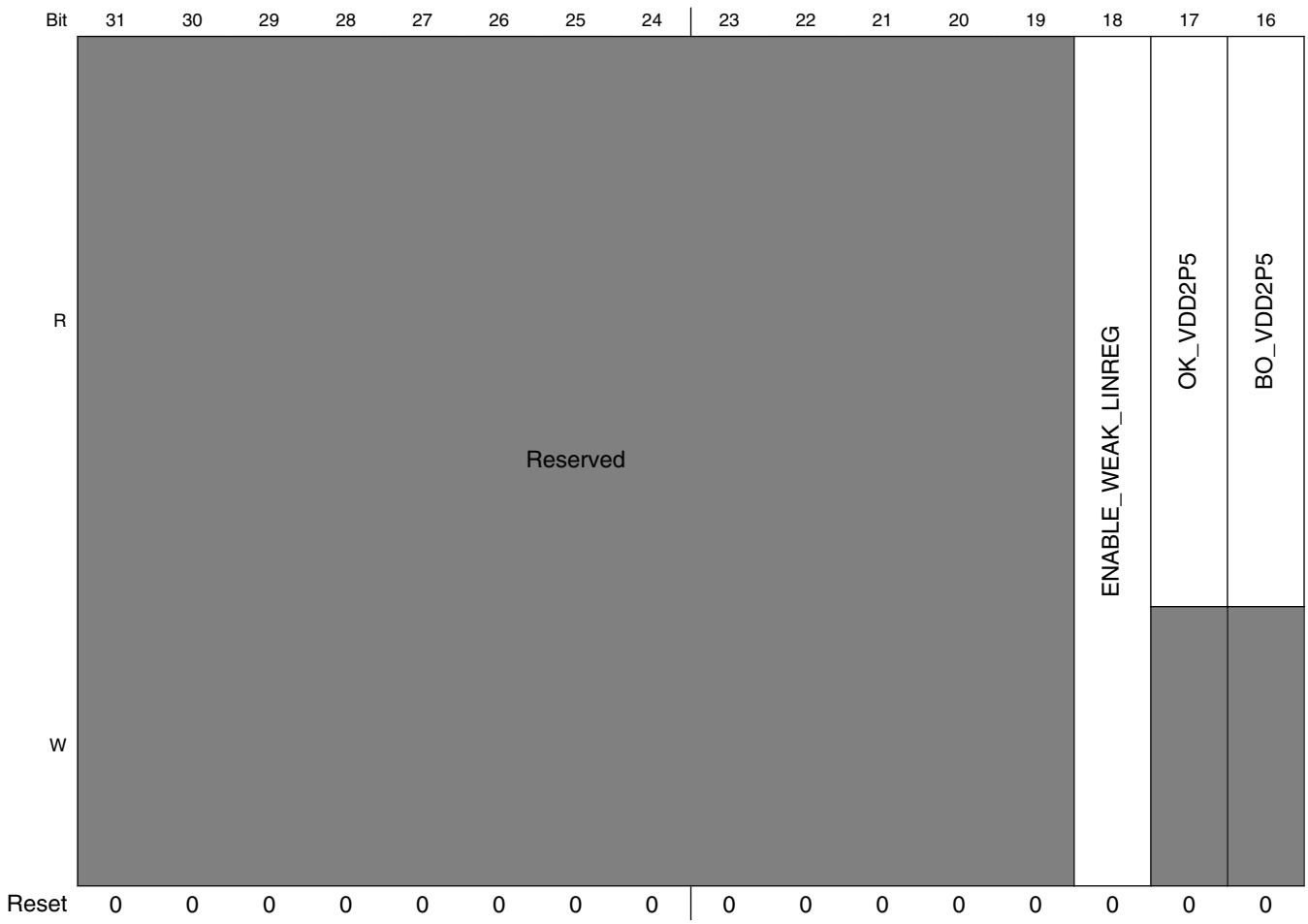
PMU_REG_3P0 field descriptions (continued)

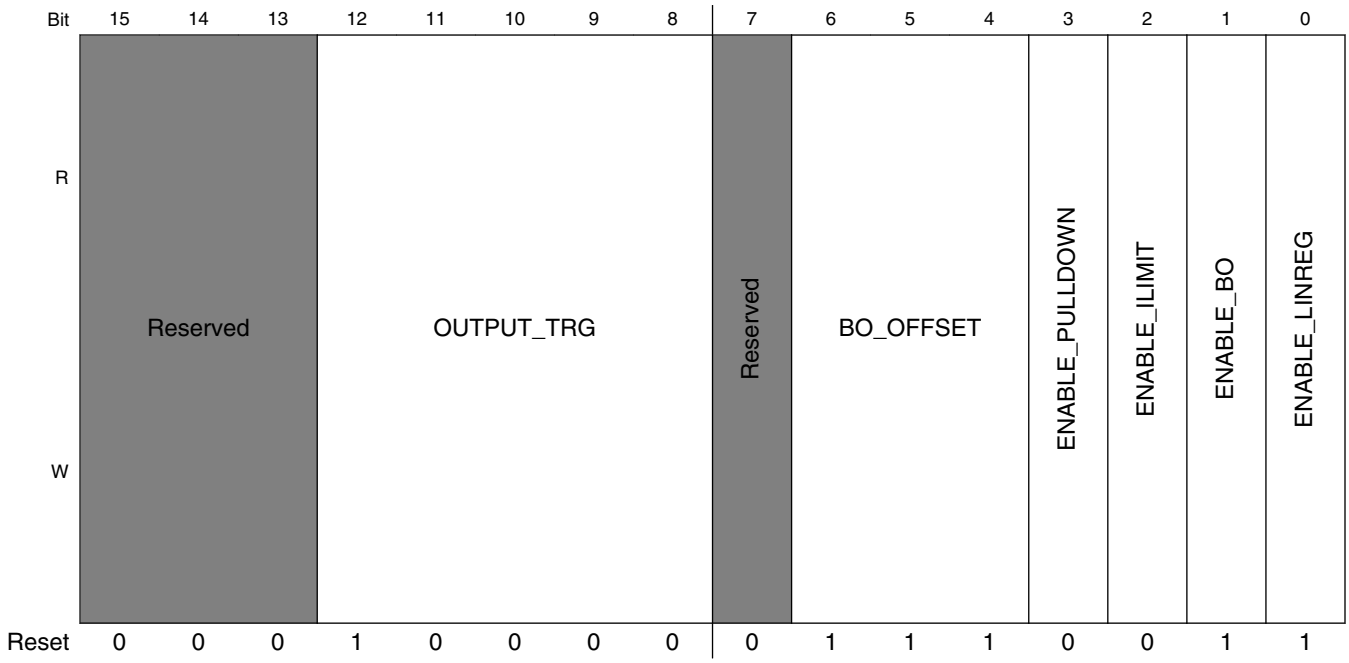
Field	Description
17 OK_VDD3P0	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD3P0	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x00 2.625V 0x0f 3.000V 0x1f 3.400V
7 VBUS_SEL	Select input voltage source for LDO_3P0 from either USB_OTG1_VBUS or USB_OTG2_VBUS. If only one of the two VBUS voltages is present, it is automatically selected. 0 **USB_OTG1_VBUS** — Utilize VBUS OTG1 for power 1 **USB_OTG2_VBUS** — Utilize VBUS OTG2 power
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may not be relevant because of input supply limitations or load operation.
3 -	Reserved
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output to be set by the programmed target voltage setting and internal bandgap reference.

37.6.3 Regulator 2P5 Register (PMU_REG_2P5)

This register defines the control and status bits for the 2.5V regulator.

Address: 20C_8000h base + 130h offset = 20C_8130h





PMU_REG_2P5 field descriptions

Field	Description
31–19 -	This field is reserved.
18 ENABLE_ WEAK_LINREG	Enables the weak 2p5 regulator. This low power regulator is used when the main 2p5 regulator is disabled to keep the 2.5V output roughly at 2.5V. Scales directly with the value of VDDHIGH_IN.
17 OK_VDD2P5	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD2P5	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x00 2.10V 0x10 2.50V 0x1f 2.875V
7 -	This field is reserved.
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator

Table continues on the next page...

PMU_REG_2P5 field descriptions (continued)

Field	Description
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

37.6.4 Digital Regulator Core Register (PMU_REG_CORE)

This register defines the function of the digital regulators

Address: 20C_8000h base + 140h offset = 20C_8140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		FET_ODRIVE	RAMP_RATE		Reserved			REG2_TARG						Reserved	
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											REG0_TARG				
Reset	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0

PMU_REG_CORE field descriptions

Field	Description
31–30 -	This field is reserved.
29 FET_ODRIVE	If set, increases the gate drive on power gating FETs to reduce leakage in the off state. Care must be taken to apply this bit only when the input supply voltage to the power FET is less than 1.1V. **NOTE:** This bit should only be used in low-power modes where the external input supply voltage is nominally 0.9V.
28–27 RAMP_RATE	Regulator voltage ramp rate. 00 Fast 01 Medium Fast

Table continues on the next page...

PMU_REG_CORE field descriptions (continued)

Field	Description
	10 Medium Slow 11 Slow
26–23 -	This field is reserved.
22–18 REG2_TARG	This field defines the target voltage for the SOC power domain. Single-bit increments reflect 25mV core voltage steps. Some steps may not be relevant because of input supply limitations or load operation. **NOTE:** This register is capable of programming an over-voltage condition on the device. Consult the datasheet Operating Ranges table for the allowed voltages. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V 00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.
17–5 -	This field is reserved. Reserved
REG0_TARG	This field defines the target voltage for the ARM core power domain. Single-bit increments reflect 25mV core voltage steps. Some steps may not be relevant because of input supply limitations or load operation. **NOTE:** This register is capable of programming an over-voltage condition on the device. Consult the datasheet Operating Ranges table for the allowed voltages. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V 00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.

37.6.5 Miscellaneous Register 0 (PMU_MISC0)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 20C_8000h base + 150h offset = 20C_8150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VID_PLL_PREDIV	XTAL_24M_PWD	RTC_XTAL_SOURCE	CLKGATE_DELAY			CLKGATE_CTRL	Reserved								OSC_XTALOK_EN
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OSC_XTALOK	OSC_I		DISCON_HIGH_SNVS	STOP_ MODE_ CONFIG		Reserved		REFTOP_VBGUP	REFTOP_VBGADJ			REFTOP_SELFBIASOFF	Reserved		REFTOP_PWD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMU_MISC0 field descriptions

Field	Description
31 VID_PLL_ PREDIV	Predivider for the source clock of the PLL's. 0 Divide by 1 1 Divide by 2
30 XTAL_24M_PWD	This field powers down the 24M crystal oscillator if set true.
29 RTC_XTAL_ SOURCE	This field indicates which chip source is being used for the rtc clock. 0 Internal ring oscillator 1 RTC_XTAL
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to PMU. 000 0.5ms 001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms 101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to PMU. 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–17 -	This field is reserved. Always set to zero.
16 OSC_XTALOK_ EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable. **NOTE:** Not related to PMU, Clocking content
15 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency. **NOTE:** Not related to PMU, clocking content.
14–13 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable.

Table continues on the next page...

PMU_MISC0 field descriptions (continued)

Field	Description
	NOTE: Not related to PMU. 00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%
12 DISCON_HIGH_SNVS	This bit controls a switch from VDD_HIGH_IN to VDD_SNVS_IN. 0 Turn on the switch 1 Turn off the switch
11–10 STOP_MODE_CONFIG	Configure the analog behavior in stop mode. 00 **SUSPEND (DSM)** — All analog except rtc powered down on stop mode assertion. 01 **STANDBY** — Analog regulators are ON. 10 **STOP (lower power)** — Analog regulators are ON. 11 **STOP (very lower power)** — Analog regulators are OFF.
9–8 -	This field is reserved. Reserved
7 REFTOP_VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable.
6–4 REFTOP_VBGADJ	000 Nominal VBG 001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2–1 -	This field is reserved.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, pll, and other analog functions on the die.

37.6.6 Miscellaneous Register 1 (PMU_MISC1*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 20C_8000h base + 160h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IRQ_DIG_BO	IRQ_ANA_BO	IRQ_TEMPHIGH	IRQ_TEMPLOW	IRQ_TEMPPANIC	Reserved									PFD_528_AUTOGATE_EN	PFD_480_AUTOGATE_EN
W	w1c	w1c	w1c	w1c	w1c											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Reserved	LVDSCLK1_IBEN	Reserved	LVDSCLK1_OBEN	Reserved					LVDS1_CLK_SEL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMU_MISC1*n* field descriptions

Field	Description
31 IRQ_DIG_BO	This status bit is set to one when when any of the digital regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted.
30 IRQ_ANA_BO	This status bit is set to one when when any of the analog regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted.
29 IRQ_TEMPHIGH	This status bit is set to one when the temperature sensor high interrupt asserts for high temperature. **NOTE:** Not related to PMU, Temperature Monitor content.
28 IRQ_TEMPLOW	This status bit is set to one when the temperature sensor low interrupt asserts for low temperature. **NOTE:** Not related to PMU, Temperature Monitor content.
27 IRQ_ TEMPPANIC	This status bit is set to one when the temperature sensor panic interrupt asserts for a panic high temperature. **NOTE:** Not related to PMU, Temperature Monitor content.
26–18 -	This field is reserved. Reserved
17 PFD_528_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_528 clocks anytime the PLL_528 is unlocked or powered off.
16 PFD_480_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_480 clocks anytime the USB1_PLL_480 is unlocked or powered off.
15–14 -	This field is reserved. Reserved
13 -	This field is reserved. Reserved
12 LVDSCLK1_ IBEN	This enables the LVDS input buffer for anaclk1/1b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU, Clocking content.
11 -	This field is reserved. Reserved
10 LVDSCLK1_ OBEN	This enables the LVDS output buffer for anaclk1/1b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU, clocking content.
9–5 -	This field is reserved. Reserved
LVDS1_CLK_ SEL	This field selects the clk to be routed to anaclk1/1b. **NOTE:** Not related to PMU. 00000 **ARM_PLL** — Arm PLL 00001 **SYS_PLL** — System PLL 00010 **PFD4** — ref_pfd4_clk == pll2_pfd0_clk 00011 **PFD5** — ref_pfd5_clk == pll2_pfd1_clk

Table continues on the next page...

PMU_MISC1*n* field descriptions (continued)

Field	Description	
	00100	**PFD6** — ref_pfd6_clk == pll2_pfd2_clk
	00101	**PFD7** — ref_pfd7_clk == pll2_pfd3_clk
	00110	**AUDIO_PLL** — Audio PLL
	00111	**VIDEO_PLL** — Video PLL
	01001	**ETHERNET_REF** — ethernet ref clock (ENET_PLL)
	01100	**USB1_PLL** — USB1 PLL clock
	01101	**USB2_PLL** — USB2 PLL clock
	01110	**PFD0** — ref_pfd0_clk == pll3_pfd0_clk
	01111	**PFD1** — ref_pfd1_clk == pll3_pfd1_clk
	10000	**PFD2** — ref_pfd2_clk == pll3_pfd2_clk
	10001	**PFD3** — ref_pfd3_clk == pll3_pfd3_clk
	10010	**XTAL** — xtal (24M)
	10101 to 11111	ref_pfd7_clk == pll2_pfd3_clk

37.6.7 Miscellaneous Control Register (PMU_MISC2*n*)

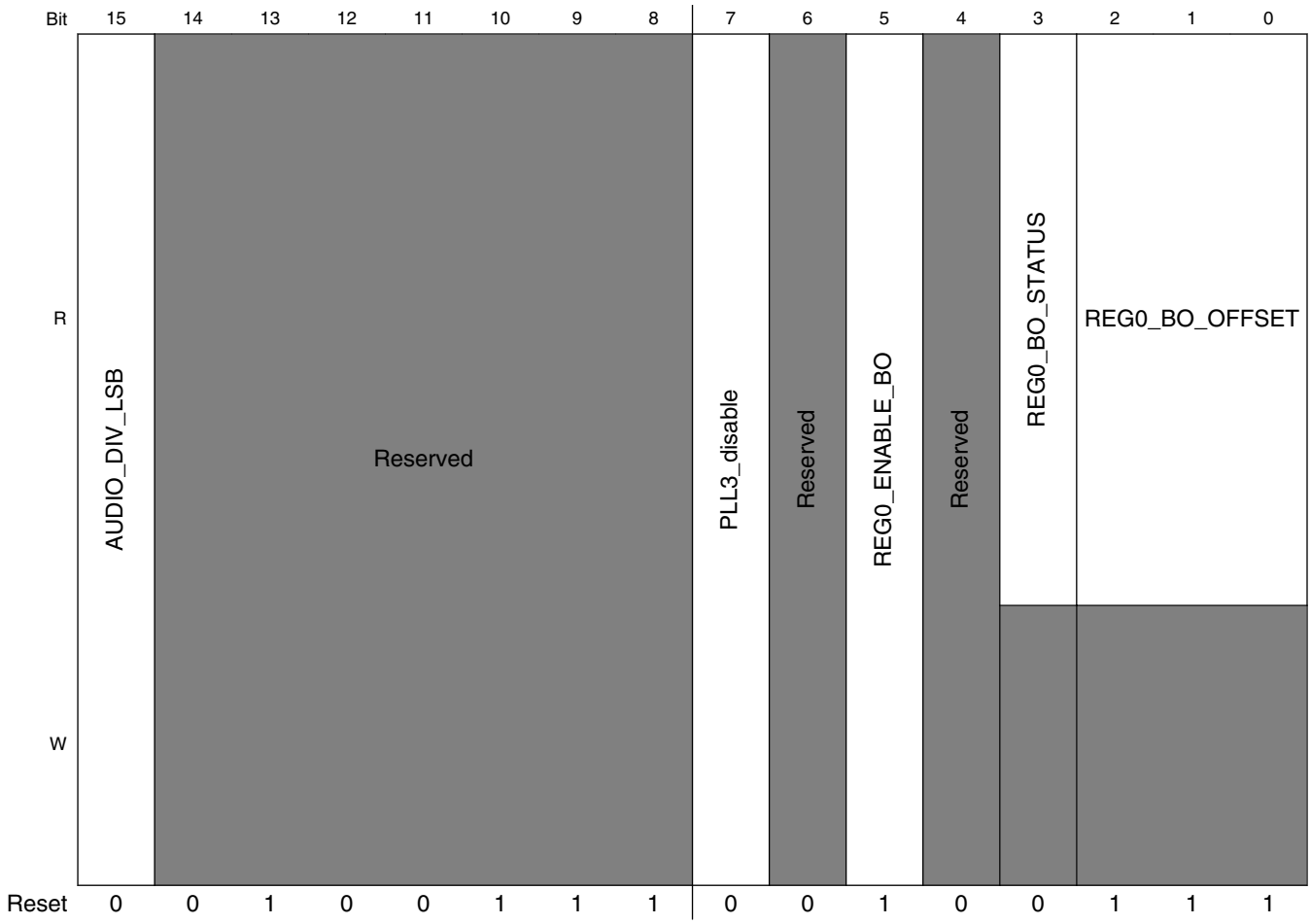
This register defines the control for miscellaneous PMU Analog blocks.

NOTE

This register is shared with CCM.

Address: 20C_8000h base + 170h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VIDEO_DIV		REG2_STEP_TIME		Reserved		REG0_STEP_TIME		AUDIO_DIV_MSB	REG2_OK	REG2_ENABLE_BO	Reserved	REG2_BO_STATUS	REG2_BO_OFFSET		
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1



PMU_MISC2*n* field descriptions

Field	Description
31–30 VIDEO_DIV	Post-divider for video. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_VIDEOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
29–28 REG2_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
27–26 -	This field is reserved. Reserved

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
25–24 REG0_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
23 AUDIO_DIV_MSB	MSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** MSB bit value pertains to the first bit, please program the LSB bit (bit 15) as well to change divider value **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
22 REG2_OK	Signals that the voltage is above the brownout level for the SOC supply. 1 = regulator output > brownout_target
21 REG2_ENABLE_BO	Enables the brownout detection.
20 -	This field is reserved.
19 REG2_BO_STATUS	Reg2 brownout status bit.
18–16 REG2_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V
15 AUDIO_DIV_LSB	LSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** LSB bit value pertains to the last bit, please program the MSB bit (bit 23) as well, to change divider value **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
14–8 -	This field is reserved.

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
7 PLL3_disable	Default value of "0". Should be set to "1" to turn off the USB-PLL(PLL3) in run mode. **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information.
6 -	This field is reserved.
5 REG0_ENABLE_BO	Enables the brownout detection.
4 -	This field is reserved.
3 REG0_BO_STATUS	Reg0 brownout status bit. 1 Brownout, supply is below target minus brownout offset.
REG0_BO_OFFSET	This field defines the brown out voltage offset for the CORE power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V

37.6.8 Low Power Control Register (PMU_LOWPWR_CTRL*n*)

This register defines the low power configuration bits.

Address: 20C_8000h base + 270h offset + (4d × i), where i=0d to 2d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved													0	MIX_PWRGATE	XTALOSC_PWRUP_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	XTALOSC_PWRUP_DELAY		RCOSC_CG_OVERRIDE	-	DISPLAY_PWRGATE	CPU_PWRGATE	L2_PWRGATE	L1_PWRGATE	REFTOP_IBIAS_OFF	LPBG_TEST	LPBG_SEL	OSC_SEL	RC_OSC_PROG			RC_OSC_EN
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1

PMU_LOWPWR_CTRL*n* field descriptions

Field	Description
31–19 -	This field is reserved.
18 Reserved	This read-only field is reserved and always has the value 0.
17 MIX_PWRGATE	Display power gate control. Used as software mask. Set to zero to force ungated.
16 XTALOSC_PWRUP_STAT	Status of the 24MHz xtal oscillator. **NOTE:** Not related to PMU. 0 Not stable 1 Stable and ready to use
15–14 XTALOSC_PWRUP_DELAY	Specifies the time delay between when the 24MHz xtal is powered up until it is stable and ready to use. **NOTE:** Not related to PMU. 00 0.25ms 01 0.5ms 10 1ms 11 2ms
13 RCOSC_CG_OVERRIDE	For debug purposes only. This bit effects clock gating of certain digital logic clocked by the 24MHz clk. **NOTE:** Not related to PMU.
12 -	Reserved

Table continues on the next page...

PMU_LOWPWR_CTRL*n* field descriptions (continued)

Field	Description
11 DISPLAY_ PWRGATE	Display logic power gate control. Used as software override.
10 CPU_PWRGATE	CPU power gate control. Used as software override. **Attention:** Test purpose only
9 L2_PWRGATE	L2 power gate control. Used as software override.
8 L1_PWRGATE	L1 power gate control. Used as software override.
7 REFTOP_IBIAS_ OFF	Low power reftop ibias disable.
6 LPBG_TEST	Low power bandgap test bit.
5 LPBG_SEL	Bandgap select. 0 Normal power bandgap 1 Low power bandgap
4 OSC_SEL	Select the source for the 24MHz clock. **NOTE:** Not related to PMU. 0 XTAL OSC 1 RC OSC
3–1 RC_OSC_PROG	RC osc. tuning values. **NOTE:** Not related to PMU.
0 RC_OSC_EN	RC Osc. enable control. **NOTE:** Not related to PMU. 0 Use XTAL OSC to source the 24MHz clock 1 Use RC OSC

Chapter 38
Pulse Width Modulation (PWM)

38.1 Overview

The Pulse Width Modulation (PWM) has a 16-bit counter, and is optimized to generate sound from stored sample audio images and it can also generate tones. It uses 16-bit resolution and a 4 x 16 data FIFO.

This section presents an overview of the PWM. A block diagram of the PWM module is shown in the figure below.

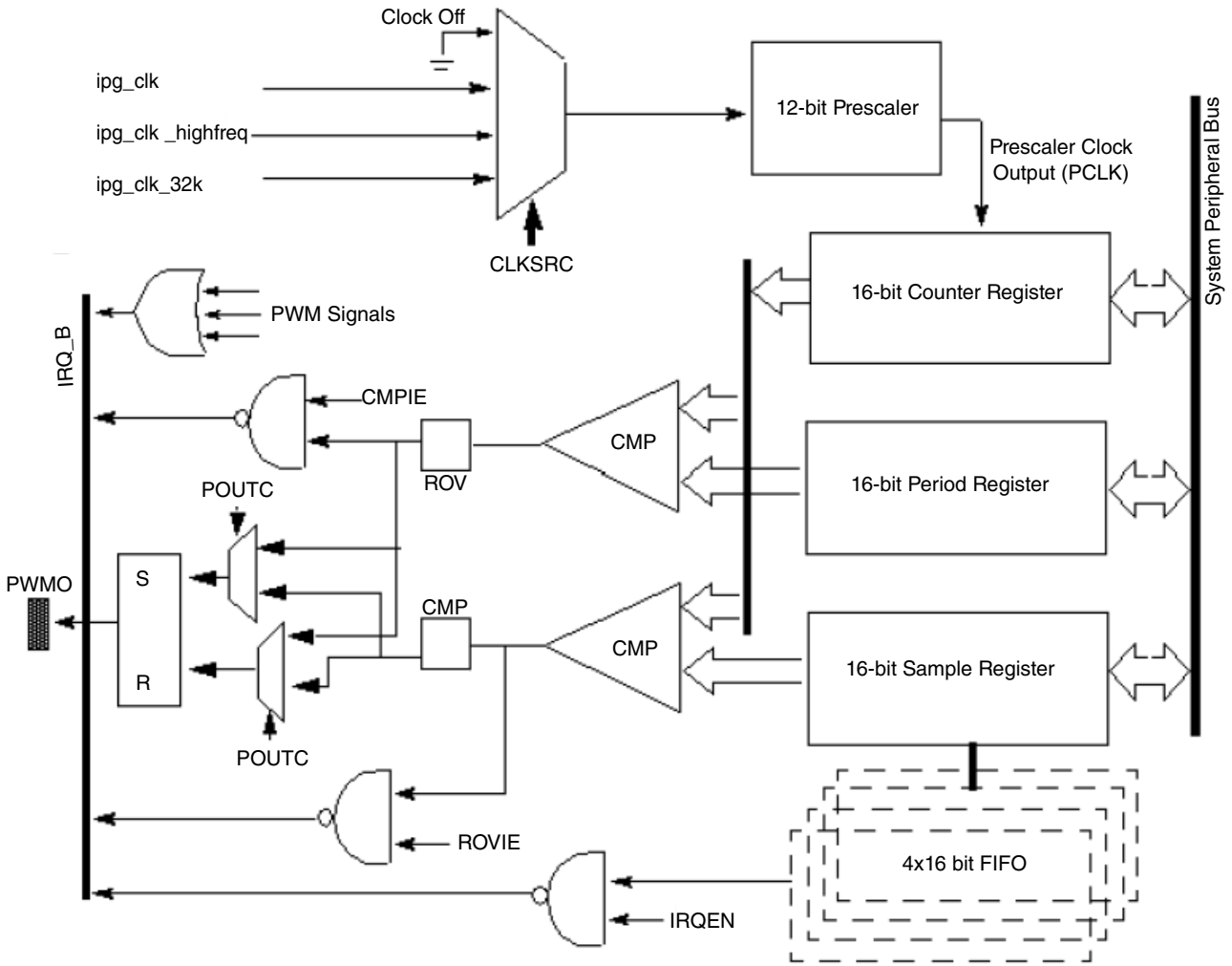


Figure 38-1. Pulse-Width Modulator Block Diagram

The following features characterize the PWM:

- 16-bit up-counter with clock source selection
- 4 x 16 FIFO to minimize interrupt overhead
- 12-bit prescaler for division of clock
- Sound and melody generation
- Active high or active low configured output
- Can be programmed to be active in low-power mode
- Can be programmed to be active in debug mode
- Interrupts at compare and rollover

38.2 External Signals

The PWM follows IP Bus protocol when interfacing with the processor core. PWM does not have any interface signals with any other block inside the chip except for clock and reset inputs from the Clock Control Module (CCM), System Reset Controller (SRC), and interrupt signals to the processor interrupt handler. There is a single output signal.

The following table outlines the external signals.

Table 38-1. PWM External Signals

Signal	Description	Pad	Mode	Direction
PWM1_OUT	This is the PWM1 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_RX_DATA0	ALT2	O
		GPIO1_IO08	ALT0	
		LCD_DATA00	ALT1	
PWM2_OUT	This is the PWM2 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_RX_DATA1	ALT2	O
		GPIO1_IO09	ALT0	
		LCD_DATA01	ALT1	
PWM3_OUT	This is the PWM3 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	GPIO1_IO04	ALT1	O
		LCD_DATA02	ALT1	
		NAND_ALE	ALT3	
PWM4_OUT	This is the PWM4 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	GPIO1_IO05	ALT1	O
		LCD_DATA03	ALT1	
		NAND_WP_B	ALT3	
PWM5_OUT	This is the PWM5 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_TX_DATA1	ALT2	O
		LCD_DATA18	ALT1	
		NAND_DQS	ALT3	
PWM6_OUT	This is the PWM6functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_TX_EN	ALT2	O
		JTAG_TDI	ALT4	
		LCD_DATA19	ALT1	
PWM7_OUT	This is the PWM7 functional output of the PWM. A modulated signal of	CSI_VSYNC	ALT6	O
		ENET1_TX_CLK	ALT2	

Table continues on the next page...

Table 38-1. PWM External Signals (continued)

Signal	Description	Pad	Mode	Direction
	the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	JTAG_TCK	ALT4	
PWM8_OUT	This is the PWM8 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	CSI_HSYNC	ALT6	O
		ENET1_RX_ER	ALT2	
		JTAG_TRST_B	ALT4	

38.3 Clocks

The table found here describes the clock sources for PWM.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 38-2. PWM Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	low-frequency reference clock (32 kHz)
ipg_clk_highfreq	perclk_clk_root	high-frequency reference clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

The clock that feeds the prescaler can be selected from:

- High-frequency reference clock (ipg_clk_highfreq) pat_ref or CKIH

  This is a high frequency clock, provided by the Clock Control Module (CCM). This clock should be on in the low power mode when the ipg_clk is turned off. Thus, the PWM can be run on this clock in the low power mode.

- Low-frequency reference clock (ipg_clk_32k, CKIL)

  This is the 32 KHz low reference clock which is provided by the CCM. This clock should be on in the low power mode when ipg_clk is turned off. Thus, PWM can be run on this clock in the low power mode.

- Peripheral clock (ipg_clk)

This clock should be on in normal operations. In low power mode, it can be switched off.

- Peripheral access clock (ipg_clk_s)

  This clock is used for register read/write.

The clock input source is determined by the PWM control register field PWM_CR[CLKSRC]. The CLKSRC value should only be changed when the PWM is disabled.

A change in the value of the PRESCALER field of the control register is immediately reflected on its output clock frequency.

38.4 Functional Description

The following sections detail the PWM operation and function.

38.4.1 Operation

The output of the PWM is a toggling signal whose frequency and duty cycle can be modulated by programming the appropriate registers. It has a 16-bit up counter which counts from 0x0000 until the counter value equals the PWM_PR + 1. After this match occurs the counter is reset to 0x0000.

At the beginning of a count period cycle, the PWMO pin is set to one (default) and the counter begins counting up from 0x0000. The sample value in the sample FIFO is compared on each count of prescaler clock. When the sample and count values match, the PWMO signal is cleared to zero (default). The counter continues counting until the period match occurs and subsequently another period cycle begins.

When the PWM is enabled, the counter starts running and generates an output with the reset values in the period and sample registers. It is recommended that the programming of these registers be done before PWM is enabled.

A hardware reset results in all the PWM count and sample registers being cleared and the FIFO being flushed. The control register shows that FIFO is empty and it can be written into, and the PWM is disabled. A software reset has the same results, however the state of the STOPEN, DOZEN, WAITEN, and DBGEN bits in the control register are not affected. Software reset can be asserted even when the PWM is in disabled state.

38.4.1.1 FIFO

Digital sample values can be loaded into the pulse-width modulator as 16-bit words. The endianess can be changed using the BCTR and HCTR bits of the control register. A 4-word (16-bit) FIFO minimizes interrupt overhead. A maskable interrupt is generated when the number of data words fall below the water level set by the FWM field in the control register.

A write to the PWM_SAR sample register results in the value being stored into the FIFO if it is not full. A write when the FIFO is full sets FWE (FIFO write error) bit in the status register and the FIFO contents remain unchanged. The FIFO can be written at any time, but can be read only when the PWM is enabled. The PWM_SR[FIFOAV] field shows how many data words are currently contained in the FIFO and whether or not it can be written into.

A read on the sample register yields the current FIFO value that is being used, or will be used, by the PWM for generation on the output signal. Therefore, a write and a subsequent read on the sample register may result in different values being obtained.

38.4.1.2 Rollover and Compare Event

The counter is reset to 0x0000 after its value equals the PWM_PR[PERIOD] + 1 and resumes counting thereafter. This event is referred to as a rollover. For example, if PWM_PR[PERIOD] = 0x0000, the counter is reset when it equals 0x0001. When PWM_PR[PERIOD] = 0xFFFF or 0xFFFE, the counter is reset when it equals 0xFFFF. For more information, see the PWM Period Register (PWM_PR) description.

During a rollover event the output is either set (default), reset or has no effect according to the programming of the POUTC field in the control register. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

When the counter value reaches the sample value, the output of the PWM is reset (default), set or has no effect according to the programming of the POUTC field of control register. This event is referred to as a compare event. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

If the rollover event sets the PWM output signal, the compare event will reset it and vice versa for a particular programming configuration of POUTC field.

38.4.1.3 Low Power Mode Behavior

In low power mode, if the clock from the selected clock source is available, the PWM counter continues to run and an output is produced, depending on whether the control bit for that mode is set or not. In the absence of the clock itself, or if the corresponding low power bit in the control register is 0, the counter is reset and resumes counting when it exits the low power mode.

38.4.1.4 Debug Mode Behavior

In debug mode, PWM has the option of continuing to run or be halted. If the DBGEN bit is not set in the PWM_PWMCR, the PWM is halted. If the DBGEN bit is set, then the PWM will continue to run in the debug mode.

38.5 Enable Sequence for the PWM

The sequence found here should be used to enable the PWM.

1. Configure the desired settings for the PWM Control Register (PWMx_PWMCR) while keeping the PWM disabled (PWMx_PWMCR[0]=0).
2. Enable the desired interrupts in the PWM Interrupt Register (PWMx_PWMIR).
3. One to three initial samples may be written to the PWM Sample Register (PWMx_PWMSAR). The initial sample values will be loaded into the PWM FIFO even if the PWM is not yet enabled. Do not write a 4th sample because the FIFO will become full and trigger a FIFO Write Error (FWE). This error will prevent the PWM from starting once it is enabled.
4. Check the FIFO Write Error status bit (FWE), the Compare status bit (CMP) and the Roll-over status bit (ROV) in the PWM Status Register (PWMx_PWMSR) to make sure they are all zero. Any non-zero status bits should be cleared by writing a 1 to them.
5. Write the desired period to the PWM Period Register (PWMx_PWMPR).
6. Enable the PWM by writing a 1 to the PWM Enable bit, PWMx_PWMCR[0], while maintaining the other register bits in their previously configured state.

38.6 Disable Sequence for the PWM

The PWM can be disabled at any time by clearing the PWM enable bit, PWMx_PWMCR[0] to 0.

Any data remaining in the FIFO will not be produced at the PWM output after the PWM has been disabled and will remain in the FIFO until the PWM is enabled again. A software reset (setting PWMx_PWMCR[3] to 1) or a hardware reset will clear the FIFO and any remaining data will be lost.

38.7 PWM Memory Map/Register Definition

The PWM includes six user-accessible 32-bit registers.

PWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
208_0000	PWM Control Register (PWM1_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
208_0004	PWM Status Register (PWM1_PWMSR)	32	w1c	0000_0008h	38.7.2/2306
208_0008	PWM Interrupt Register (PWM1_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
208_000C	PWM Sample Register (PWM1_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
208_0010	PWM Period Register (PWM1_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
208_0014	PWM Counter Register (PWM1_PWMCNR)	32	R	0000_0000h	38.7.6/2310
208_4000	PWM Control Register (PWM2_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
208_4004	PWM Status Register (PWM2_PWMSR)	32	w1c	0000_0008h	38.7.2/2306
208_4008	PWM Interrupt Register (PWM2_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
208_400C	PWM Sample Register (PWM2_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
208_4010	PWM Period Register (PWM2_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
208_4014	PWM Counter Register (PWM2_PWMCNR)	32	R	0000_0000h	38.7.6/2310
208_8000	PWM Control Register (PWM3_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
208_8004	PWM Status Register (PWM3_PWMSR)	32	w1c	0000_0008h	38.7.2/2306
208_8008	PWM Interrupt Register (PWM3_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
208_800C	PWM Sample Register (PWM3_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
208_8010	PWM Period Register (PWM3_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
208_8014	PWM Counter Register (PWM3_PWMCNR)	32	R	0000_0000h	38.7.6/2310
208_C000	PWM Control Register (PWM4_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
208_C004	PWM Status Register (PWM4_PWMSR)	32	w1c	0000_0008h	38.7.2/2306
208_C008	PWM Interrupt Register (PWM4_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
208_C00C	PWM Sample Register (PWM4_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
208_C010	PWM Period Register (PWM4_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
208_C014	PWM Counter Register (PWM4_PWMCNR)	32	R	0000_0000h	38.7.6/2310
20F_0000	PWM Control Register (PWM5_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
20F_0004	PWM Status Register (PWM5_PWMSR)	32	w1c	0000_0008h	38.7.2/2306

Table continues on the next page...

PWM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20F_0008	PWM Interrupt Register (PWM5_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
20F_000C	PWM Sample Register (PWM5_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
20F_0010	PWM Period Register (PWM5_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
20F_0014	PWM Counter Register (PWM5_PWMCNR)	32	R	0000_0000h	38.7.6/2310
20F_4000	PWM Control Register (PWM6_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
20F_4004	PWM Status Register (PWM6_PWMSR)	32	w1c	0000_0008h	38.7.2/2306
20F_4008	PWM Interrupt Register (PWM6_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
20F_400C	PWM Sample Register (PWM6_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
20F_4010	PWM Period Register (PWM6_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
20F_4014	PWM Counter Register (PWM6_PWMCNR)	32	R	0000_0000h	38.7.6/2310
20F_8000	PWM Control Register (PWM7_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
20F_8004	PWM Status Register (PWM7_PWMSR)	32	w1c	0000_0008h	38.7.2/2306
20F_8008	PWM Interrupt Register (PWM7_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
20F_800C	PWM Sample Register (PWM7_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
20F_8010	PWM Period Register (PWM7_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
20F_8014	PWM Counter Register (PWM7_PWMCNR)	32	R	0000_0000h	38.7.6/2310
20F_C000	PWM Control Register (PWM8_PWMCR)	32	R/W	0000_0000h	38.7.1/2304
20F_C004	PWM Status Register (PWM8_PWMSR)	32	w1c	0000_0008h	38.7.2/2306
20F_C008	PWM Interrupt Register (PWM8_PWMIR)	32	R/W	0000_0000h	38.7.3/2307
20F_C00C	PWM Sample Register (PWM8_PWMSAR)	32	R/W	0000_0000h	38.7.4/2308
20F_C010	PWM Period Register (PWM8_PWMPR)	32	R/W	0000_FFFEh	38.7.5/2309
20F_C014	PWM Counter Register (PWM8_PWMCNR)	32	R	0000_0000h	38.7.6/2310

38.7.1 PWM Control Register (PWMx_PWMCR)

The PWM control register (PWM_PWMCR) is used to configure the operating settings of the PWM. It contains the prescaler for the clock division.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				FWM		STOPEN	DOZEN	WAITEN	DBGEN	BCTR	HCTR	POUTC		CLKSRC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

PWMx_PWMCR field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 FWM	FIFO Water Mark. These bits are used to set the data level at which the FIFO empty flag will be set and the corresponding interrupt generated 00 FIFO empty flag is set when there are more than or equal to 1 empty slots in FIFO 01 FIFO empty flag is set when there are more than or equal to 2 empty slots in FIFO 10 FIFO empty flag is set when there are more than or equal to 3 empty slots in FIFO 11 FIFO empty flag is set when there are more than or equal to 4 empty slots in FIFO
25 STOPEN	Stop Mode Enable. This bit keeps the PWM functional while in stop mode. When this bit is cleared, the input clock is gated off in stop mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in stop mode 1 Active in stop mode
24 DOZEN	Doze Mode Enable. This bit keeps the PWM functional in doze mode. When this bit is cleared, the input clock is gated off in doze mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in doze mode 1 Active in doze mode
23 WAITEN	Wait Mode Enable. This bit keeps the PWM functional in wait mode. When this bit is cleared, the input clock is gated off in wait mode. This bit is not affected by software reset. It is cleared by hardware reset.

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
	0 Inactive in wait mode 1 Active in wait mode
22 DBGEN	Debug Mode Enable. This bit keeps the PWM functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in debug mode 1 Active in debug mode
21 BCTR	Byte Data Swap Control. This bit determines the byte ordering of the 16-bit data when it goes into the FIFO from the sample register. 0 byte ordering remains the same 1 byte ordering is reversed
20 HCTR	Half-word Data Swap Control. This bit determines which half word data from the 32-bit IP Bus interface is written into the lower 16 bits of the sample register. 0 Half word swapping does not take place 1 Half words from write data bus are swapped
19–18 POUTC	PWM Output Configuration. This bit field determines the mode of PWM output on the output pin. 00 Output pin is set at rollover and cleared at comparison 01 Output pin is cleared at rollover and set at comparison 10 PWM output is disconnected 11 PWM output is disconnected
17–16 CLKSRC	Select Clock Source. These bits determine which clock input will be selected for running the counter. After reset the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the PWM is disabled 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
15–4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the value by which the clock will be divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 0xfff Divide by 4096
3 SWR	Software Reset. PWM is reset when this bit is set to 1. It is a self clearing bit. A write 1 to this bit is a single wait state write cycle. When the block is in reset state this bit is set and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values except for the STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 PWM is out of reset 1 PWM is undergoing reset
2–1 REPEAT	Sample Repeat. This bit field determines the number of times each sample from the FIFO is to be used. 00 Use each sample once 01 Use each sample twice 10 Use each sample four times 11 Use each sample eight times

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
0 EN	PWM Enable. This bit enables the PWM. If this bit is not enabled, the clock prescaler and the counter is reset. When the PWM is enabled, it begins a new period, the output pin is set to start a new period while the prescaler and counter are released and counting begins. To make the PWM work with softreset and disable/enable, users can do software reset by seting the SWR bit, wait software reset done, configure the registers, and then enable the PWM by setting this bit to "1" Users can also disable/enable the PWM if PWM would like to be stopped and resumed with same registers configurations . 0 PWM disabled 1 PWM enabled

38.7.2 PWM Status Register (PWMx_PWMSR)

The PWM status register (PWM_PWMSR) contains seven bits which display the state of the FIFO and the occurrence of rollover and compare events. The FIFOAV bit is read-only but the other four bits can be cleared by writing 1 to them. The FE, ROV, and CMP bits are associated with FIFO-Empty, Roll-over, and Compare interrupts, respectively.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0									FWE	CMP	ROV	FE	FIFOAV		
W										w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

PWMx_PWMSR field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FWE	FIFO Write Error Status. This bit shows that an attempt has been made to write FIFO when it is full. 0 FIFO write error not occurred 1 FIFO write error occurred
5 CMP	Compare Status. This bit shows that a compare event has occurred. 0 Compare event not occurred 1 Compare event occurred

Table continues on the next page...

PWMx_PWMSR field descriptions (continued)

Field	Description
4 ROV	Roll-over Status. This bit shows that a roll-over event has occurred. 0 Roll-over event not occurred 1 Roll-over event occurred
3 FE	FIFO Empty Status Bit. This bit indicates the FIFO data level in comparison to the water level set by FWM field in the control register. 0 Data level is above water mark 1 When the data level falls below the mark set by FWM field
FIFOAV	FIFO Available. These read-only bits indicate the data level remaining in the FIFO. An attempted write to these bits will not affect their value and no transfer error is generated. 000 No data available 001 1 word of data in FIFO 010 2 words of data in FIFO 011 3 words of data in FIFO 100 4 words of data in FIFO 101 unused 110 unused 111 unused

38.7.3 PWM Interrupt Register (PWMx_PWMIR)

The PWM Interrupt register (PWM_PWMIR) contains three bits which control the generation of the compare, rollover and FIFO empty interrupts.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

PWMx_PWMIR field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CIE	Compare Interrupt Enable. This bit controls the generation of the Compare interrupt.

Table continues on the next page...

PWMx_PWMIR field descriptions (continued)

Field	Description
	0 Compare Interrupt not enabled 1 Compare Interrupt enabled
1 RIE	Roll-over Interrupt Enable. This bit controls the generation of the Rollover interrupt. 0 Roll-over interrupt not enabled 1 Roll-over Interrupt enabled
0 FIE	FIFO Empty Interrupt Enable. This bit controls the generation of the FIFO Empty interrupt. 0 FIFO Empty interrupt disabled 1 FIFO Empty interrupt enabled

38.7.4 PWM Sample Register (PWMx_PWMSAR)

The PWM sample register (PWM_PWMSAR) is the input to the FIFO. 16-bit words are loaded into the FIFO. The FIFO can be written at any time, but can be read only when the PWM is enabled. The PWM will run at the last set duty-cycle setting if all the values of the FIFO has been utilized, until the FIFO is reloaded or the PWM is disabled. When a new value is written, the duty cycle changes after the current period is over.

A value of zero in the sample register will result in the PWMO output signal always being low/high (POUTC = 00 it will be low and POUTC = 01 it will be high), and no output waveform will be produced. If the value in this register is higher than the PERIOD + 1, the output will never be set/reset depending on POUTC value.

Address: Base address + Ch offset

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

PWMx_PWMSAR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SAMPLE	Sample Value. This is the input to the 4x16 FIFO. The value in this register denotes the value of the sample being currently used.

38.7.5 PWM Period Register (PWMx_PWMPR)

The PWM period register (PWM_PWMPR) determines the period of the PWM output signal. After the counter value matches PERIOD + 1, the counter is reset to start another period.

PWMO (Hz) = PCLK(Hz) / (period +2)

A value of zero in the PWM_PWMPR will result in a period of two clock cycles for the output signal. Writing 0xFFFF to this register will achieve the same result as writing 0xFFFE.

A change in the period value due to a write in PWM_PWMPR results in the counter being reset to zero and the start of a new count period.

NOTE

Settings PWM_PWMPR to 0xFFFF when PWMx_PWMCR REPEAT bits are set to non-zero values is not allowed.

Address: Base address + 10h offset

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

PWMx_PWMPR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PERIOD	Period Value. These bits determine the Period of the count cycle. The counter counts up to [Period Value] +1 and is then reset to 0x0000.

38.7.6 PWM Counter Register (PWMx_PWMCNR)

The read-only pulse-width modulator counter register (PWM_PWMCNR) contains the current count value and can be read at any time without disturbing the counter.

Address: Base address + 14h offset

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

PWMx_PWMCNR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Counter Value. These bits are the counter register value and denotes the current count state the counter register is in.

Chapter 39
Pixel Pipeline (PXP)

39.1 Overview

This document describes the micro-architecture for the Pixel Processing Pipeline used to process graphics buffers or composite video and graphics data before sending to an LCD display or TV encoder.

It is used to minimize the memory footprint required for the display pipeline and provide an area and performance optimized to both SDRAM-less and SRAM-based systems.

The PXP integrates of several independent processing stages into a cohesive strategy to create flexible pixel pipeline.

The PXP combines scaling, color space conversion (or CSC), alpha-blending, secondary color space conversion (or CSC2), pixel conversion lookup memory table (or LUT) and rotation into a single processing engine, as shown in Figure 39-1. By integrating multiple blocks, intermediate buffer operations to external memory are removed, reducing external memory bandwidth, power, and software control complexity.

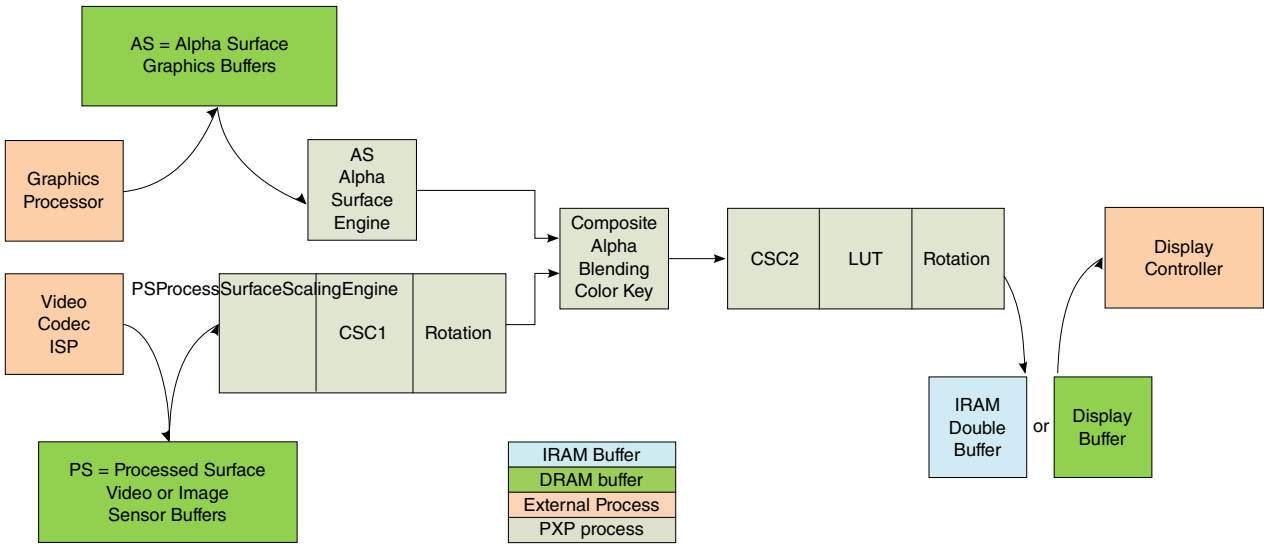


Figure 39-1. PXP Architecture

39.2 Clocks

The following table describes the clock sources for PXP. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 39-1. PXP Clocks

Clock name	Clock Root	Description
clk	axi_clk_root	PXP clock

39.3 Top-level architecture

The PXP consist of several pipelined blocks that perform the video source frame scaling, color space conversion, alpha-blending/color key algorithm, secondary CSC, pixel correction.

The entire pipeline operate within the requirements of the PXP architecture, and perform operations on either 8x8 or 16x16 pixel blocks in the representative source buffers. The entire pipeline operate within the context of two iteration counters that iterate through the appropriate grid of input blocks to produce the rotated output grid blocks in scan-line order.

Figure 39-1 shows the high-level architecture of the scaling, color space conversion, blending, pixel correction,rotation engines.The Alpha Surface Engine fetches one RGB graphics plane alpha surface (AS). The scaling engine fetches a single processed surface (PS), which can be blended with the AS surface.Although the PXP processes NxN pixel macro blocks, each of the AS or PS surfaces can have any pixel alignment within the output buffer. There are no restrictions and any pixel coordinates within the output buffer are valid. The upper left origin of the output buffer is defined as pixel 0,0. The upper left and lower right coordinates for each of the AS and PS are inclusive within the output buffer.

Figure 39-2 represents a sample output buffer configuration with both an AS and PS included. The alignment of each AS and PS within the output buffer can be at any arbitrary pixel locations. For example, the PS has an upper left coordinate (ULC) of 2,2 and a lower right coordinate (LRC) at pixel 13,13. The maximum value for the ULC and LRC for each of the AS and PS is bounded by the LRC of the output buffer, 15,15 for this example.

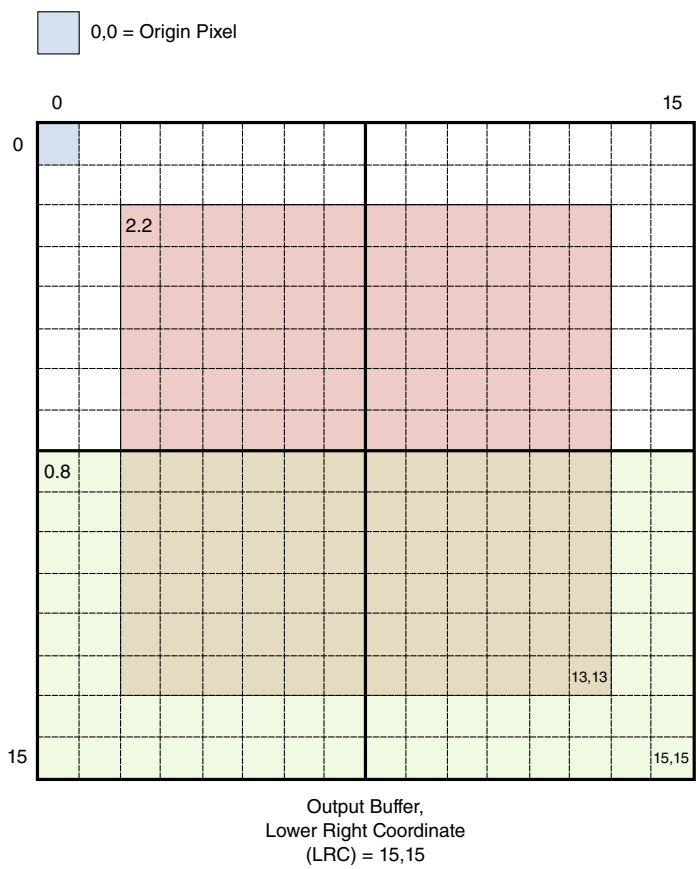


Figure 39-2. Sample output buffer configuration

The AS engine supports RGB pixel formats, and the PS engine supports RGB, YUV, and YCbCr pixel formats. The CSC1 can be used to convert to RGB pixel formats so that the PS surface can be blended with the AS surfaces in the compositing engine in the RGB color space. There is a single rotation engine in the PXP with a programmable location with two possible rotations, one after PS processing, other just before the transfer to display buffer. Rotation can occur after image composition or at the output of the PS engine. In the first scenario, all the data produced by the AS and PS engines is rotated. When the rotation module is programmed to rotate only PS images, the AS is not rotated, and AS pixels are combined with rotated PS surfaces. The CSC2 unit can convert to any RGB, YUV, or YCbCr color spaces for final output. Pixels can be corrected using a programmable LUT resource to achieve any desired pixels effects. For detailed information, please refer to Output Modes to RGBW4444CFA.

39.3.1 Processing Details

The PXP architecture has been driven primarily by the requirement that the output buffer must be processed and rotated without intermediate frame buffer stored in external memory.

This reduces the use of external memory bandwidth requirements thus reducing overall system power consumed.

Since the output of the rotation block must be NxN pixel blocks in scan order, the entire pipeline will operate on NxN pixel blocks. In essence, the pipeline will be able to operate on blocks in a random access fashion, but the entire pipeline will operate within the context of two iteration counters that will iterate through the horizontal and vertical input blocks to generate the required output block.

Processing Pipeline

The control block will coordinate the processing of the pixel blocks within the source and destination image buffers. It begins by issuing a command to each stage of the pipeline requesting that operations be done for the block at offset x, y. When the block accepts the command, it asserts its acknowledge signal for a single cycle to indicate the acceptance and allow the control unit to move to the next block.

When the PS and AS fetch engines have received a command, they will fetch the required data and place it into their fetch buffers. If compositing the RGB AS surface with the PS surface, then the output of the PS engine needs to be converted to the RGB color space using CSC1, since all compositing occurs in the RGB color space. For YUV output pixel formats, the CSC1 unit can be enabled to convert pixels into the RGB space for subsequent compositing with AS pixels. Then, the CSC2 module can convert the resulting pixels back into the YUV output color space. If the final output color space is

YUV and there is no compositing required (AS not present, for example), then both the CSC units can be bypassed and the pixel data path will pass the YUV pixels to the rotation engine. For YUV output formats, scaling operations, LUT, and rotation operations are still valid, but blending RGB AS surfaces with YUV PS surfaces is NOT supported. The two CSC units in the overall pixel data path must be used to achieve the desired source frame compositing and output pixel formatting.

The alpha blender/color key module will process a pixel any time that both inputs present valid data.

A handshake will be created between each stage and a pipeline controller to handle the advance of the pipeline and generation of the iteration counters. The pipeline controller will also maintain the interlocks with the LCD interface for the case where the LCD display and pixel processing pipeline use the SRAM to maintain the double buffer block intermediate buffer.

39.3.2 Scaling Operation

The scaling engine operates on YUV (or YCbCr) 422 or 420 and any RGB formatted pixels. Each color plane is sourced from color planes indicated by different base address registers.

The scaling source data can be stored as 3 individual planes for each Y, U, and V data, stored as two planes as a single Y and interleaved UV plane, or stored as a single plane with YUV/RGB interleaved on a per byte basis.

The scaled output image is presented to the CSC module as YUV444 or RGB888 pixels with a single byte for each color channel. The scaler can reduce an input image by a maximum factor of 16. In this case, the output image will be 1/16 the dimension of the input image in each of the X and Y axis. There are no limits, essentially, on increasing the source image size. The theoretical maximum increase is 4096 since a 12 bit fractional step function is used when scaling an input image. Scaling in either axis, X or Y is independent, so a source image can appear stretched in either direction.

All source images pass through the scale engine. The PXP alpha blend module and AS pixel streams are in the RGB888 format, so PS pixel buffers must be converted to the RGB888 format for alpha blending. The scaling engine works with the CSC1 module to translate YUV/YCbCr pixel formats to RGB888 for output frame buffer compositing using the alpha blender. The CSC2 module can by bypassed or enabled to convert pixels to any output color space. In the case of processing RGB pixels in the PS engine, the CSC1 unit can be bypassed so compositing can occur in the alpha engine.

The scaling operation is divided into two scaling steps. The first step is a decimation scaler, and the second step is a bilinear filter. The decimation filter provide a maximum down scaling factor of 8, and the subsequent bilinear filter provides a maximum scaling factor of 2. Combined, the maximum scaling factor can be up to 16. The decimation and bilinear scaling engines are independently programmable. There is also an initial offset that is programmable to allow more source data to be considered in the bilinear scaling engine.

39.3.3 Decimation Image Scaling

The first of two scaling engines is the decimation filter.

The intent of the decimation filter is to use as much source data as is possible to create the output image frame buffer. The decimation filter simply discards certain pixels from the source PS image depending on the reduction selected.

For RGB pixel formats, each color channel is treated equally since there is the same amount of pixel data within each color plane. For YUV422/420 formats, the chroma samples are already subsampled by 2. In these decimation scenarios, the chroma decimation factor is adjusted to account for the pre-decimation of the chroma samples. For example, since YUV422 is already sub-sampled by 2 horizontally, an X decimation factor of 2 does not apply to the YUV422 pixels in the X direction. All the chroma samples are passed on to the bilinear filter in this case. As another example, an X decimation factor of 4 will decimate the chroma samples by 2, since this factor combined with the pre-decimation factor of 2 in the pixel source buffers totals an overall decimation factor of 4.

The following example will show which pixels (in green) in a source RGB buffer that are passed to the bilinear filter for an X decimation factor of 2 and a Y decimation factor of 4. All pixels coincident with dashed lines are discarded.

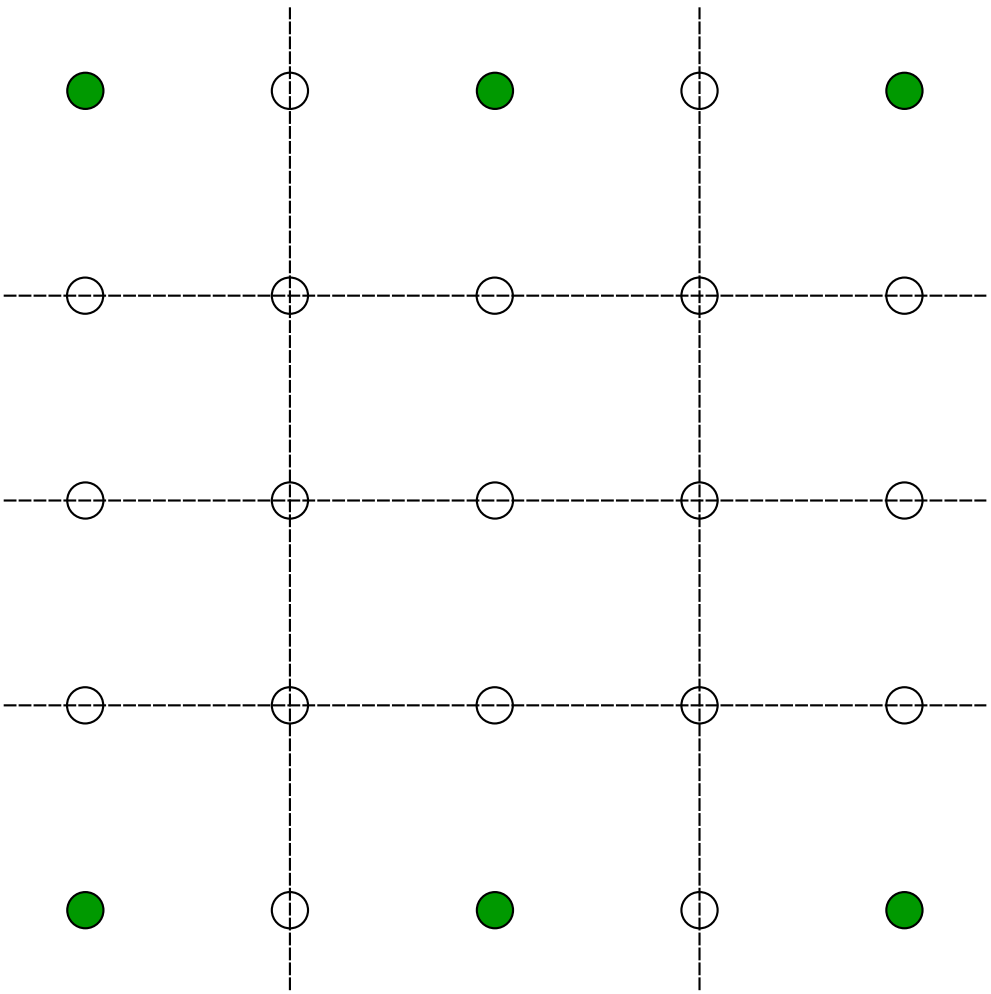


Figure 39-3. RGB decimation X /2, Y /4

Using the same decimation factor as the above scenario for RGB pixels, but using YUV420 source buffers, it can be shown that the decimation factor for the Y and UV components of data are decimated differently. This is due to the pre-decimation of the chroma samples in the source frame buffers. Figure 4: YUV420 decimation X /2, Y /4 indicates that the U/V samples in the X direction are not decimated, but the Y samples in the X direction are decimated by the factor of 2.

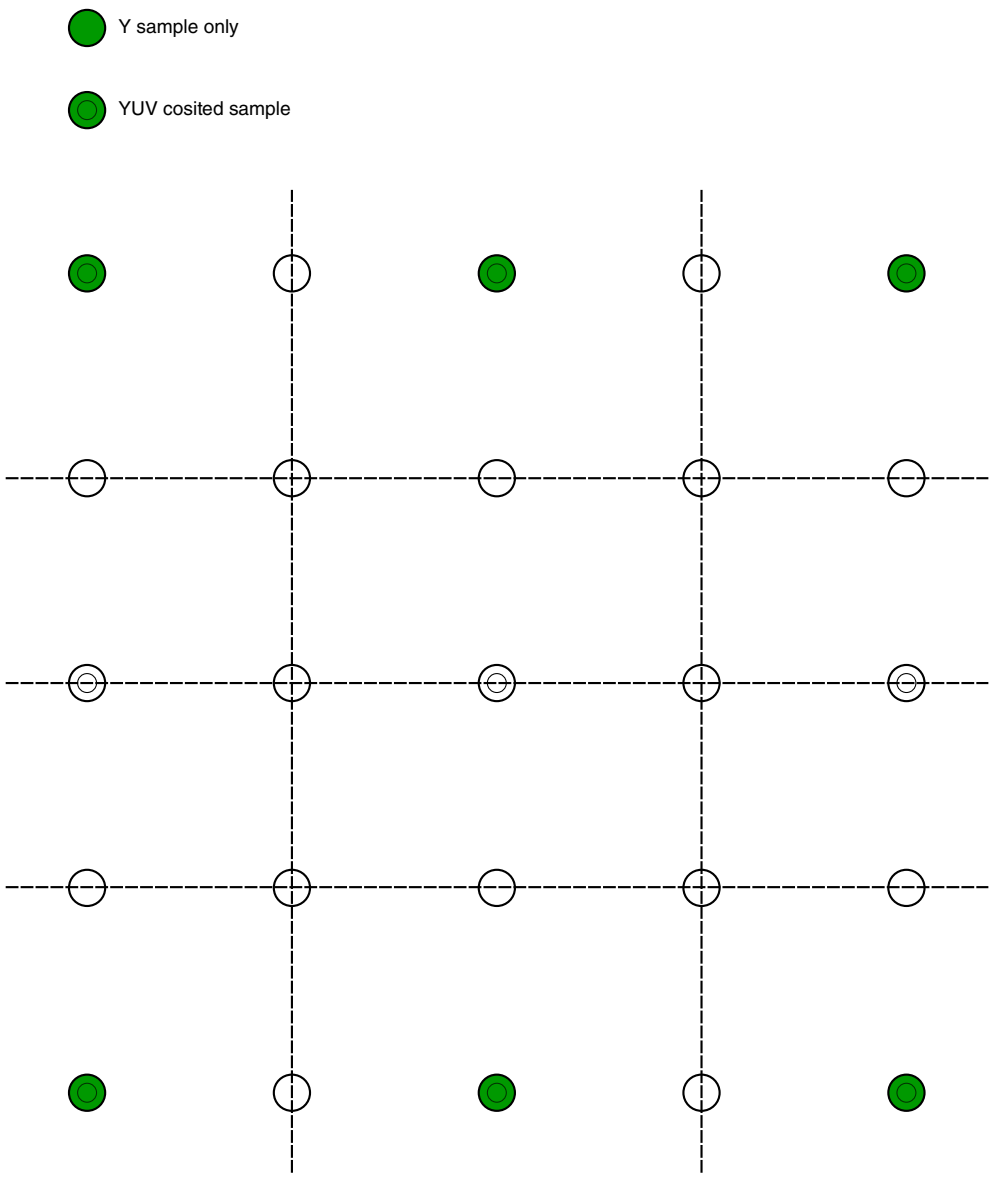


Figure 39-4. YUV420 decimation X /2, Y /4

39.3.4 Bilinear Image Scaling Filter

The PXP implements a bilinear scaling filter to resize an input image to a different resolution for display output.

The bilinear filter is a weighted average of the four nearest pixels that can be sourced to approximate the pixel in the output frame buffer.

When scaling YUV data, the UV values are offset by 0x80 (top bit inverted) to shift the signed UV bits into an unsigned equivalent with a range of 0 to 255. YCbCr data does not have to be shifted since it is defined as an unsigned byte. The REG_CSC1_COEF0[YCBCR_MODE] bit controls whether this operation is applied to the input UV bytes.

After scaling, the offset is removed so that the range for UV data is signed from -128 to 127.

The reason for this adjustment is based on the implementation of an unsigned scaling engine, and therefore, is to ensure that the scaled values are handled properly. Consider the following table:

Format	pixel0	pixel1	average	Result
decimal	-2	+2	0	Correct
CbCr	0x7E	0x82	0x80	Correct (0x80 is 0 in CbCr)
UV	0xFE	0x02	0x80	Incorrect (0x80 is -128 in UV)
decimal	-32	+16	-8	Correct
CbCr	0x60	0x90	0x78	Correct (0x78 is -8 in CbCr)
UV	0xE0	0x10	0x78	Incorrect (0x78 is +120 in UV)

To compute the output pixel value at position as indicated by P, consider the diagram below.

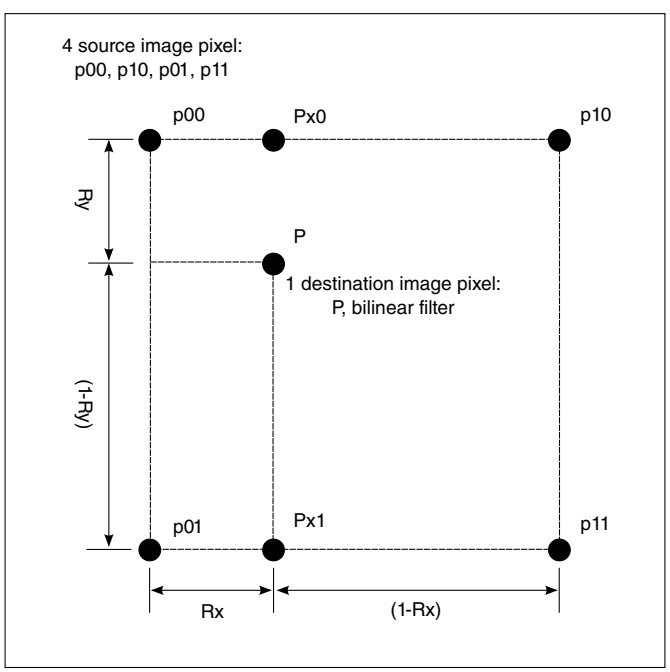


Figure 39-5. Output Pixel Value

A step function is used to indicate the position of the pixel "P" in the output frame. This position may not coincide with a single pixel position in the input frame buffer. In this case, the four closest pixels in the input frame are used to approximate the value of the pixel in the output frame.

The PXP scaler first computes a linear filter in the X axis to create the two intermediate pixel values Px0 and Px1. The step function's X fractional component is used to provide the weighting factor for blending p00 with p10 to provide Px0. Likewise, Px1 is also derived from a linear filter using p01 and p11.

The equations for Px0 and Px1 are as follows:

Px0 = p00*(1-Rx) + p10*Rx

Px1 = p01*(1-Rx) + p11*Rx

The PXP scaler uses the intermediate X pixels Px0 and Px1 and implements a bilinear filter on these two pixel values to produce the final pixel value at position P. The remainder of the step function for the Y axis is used to compute the weighted average pixel result. The equation for final filtered pixel is:

P = Px0*(1-Ry) + Px1*Ry

39.3.5 YUV 4:2:2 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:2 formats. There are twice as many Y luma samples then U and V chroma samples horizontally.

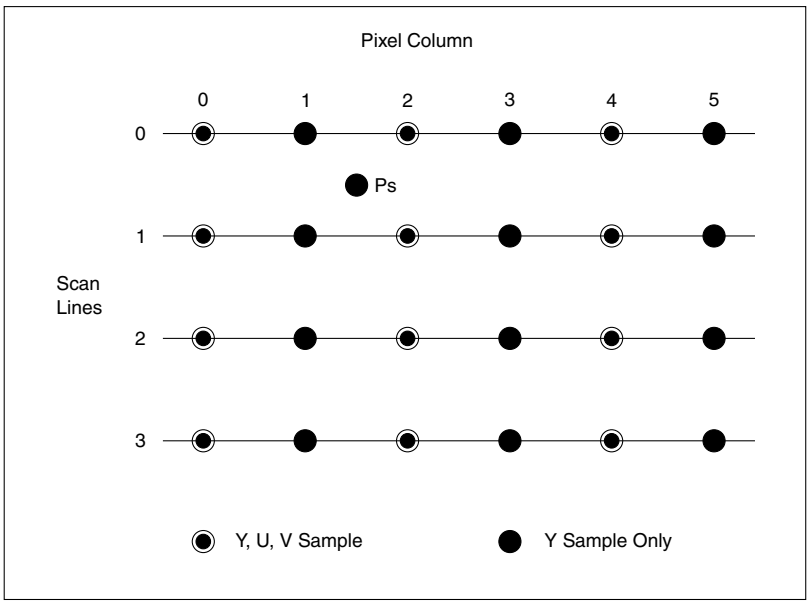


Figure 39-6. YUV Sample Positioning, 4:2:2

Consider the scaled output pixel Ps (pixel scaled) which has an accumulated step function of X=1.5 and Y=0.5. The remainder for the step function is Rx = 0.5 and Ry = 0.5. Or, the sub pixel position of output pixel Ps is half way between line 0 and 1 and half way between column 1 and 2.

The Y output component of Ps is simply the bilinear function of the four nearest Y samples from the input image. Specifically, the Y values at [1,0], [2,0], [1,1], and [2,1] are used to compute the Y for Ps.

For the U and V components of Ps, there are no samples present in the column position 1. The bilinear filter uses chroma components located at [0,0], [2,0], [0,1] and [2,1]. Since the chroma components are not sub sampled vertically, the remainder used to combine pixels vertically is Ry=0.5 (the same as for Y). However, horizontally, the scaling engine shifts the remainder by a factor of 2. So an X axis step function value of X=1.5 has a remainder Rx=0.75. Source chroma values are not replicated, they are completely interpolated using the four nearest chroma samples to approximate U and V at Ps.

39.3.6 YUV 4:2:0 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:0 formats. Chroma is sub sampled both horizontally and vertically. In this format, the chroma frame buffers contain ¼ the data that the luma frame buffers store.

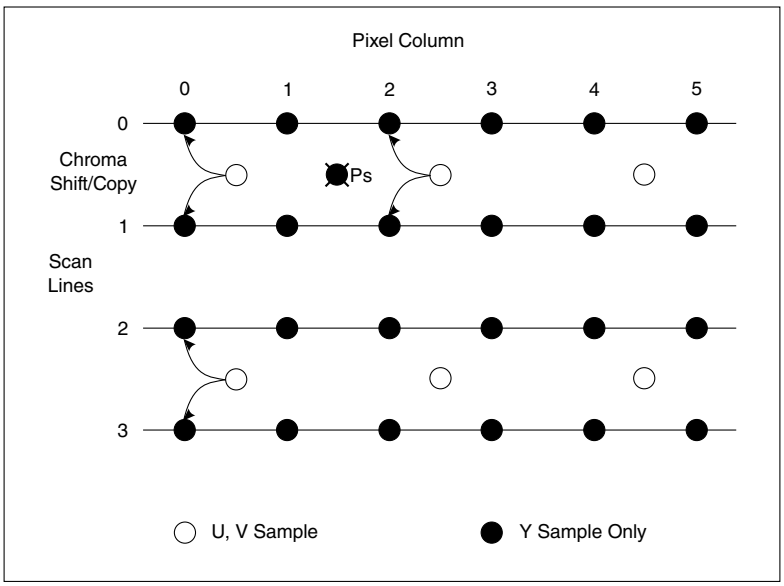


Figure 39-7. YUV Sample Positioning, 4:2:0

The Y output component for all scaled pixels in 4:2:0 formats are the same as for the 4:2:2 pixel formats.

The U and V output components have two considerations when computing the output pixel Ps.

1. All chroma samples from the input source image are shifted left and up by ½ a sample position of the input pixel matrix.
2. Odd scan lines are replicated using the previous even chroma scan line values. So, output image chroma values that map between even to odd scan lines are replicated in the vertical axis. In contrast, output image chroma values between odd to even scan lines are interpolated vertically.

The chroma values are interpolated horizontally as in the 4:2:2 pixel format.

As an example, consider the interpolated pixel Ps in the 4:2:0 diagram above. For the Y component, the interpolated output luma is a function of the Y values in the source frame buffer at position [1,0], [2,0], [1,1], [2,1].

For the U and V interpolated samples, the chroma values on scan line position 0.5 are shifted so that they coincide with the even luma sample points. They are also replicated so that a single chroma scan line is used twice. The chroma scan line at 0.5 is replicated to represent the 4:2:2 sample points for scan line 0 and 1. The chroma scan line at 2.5 is replicated to represent the 4:2:2 sample points for scan line 2 and 3. This pattern of chroma replication occurs for the entire source frame buffer during the scaling operation.

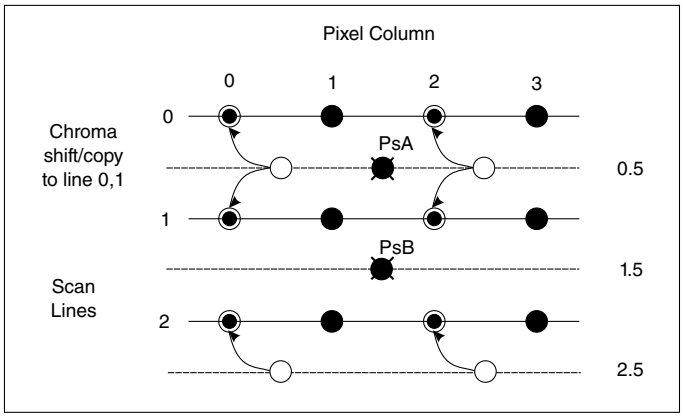


Figure 39-8. Scaled Chroma Computation Examples

The preceding diagram has two examples for the computation of the scaled chroma output pixel. For chroma at output position PsA (vertical position 0.5), interpolation occurs in the X axis using chroma values at column 0 and column 2. However, since line 0 and line 1 have equal chroma values due to chroma line replication, scaling in the Y axis results in replication of chroma values.

For chroma at output position PsB (vertical position 1.5), interpolation occurs in both the X and Y axis. The Y axis is an interpolation since the chroma values copied to scan line 1 and 2 and not the same.

In summary, any output image pixels that map to an odd scan line above and an even scan line below are interpolated vertically. Output image pixels that map to an even scan line above and an odd scan line below are replicated vertically.

39.3.7 RGB/YUV444 Image Scaling

For all RGB formats, the RGB pixels are converted up to RGB888 with 8 bits per each color component.

Then each color component is passed to the scaling engine and each component is treated in the same manor. The RGB scaling operation is the same as for the Y scaling operation described in the preceding sections. Also, YUV444 contains a byte for each color plane at each pixel location, so all three color components are scaled in the same manor.

39.3.8 Color Space Conversion (CSC)

There are two modules in the PXP to convert pixels between color spaces. They are referred to as CSC1 and CSC2 (for lack of a better naming convention).

CSC1 exists after the scaling unit and is dedicated to converting from YUV to RGB. CSC2 is a full duplex color space converter in that is can convert into either RGB or YUV (or YCbCr) color spaces depending on the desired output pixel format. All coefficients are programmed as two's compliment numbers and both CSC units can be bypassed if CSC is not desired at either position of these CSC units in the pixel data path.

39.3.9 CSC1 Operation

The CSC1 module receives scaled YUV/YCbCr444 pixels from the scale engine and converts the pixels to the RGB888 color space only if CSC1 is enabled.

The CSC1 module will convert only to the RGB color space and it can be bypassed to allow YUV pixels through the data path. These pixels are loaded into the pixel FIFO for processing by subsequent modules in the pixel data path.

The following equations are used to perform YUV/YCbCr -> RGB conversion. The constants will be stored in the PXP control registers as two's compliment values to allow flexibility in the implementation and to allow for differences in the video encode and decode operations. In addition, this provides a software mechanism to manipulate brightness or contrast.

R = C0(Y+Yoffset) + C1(V+UVoffset)

G = C0(Y+Yoffset) + C3(U+UVoffset) + C2(V+UVoffset)

B = C0(Y+Yoffset) + C4(U+UVoffset)

Note: In the equations above, U and V are synonymous with Cb and Cr in regards to the color space format of the source frame buffer.

Saturation of each color channel is checked and corrected for excursions outside the nominal YUV/YCbCr color spaces. Overflow for the three channels are saturated at 0x255 and underflow is saturated at 0x00.

The table below indicates the expected coefficients for YUV and YCbCr modes of operation:

Coefficient	YUV	YCbCr
Yoffset	0x000	0x1F0 (-16)
UVoffset	0x000	0x180 (-128)
C0	0x100 (1.00)	0x12A (1.164)
C1	0x123 (1.140)	0x198 (1.596)
C2	0x76B (-0.581)	0x730 (-0.813)
C3	0x79B (-0.394)	0x79C (-0.392)
C4	0x208 (2.032)	0x204 (2.017)

39.3.10 YUV versus YCbCr Support

By default, the PXP color space coefficients are set to support the conversion of YUV data to RGB data.

If YCbCr input is present, software must change the coefficient registers appropriately (see the register definitions for values). Software must also set the YCBCR_MODE bit in the COEFF0 register to ensure proper conversion of YUV versus YCBCR data.

39.3.11 CSC2 operation

The CSC2 module receives pixels in any color space and can convert the pixels into any of RGB, YUV, or YCbCr color spaces.

All coefficients are programmable and in the two's compliment notation. The output pixels are passed onto the LUT and rotation engine for further processing.

The following equations indicate the CSC2 modules ALU architecture.

Selecting RGB output in REG_CSC2_CTRL[CSC_MODE] configures the ALU in the following manor:

R = A1(Y-D1) + A2(U-D2) + A3(V-D3)

G = B1(Y-D1) + B2(U-D2) + B3(V-D3)

B = C1(Y-D1) + C2(U-D2) + C3(V-D3)

Selecting YUV output configures the ALU in the alternate manor:

Y = A1*R + A2*G + A3*B + D1

U = B1*R + B2*G + B3*B + D2

V = C1*R + C2*G + C3*B + D3

Saturation of each color channel is checked and corrected for excursions outside the nominal color space. Overflow for the three channels are saturated at 0x255 and underflow is saturated at 0x00.

39.3.12 Alpha Blending/Color Key

Regardless of pixel input format, the PS and AS pixels are normalized to 32-bits, organized as one alpha and three data bytes.

Alpha blending occurs in the RGB space, if blending is required, PS pixels should be converted to RGB space. If no alpha blending is required, then YUV pixels can bypass the alpha blending ALU without color space conversion.

All pixels are processed by the pixel ALU, but the ALU operations can be disabled to achieve pixel pass through for either PS or AS source pixels.

39.3.13 Alpha Blend

The alpha value for an individual pixel represents a mathematical weighting factor applied to the AS pixel.

An alpha value of 0x00 corresponds to a transparent pixel and a value of 0xFF corresponds to an opaque pixel.

The effective alpha value for an AS pixel is determined by the REG_AS_CTRL[ALPHA] and REG_AS_CTRL[ALPHA_CTRL] register fields. If REG_AS_CTRL[ALPHA_CTRL] = ALPHA_OVERRIDE , the alpha value for the pixel is taken from the REG_AS_CTRL[ALPHA] . This can be useful for applying a constant alpha to an entire image or for image formats that don't include an alpha value. If REG_AS_CTRL[ALPHA_CTRL] = ALPHA_MULTIPLY, the pixel's alpha value will be multiplied by the pixel's ALPHA value in order to allow scaling of the pixel's alpha or to provide better control for pixel formats such as RGB1555, which only contains a single bit of alpha.

For each color channel, the equation used to blend two source pixels is defined below:

Gá = PIO programmed global alpha (8-bit value).

Eá = Embedded alpha associated with AS pixel.

á = Gá * Eá + 0x80

The result for the red channel as an example:

R[7:0] = (á * PS.r) + ((1 - á) * AS.r)

When á is 0xff, the PS pixel will not be blended with the AS pixel, but PS will be passed as the output pixel and will not be blended with AS. In this case, AS will be discarded. Likewise, if á is 0x00 for a given pixel, PS will be loaded as the output pixel.

REG_AS_CTRL[ALPHA_INVERT] provides the option to invert the final alpha value. This essentially inverts the effect the alpha value has on the AS and PS blending operation.

39.3.14 Color Key

The color key function is provided to create transparent effects on the output pixel.

Color keying is applied on the input pixels after they are converted to 8-bits for each red, green, and blue color channels (color keys are not applied directly to 16-bit pixel formats but to their corresponding 24-bit representation). A color key range is programmable for both PS and AS pixels. If the PS 24-bit pixel is within the PS color key range, then AS is passed through the pixel pipeline. In this case, alpha blending does NOT occur. Conversely, if PS is within the AS color key range, then PS is passed via the PXP data pipeline. If both PS and AS color key tests pass, then the back ground color register is passed onto following PXP processing components in the pipeline.

The condition for color keying to be satisfied is:

CK0.r.low <= PS.r <= CK0.r.high

CK0.g.low <= PS.g <= CK0.g.high

CK0.b.low <= PS.b <= CK0.b.high

For example, if the "red" 8-bit value for the PS pixel (or PS.r) is between the color key low and high values (CK0.r.l and CK0.r.h), the condition is true for the red color plane. When ALL three color planes meet this condition, then only the PS pixel is loaded into the output register.

To disable color keying, program the low color key register value to 0xff and the high value to 0x00. This will guarantee that the color key range test will never be true.

39.3.15 LUT

The lookup table (LUT) is used to modify pixels in a manner that is not linear and that cannot be achieved by the color space conversion modules.

Nonlinear response to the input pixels can be achieved based on how the lookup table is programmed.

Programming of the direct access LUT table can be facilitated by single PIO register writes or DMA access. For efficient loading of the LUT, DMA access should be used.

39.3.16 Lookup Modes

The LUT has four lookup modes. The lookup modes determine how the src_pixel is used to address the LUT memory.

The four lookup modes are:

1. DIRECT_Y8
2. DIRECT_RGB444
3. DIRECT_RGB454
4. CACHE_RGB565

The DIRECT modes access the LUT memory as a monolithic SRAM. The CACHE_RGB565 will access the memory as a 2-way set associative cache.

39.3.17 DIRECT_Y8

DIRECT_Y8 is used for a 256-byte lookup. In DIRECT_Y8, the most significant byte of the pixel is used to address the LUT entry.

This byte reflects the Y/R channel of the pixel data path. Luma, or monochrome, transformations are possible with this lookup mode. The address is generated as: src_pixel[23:16]. In DIRECT_Y8 operation, the memory is byte addressable.

39.3.18 DIRECT_RGB444

DIRECT_RGB444 is used for a 8KB (4K pixel) RGB444 to RGB565 lookup.

Pixel formats that are in the YUV color space at the position of the LUT in the PXP data path can also be converted. To take advantage of the full 16KBmemory, the REG_LUT_CTRL[SEL_8KB] bit can be used to select the upper or lower 8KB memory, thus facilitating the use of 2 separate 444 LUT tables. In DIRECT_RGB444, the src_pixel is RGB/YUV[23;0] data is used to generate the lookup address. The address is generated as: {R/Y[23:20],G/U[15:12],B/V[7:4]}. In DIRECT_RGB444, the memory is pixel (2 byte) addressable.

39.3.19 DIRECT_RGB454

DIRECT_RGB454 is used for a 16KB (8K pixel) RGB454 to RGB565 lookup.

Pixel formats that are in the YUV color space at the position of the LUT in the PXP data path can also be converted. In DIRECT_RGB454, the src_pixel is RGB/YUV[23;0] data is used to generate the lookup address. The address is generated as: {R/Y[23:20],G/U[15:11],B/V[7:4]}. In DIRECT_RGB454, the memory is pixel (2 byte) addressable.

39.3.20 CACHE_RGB565

The CACHE_RGB565 lookup is used for a 128KB (65K pixel) RGB/YUV565 to RGB/YUV565 lookup.

The 128KB memory requirement is too costly from an area perspective to implement a complete lookup table in the LUT's on chip memory. For this reason, a LRU (least recently used) 16KB 2-way set associative cache has been implemented to reference the full 128KB lookup table stored in external memory.

The 2-way set associative cache is organized in the following way:

- 16KB total data storage
- 512 entries split between 256 ways
- 6 pixels/entry cache line

Cache efficiency is very critical. For a cache miss, the PXP will be stalled until the cache line can be filled. For a 32 bit DDR memory interface, the latency can be calculated as follows:

cycles latency for read command through DDR controller + CAS Latency + # burst cycles for read data to be returned + # cycles latency for data through DDR controller to LUT + 1 cycle for cache access of new data

The src_pixel is RGB[23:0] data used to generate the lookup address. To improve the cache efficiency the RGB/YUV565 address and the lookup table must be formatted in the following way:

Address: A[15:0] = $R_7G_7G_6B_7$ $R_6G_5B_6$ $R_5G_4B_5$ $R_4G_3B_4$ $R_3G_2B_3$

The address is organized as follows:

- A[15:12] tag address, used to compare a hit between cache ways
- A[11:4] index address, used to select cache set (i.e. row of memory)
- A[3:0] block address, used to select Pixel in the cache line

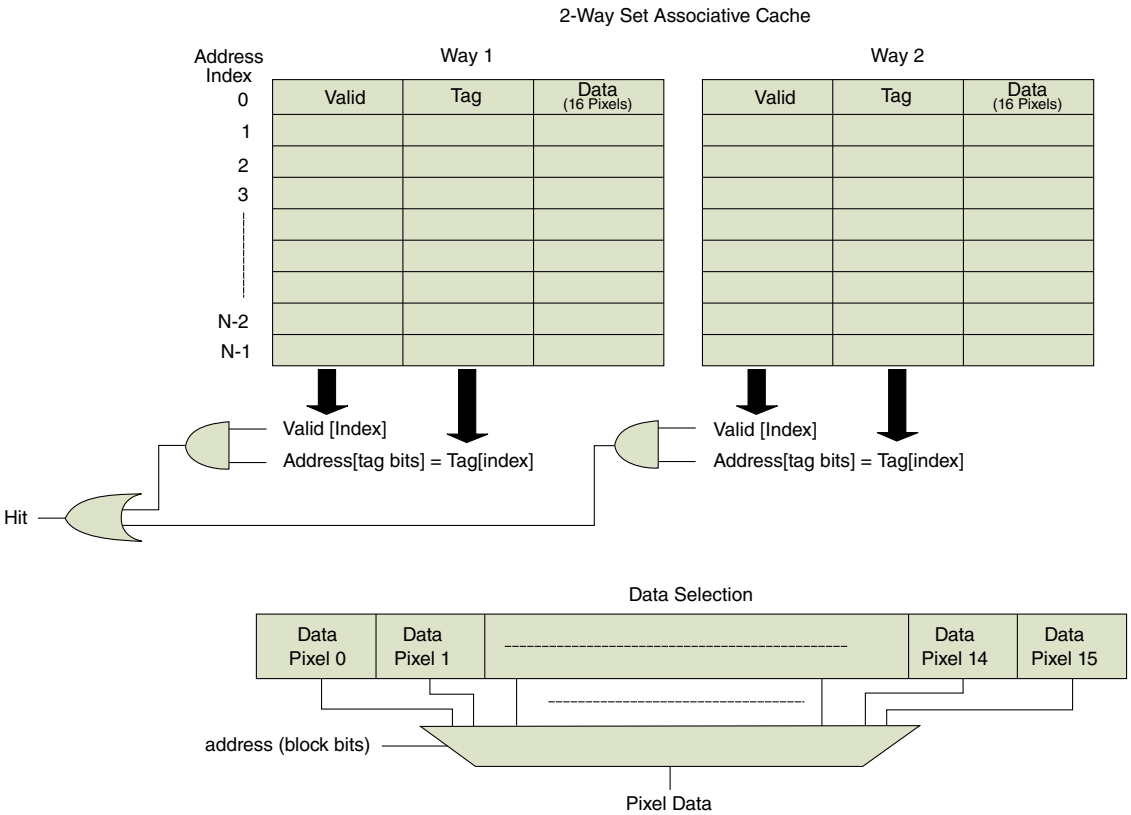


Figure 39-9. 2-Way Set Associative Cache and Data Selection

39.3.21 Output Modes

The LUT has three output modes for color space conversion.

1. Y8
2. RGBW4444CFA
3. RGB888

39.3.22 Y8

With out_mode set to Y8, in conjunction with DIRECT_Y8 lookup mode, the intended operation is Gama Correction.

Only the third byte is processed by the lookup table. The third byte is represented by the Y value or the R value in the data path since pixel data is either YUV[23:0] or RGB[23:0] where the Y or R byte encompass bits [23:16] respectfully. So, bits 15:0 are

always bypassed and left unchanged. Currently, the LUT is intended to process Y data when any YUV, Y8, or Y4 output pixel formats are selected. However, this resource can be enabled and used for any conceivable purpose.

Note: When the DIRECT_RGB444, DIRECT_RGB454 or CACHE_RGB565 lookup mode is selected in conjunction with the Y8 output mode the low order byte of the two bytes read from the LUT memory will be used as the Gama Correction value.

39.3.23 RGBW4444CFA

With REG_LUT_CTRL[OUT_MODE] set to RGBW4444CFA, the REG_CFA[DATA] is used to select one nibble from the LUT 16 bit output value.

The LUT memory lookup will contain a RGBW4444 value. The REG_CFA[DATA] will select the R,G,B or W nibble as the pixel value to present to the PXP data path based on the matrix defined by the REG_CFA[DATA] register. The 4 bit value is presented in the Y, or third byte lane, of the PXP data path. The final pixel transferred to the next PXP stage is {CFA[3:0],CFA[3:0],LUT[15:0]}

39.3.23.1 CFA Correction

The 32-bit REG__CFA[DATA] register is used to encode 16 CFA correction values.

The CFA correction values are encoded as follows:

- 00 selects R
- 01 selects G
- 10 selects B
- 11 selects W

The CFA correction uses 4x4 block processing. Figure 5; CFA mapping translation shows how CFA 4x4 blocks will iterate over the PXP's 8x8 or 16x16 pixel block being processed:

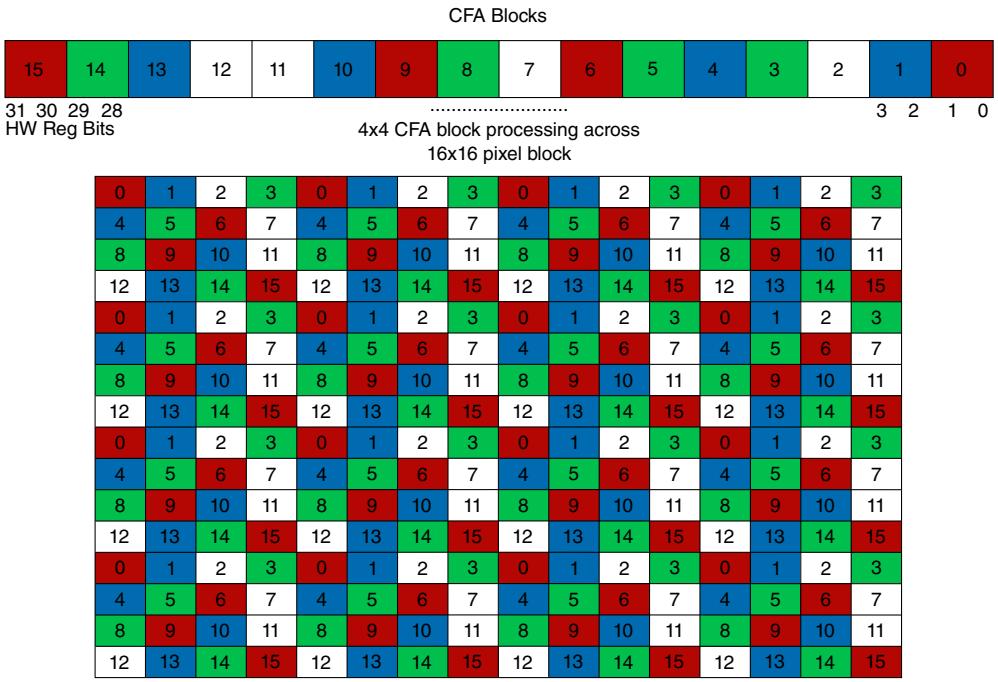


Figure 39-10. CFA mapping translation

39.3.24 RGB888

With out_mode set to RGB888, the memory output data is interpolated from RGB565 to RGB888.

The RGB[23:0] data is formatted as follows: R[7:3]R[7:5],G[7:2]G[7:6],B[7:3]B[7:5].

39.3.25 Rotation

There is a single rotation resource integrated into the PXP. The location of this resource within the PXP data path is programmable. Rotation can occur after compositing the AS and PS buffers in the output stage.

As an alternative configuration, the PS buffer can be rotated and later composited with the AS surface that is not rotated. There is a single configuration bit that provides the configuration of where rotation is implemented within the PXP.

To rotate graphics, the hardware must read pixels in one direction across a frame buffer and write them in a alternate orientation. For the 90 and 270 degree cases, this means that lines of pixels must either be read or written vertically in a frame buffer.

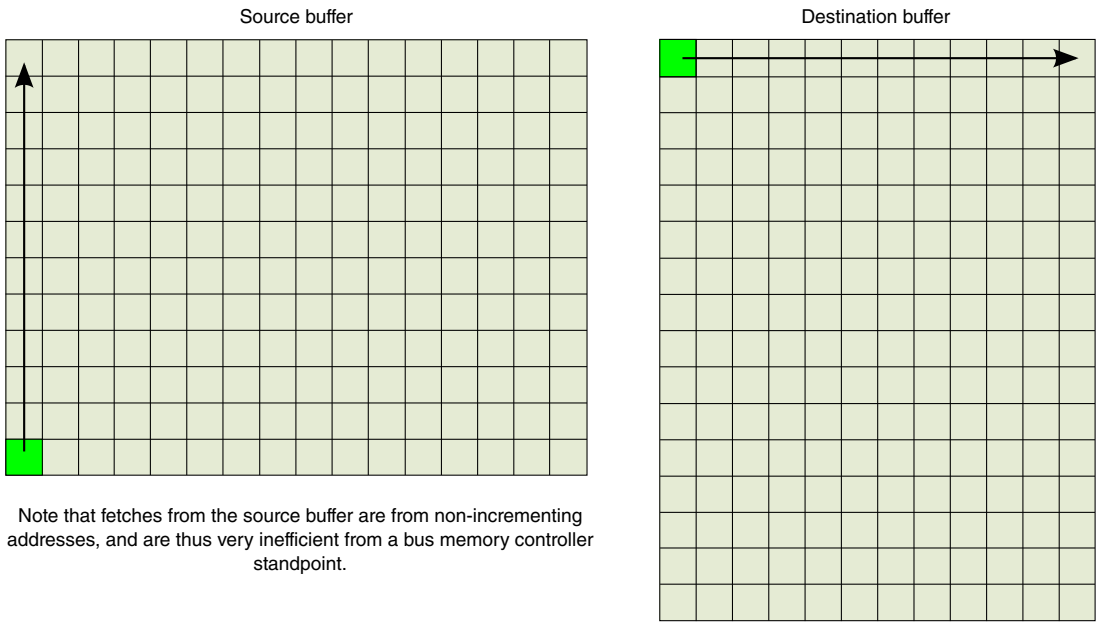


Figure 39-11. Rotation Read and Write

In order to rotate efficiently, multiple columns must be rotated to enable the engine to both fetch and store bursts of pixels, thus improving memory performance. The simplest method of doing this is to operate on square blocks of pixels. To rotate the image, each sub-block of pixels must be rotated by the required rotation angle.

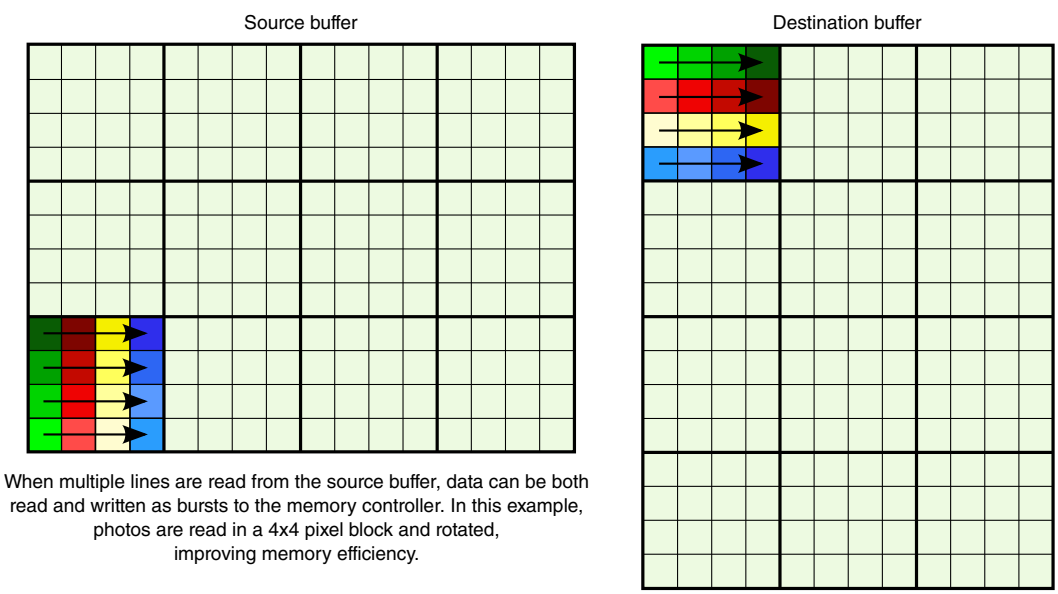


Figure 39-12. Rotated Sub-blocks

To manage the rotation process, the source image can be broken into a grid of sub-blocks that have coordinates as shown in the diagram below. In addition to rotating the sub-block, each block must be translated to a new coordinate location. For each of the rotation angles (0, 90, 180, 270), it is possible to define a simple algorithm for computing the new translated grid address. The hardware must then simply compute the memory address from the base grid address for both load and store operations.

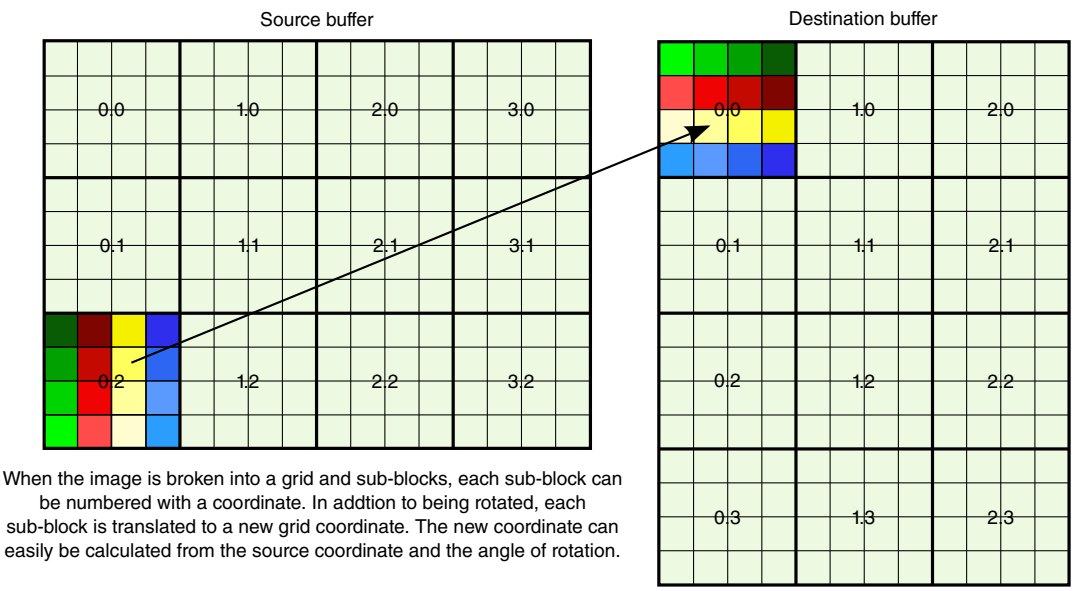


Figure 39-13. Grid of Sub-Blocks with Coordinates

In order to balance the requirements of reasonable burst sizes to the memory controller as well as keep the hardware storage requirements to a minimum, the blending/rotation engine will operate on either 8x8 or 16x16 pixel blocks. When using the Rotate engine with the input fetch engine, you need to program the input fetch engine to work in 8x8 block mode.

NOTE

An important artifact of the PXP is when rotating a source image and the output is NOT divisible by the block size selected. The output engine essentially truncates any output pixels after the desired number of pixels has been written. Since the output buffer is written as a horizontal row of blocks, the incorrect pixels could be truncated and the final output image can look shifted. In the case where the block size is programmed to 8x8, and the output size that is programmed is 12x12, then there is a remainder of 4 pixels that will be truncated in either the X and/or Y axis when the PXP operation

is complete. The output will be shifted by 4 pixels in this example. To compensate for this, the source base address needs to be adjusted so the correct pixels get truncated and the image does not look shifted. In this example, with 90 degrees of rotation, the PS base address should be adjusted by 4 times the actual PS base address -(4*pitch).

39.3.26 Output Buffer

The output buffer engine accepts data from the PXP pixel pipeline and issues requests to transfer the output pixels to external DRAM or the internal SRAM double buffer row of blocks.

39.3.27 Address calculator

Each of the blocks will manage its own fetch address using a common address calculator block that computes real addresses from a base address and relative block offset from the base.

Each block will then perform the multiple line fetches (or stores) required to perform the operation. This hides all the address buffer computations from the processing blocks and allows each block to simply track the coordinate of the block it is working on.

39.3.28 Block size selection

The PXP can be configured to process blocks that are either 8x8 pixels or 16x16 pixels with the REG_CTRL[BLOCK_SIZE] control bit.

When selecting a 16x16 pixel block size, the accesses to fetch AS and PS images and write the final frame buffer are more efficient since twice as much data is requested and processed per memory request.

When optimizing the system for memory bandwidth and image processing time, configure the PXP to process 16x16 pixel blocks.

39.3.29 Interlaced Video Support

The PXP has some minimal ability to generate interlaced video content from a progressive source. There two available options, based on the bandwidth requirements and how software is managing video frames.

The PXP can either interlace on the input side (by reading every other line of input data) or on the output side (by writing the individual lines of video into two separate fields). Generally, output interleaving should be used since it is the most flexible mode (it allows scaling and full overlay support) and it only requires a single pass of the PXP to generate two separate output fields.

Input interleaving can be beneficial in cases where the PXP is running at 60fps, since it requires fewer fetches to produce the output data. There is no direct hardware support for input interleaving, in that, there is no configuration bit that can be set to alter how the PXP processes a frame for input interleaving. Input interleaving is achieved by simply setting the source frame buffer pitch value to twice the value it would normally be set to for the equivalent progressive frame. The output parameters also need to be consistent with the desired processing effect. For example, the vertical resolution would be set to account for the reduced resolution to process the interlaced input buffers.

39.3.30 LCDIF Handshake

The PXP and LCDIF support a mode where the internal SRAM can be used for the frame buffer to minimize external memory bandwidth required.

This is accomplished by creating two buffers in SRAM, a double buffer row of blocks, where each correspond to 8/16-lines of the frame buffer. The buffers must be consecutive and allocated as a single block of data.

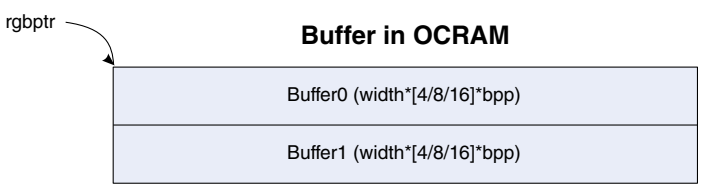


Figure 39-14. Buffer in OCRAM

The storage required can be calculated for an 8x8 block size as

- storage = 16 (lines) * rotated_row_length * pixel_size

and for 16x16 block size as

- storage = 32 (lines) * rotated_row_length * pixel_size

where pixel_size = 4 for 32bpp or 2 for 16bpp modes. The following table lists the storage requirements for common image sizes using 8x8 block size:

Image Size	Storage (16bpp)	Storage (24bpp)	Storage (32bpp)
320x240 (QVGA) - 0/180 rotation	10KB	15KB	20KB
320x240 (QVGA) - 90/270 rotation	7.5KB	11.5KB	15KB
640x480 (VGA) - 0/180 rotation	20KB	30KB	40KB
640x480 (VGA) - 90/270 rotation	15KB	22.5KB	30KB

The following diagram shows how the minimal rotation buffer would be organized. As the engine and LCD progress down the image, they continually swap roles of filling and emptying each eight-line buffer.

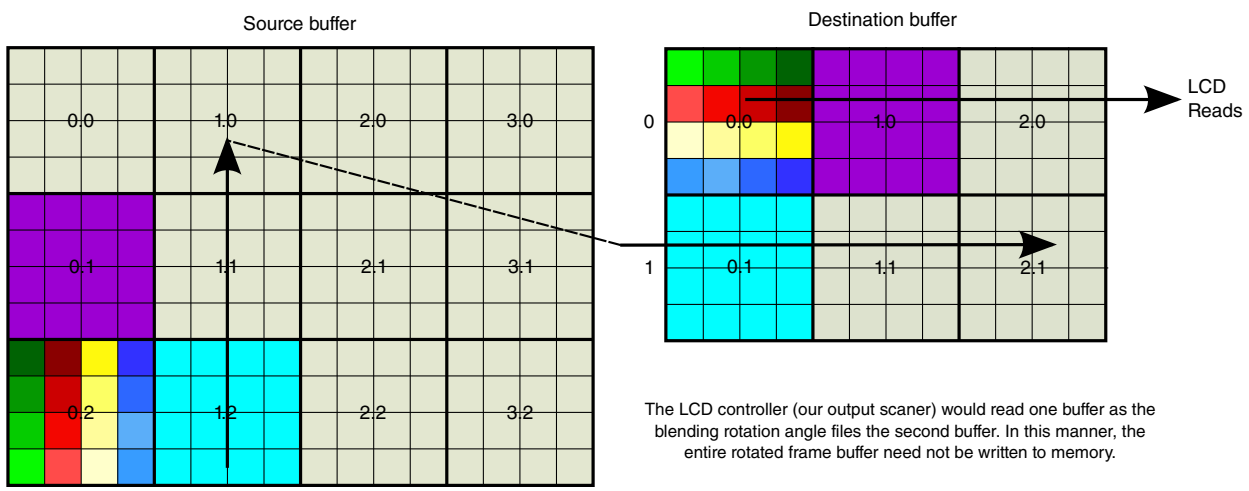


Figure 39-15. Minimal Rotation Buffer Organization

When this mode is enabled, the PXP will process one row of pixel blocks and write the results to the first SRAM buffer (buffer 0). The PXP will then alternate between writing subsequent rows to buffer 0 and buffer 1. After the PXP generates the data for one buffer, the LCDIF will begin reading that buffer and send the contents to the display device. Once the LCDIF finishes reading a buffer, it will start displaying from the other buffer while the PXP continues filling the previously processed buffer.

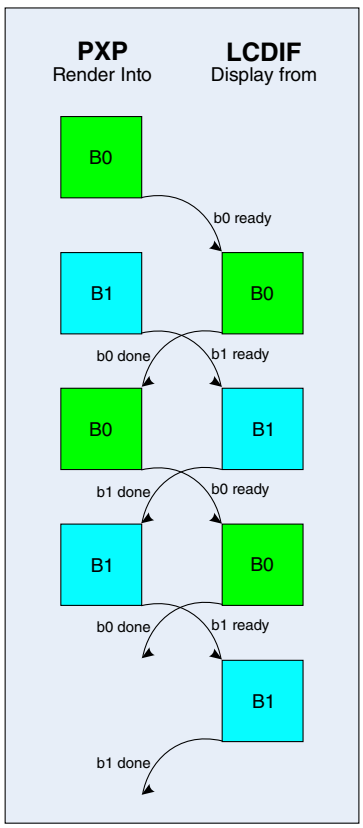


Figure 39-16. PXP and LCDIF Buffer Sharing

To accomplish the buffer sharing, the PXP and LCDIF will maintain buffer status using a pair of handshake signals. When a buffer is filled by the PXP, it will assert the pxp_lcdif_bx_ready (where x is 0 or 1) signal to indicate to the LCDIF that the buffer has valid data. The LCDIF will then release the buffer by asserting the lcdif_pxp_bx_done signal.

The basic protocol is shown in the diagram below:

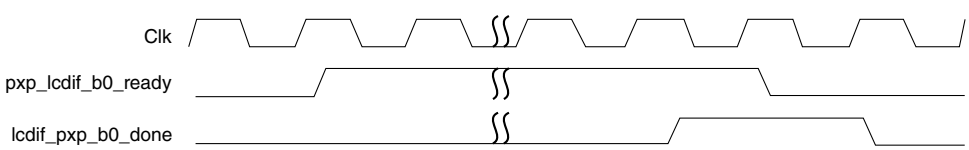


Figure 39-17. Buffer Sharing Protocol

The PXP will continue to assert the bx_ready signal until the corresponding bx_done signal is sampled high for one clock cycle. It will then deassert the bn_ready until the next time the buffer has been filled. After the PXP samples the bx_done signal asserted, it is free to begin filling the buffer with the next block size lines of display data. If a buffer has not been released when the PXP is ready to process data for that buffer, it will suspend rendering operations until the buffer has been released by the LCDIF.

39.3.31 LCDIF Abort

When the memory subsystem is not loaded, the PXP should be able to render the buffers faster than the LCDIF can drain the buffers.

It is possible under some scenarios (high LCDIF output rates with high memory latency) that the PXP may not be able to keep up with the LCDIF, even in the SRAM mode of operation. When this happens, the LCDIF will signal that it has completed one of the buffers before the other has been rendered by the PXP. This condition will be detected by the PXP's control logic as an "LCDIF Abort", which will cause the PXP to abort processing in the current row and proceed to the following row. It will acknowledge the abort to the LCDIF by raising the buffer_ready signal for the current buffer to enable the LCDIF to begin displaying the partially-filled buffer. While an abort will create artifacts in the video display, it does minimize the artifacts by limiting them to the remaining pixels blocks in the current row versus ruining the entire frame buffer.

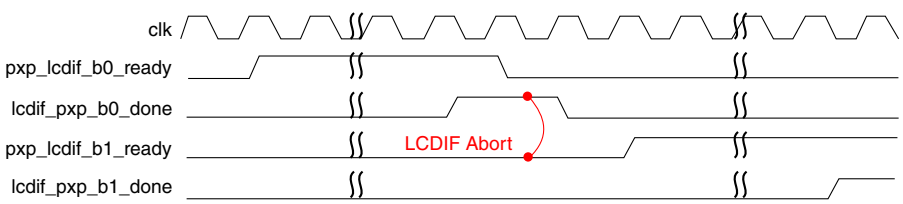


Figure 39-18. LCDIF Abort

39.3.32 Theory of Operation

The PXP can be used to accelerate graphics operations by offloading graphics processing from the processor. The block can perform alpha blending and color key substitution on two RGB graphics buffers.

The PXP is organized as having a processed surface (PS) and an alpha surface (AS) that can be blended with the processed surface. There are no restrictions on the location of the AS or PS within the output surface (OS). As the PXP processes NxN blocks, operations

are performed on a pixel by pixel basis. The AS and PS pixels are alpha blended, color keyed, process by CSC and LUT resources as individual pixel components. This allows efficient block processing with supporting arbitrary alignment for both the AS and PS surfaces. The resulting pixel block is then written to the corresponding block in the output buffer.

39.3.33 Pixel Handling

All pixels are internally represented as 24-bit values regardless of input or output pixel formats.

The pixels get converted in the AS and PS buffer engines to 24-bit pixel values.. There is also an 8-bit alpha value at stages up to the alpha blender within the PXP for blending within the RGB color space. Compositing of AS and PS images can only occur in the RGB color space. If composting is not required, then YUV pixels can be transferred and processed at all PXP pixel resource components.. The color orientation of pixels within the PXP can be controlled by the CSC1, CSC2, and LUT resources.

For RGB, input pixels are converted into 24-bit pixel values using the following rules:

1. 32-bit ARGB8888 pixels are read directly with no conversion for both the AS and PS.
2. 32-bit RGB888 pixels are assumed to have an alpha value of 0xFF (full opaque).
3. 6-bit RGB565 and RGB555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of 0xFF (opaque). The expansion process replicates the upper pixel bits into the lower pixel bits (for instance a 16-bit RGB555 triplet of 0x1F/0x10/0x07 would be expanded to 0xFF/0x84/0x39).
4. 16-bit RGB1555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of either 0x00 or 0xFF, based on the 1-bit alpha value in the pixel. The ALPHA_MULTIPLY function is useful in this scenario to allow scaling of the opaque pixels to a semi-transparent value.

Alpha values can be passed through the entire PXP data path and output in ARGB888 and ARGB555 pixel modes. Also, output pixels can be assigned an alpha value using the REG_OUT_CTRL[ALPHA] register. 16-bit pixels values are formed from the most significant bits of the 24-bit pixel values.

When YUV/YCbCr output formats are selected, all pixels are internally represented as either RGB or YUV pixels values. The CSC2 or LUT can convert internal RGB/YUV pixels into the correct output format. In this way, any PS color space can be blended with AS RGB pixels and output in any color space.

39.3.34 Output Buffer Composition

The output buffer will be rendered by composing each pixel block from the associated PS and AS buffers.

The AS pixel buffer can be blended or color-keyed with the associated data from the PS buffer (either the PS image pixels or REG_PS_BACKGROUND register based on PS programmed coordinates).

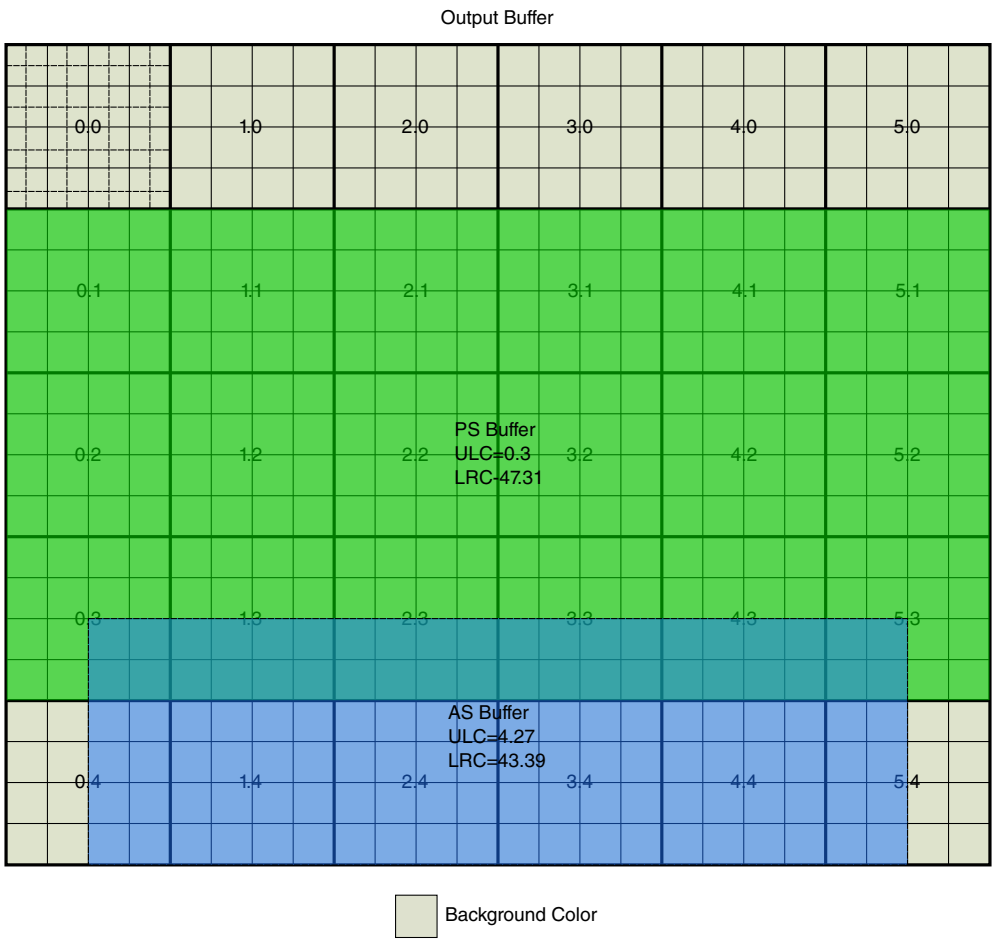


Figure 39-19. Output Buffer Composition

39.3.35 PS Image Processing

As the PXP processes image buffers, it iterates over the output buffer by fetching the corresponding input buffer blocks and processing the pixels embedded in these.

39.3.36 Letterboxing

At each pixel coordinate, the control logic determines if the PS pixel (argument also applies to AS pixels) will be used in rendering the output pixel.

This is determined by checking the output pixel's coordinates against the REG_OUT_PS_ULC and REG_OUT_PS_LRC (ULC and LRC in short) register contents. For pixels outside this region, the PS pixel will be loaded with the pixel value from REG_PS_BACKGROUND, which can be used to effectively control the letterboxing color. There are no block size or block boundary restrictions when setting the ULC or LRC for either the AS or PS. The only restriction is that the ULC and LRC are within the OUT LRC extents.

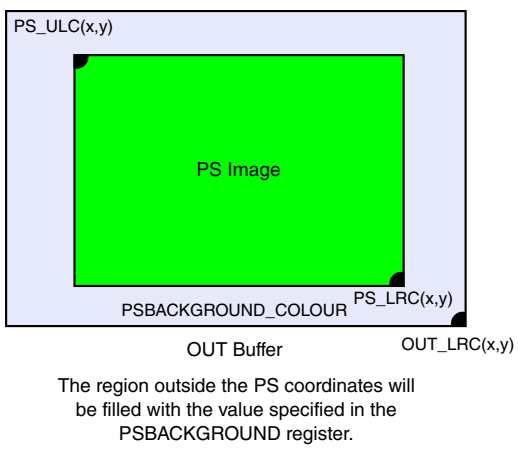


Figure 39-20. OUT Buffer

39.3.37 Clipping source images

A subset of the PS buffer can be used in rendering the output buffer. The PXP_PS_BUF register can indicate an offset into the PS buffer that will be used for display within the OUTPUT buffer.

The pixel at the address defined in the PXP_PS_BUF register will be the pixel that is displayed at the pixel coordinate indicated by PXP_OUT_PS_ULC within the output buffer. Essentially, the PXP_PS_BUF register can be used to establish an offset into the PS buffer thus clipping all PS buffer pixels that are at a lower address. The PXP_PS_PITCH will always indicate the number of bytes that are vertically adjacent in

the PS buffer. The settings in the PXP_PS_BUF, PXP_OUT_PS_ULC, and PXP_OUT_PS_LRC will determine the subset of the PS buffer, or clipped PS source buffer, that will be used in the output buffer.

It is important to note that when scaling the PS buffer, the coordinates of the PS buffer within the output buffer need to be consistent with the scaling factors and original PS buffer size.

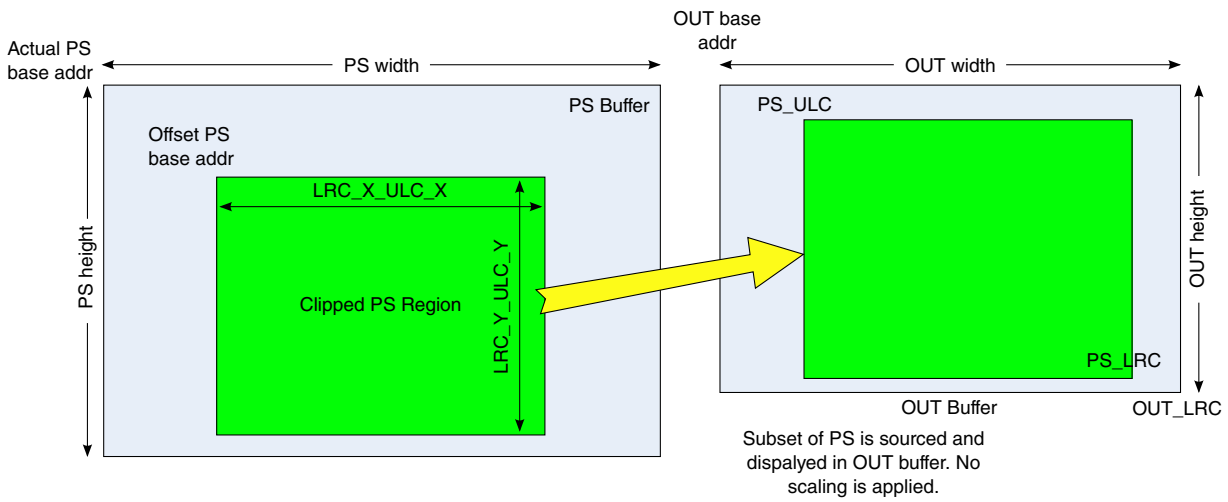


Figure 39-21. PS Buffer Scaling

When sourcing a subset of the PS image, it should fall completely within the PS buffer to avoid displaying incorrect data. The following conditions should be met:

x_base_addr_offset + x_scale * (LRC_X-ULC_X) <= PS_pitch

y_base_addr_offset + y_scale * (LRC_Y-ULC_Y) <= PS_size

The PXP hardware does not check for these conditions and will render the image as programmed. The following case could indicate invalid programming parameters for the PXP:

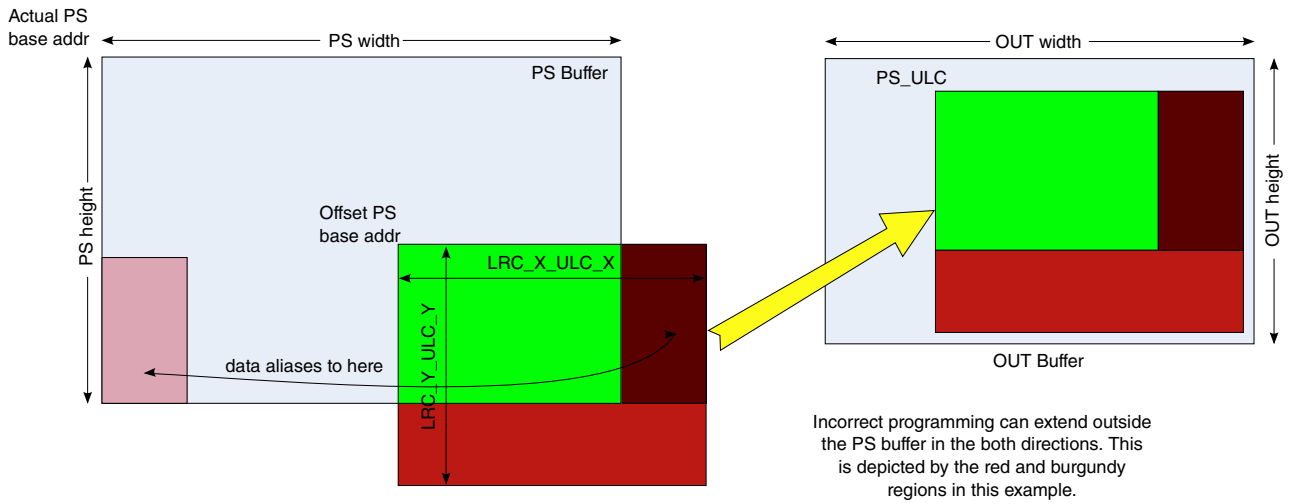


Figure 39-22. Example with Invalid Parameters

39.3.38 Color Key Processing

Pixels may be made transparent to the corresponding AS by using the PS color key registers.

If a PS pixel matches the range specified by the REG_PS_COLORKEYLOW and REG_PS_COLORKEYHIGH registers, the pixel from the associated AS will be displayed. If no AS is present for the pixel, a black pixel will be generated since the default AS pixel is 0x00000000 (transparent black pixel).

The most common use for this is when a bitmap does not support an alpha-field or for applications such as "green screen" where an image is substituted for a solid background color .



Figure 39-23. The PS image (player) and AS image (stadium)

The green portion of the background image can be color keyed to display the contents of the AS buffer for locations that match the color range. For this example, the color range is:

PS Colorkey: 00<R<80 70<G<ff 00<B<80

The resulting image becomes:



Figure 39-24. Resulting Image

39.3.39 In Place Processing (PS buffer is destination buffer)

The PXP also has the ability to process an image and write the resulting buffer back to the original PS buffer. This is referred to as "in place" rendering.

This could be useful for basic blit operations into the PS buffer. IN_PLACE operations are achieved by programming the OUT base address to the pixel location in the PS buffer that marks the upper left pixel of the update region. The actual region that is updated should be indicated by programming the ULC = (0,0) and the LRC = (X,Y). The region bounded by the coordinates will be updated, and the rest of the PS buffer will not be modified.

39.3.40 Alpha Surface (AS) Processing

The AS surface has a complete set of registers that determines how the AS effects the final OUT surface.

Most of the registers that exist for the PS surface also are defined for the AS surface where applicable. This is provided to replicate the SW interface for each PS and AS processes.

39.3.41 Alpha Handling

Alpha values in the AS are embedded in the source image pixels. For AS pixel formats that do not support an alpha value, the pixel is assigned an alpha value of 0xFF (opaque).

This can be modified by the AS control by setting either the ALPHA_MULTIPLY or ALPHA_OVERRIDE bit in the associated AS_CTRL register. If ALPHA_MULTIPLY is enabled, the 8-bit ALPHA value from the AS_CTRL register is multiplied by the source alpha before blending with the PS image. If the ALPHA_OVERRIDE bit is set, the 8-bit ALPHA value is simply substituted for the pixel.

39.3.42 Color Key Processing (AS_CTRL)

The AS_CTRL register also contains an ENABLE_COLORKEY bit that can be used to enable or disable color key substitution for the AS.

When enabled, the pixel values are compared to the ASCOLORKEYLOW and ASCOLORKEYHIGH registers to determine if a match has occurred. When an AS pixel matches the color key range, the pixel from the AS image is considered transparent and the corresponding PS pixel is rendered. If both the PS and AS pixels match their corresponding color key ranges, the AS pixel is displayed unmodified.

AS color keys are handled in a manner similar to PS color keys. The same images used in the PS color key example could be used with the images swapped. In this case, matches on the AS image to the ASCOLORKEY register would display the PS pixels.

39.4 Output Image Processing

Several PXP options affect the resulting output image.

39.4.1 Output Image Size

The PXP generates an output image in the resolution programmed by the REG_OUT_LRC. As the PXP processes pixels, it iterates over the NxN blocks (in output scan-block order) based on the final image resolution.

39.4.2 Output Format

The result of PXP operations are written to the buffer pointed to by the REG_OUT_BUF/ REG_OUT_BUF2 registers. The pixel format is controlled by the REG_OUT_CTRL[FORMAT] bit-field.

32-bit pixels are formed directly from the internal 24-bit representations and 16-bit pixel formats are generated by truncating the internal 24-bit values to the appropriate number of bits. For formats supporting an alpha value, the PXP assigns the alpha using the 8-bit value in the REG_OUT_CTRL[ALPHA] field. For ARGB1555, the most significant alpha bit is appended to the output pixel. Also, for ARGB4444, the most significant nibble is appended to the output pixel. Single and dual buffer YUV output formats are also available. Since each pixel in the data path is represented by a full YUV444 24bpp value, decimation reduces the output in cases of YUV422/420 output formats.

39.4.3 Rotation/Flip operations

The PXP supports four rotation angles in conjunction with vertical and horizontal flip options. The flip operations effectively take place before the rotation.

Rotations of 0, 90, 180, and 270 degrees are supported and any combination of rotation and flip are supported. There is no performance difference between any of these modes of operation.

39.5 Queuing PXP transactions

The PXP supports a primitive ability to queue up one operation while the current operation is running. This is enabled through the use of the REG_NEXT register.

When this register is written, it enables the PXP to reload its current register contents with the data found at the location pointed to by this address when it completes processing of the current frame. This feature may be useful in helping to reduce the interrupt latency in servicing the PXP, especially in cases where the PXP and LCDIF are using the on-chip SRAM buffer handshake (since the PXP must begin generating next frame data immediately).

If the PXP is idle when the REG_NEXT register is written, the PXP treats this as an indication that it should immediately load the values at the pointer and begin processing the frame. This ability should allow software to use the same routines when programming the PXP (so that the first frame doesn't differ from subsequent frames).

When loading values from the NEXT register, all registers in the PXP are reloaded. Some register loads have no effect

After writing the REG_NEXT register, the PXP will set the REG_NEXT[ENABLED] bit of the REG_NEXT register to indicate that the next command has been queued. Software should first check the status of this bit to ensure that a previous command has not been enabled. Likewise, after programming the first frame in a sequence of frames, software should poll this bit until it is sampled logic 1'b0 before queuing the next operation.

The PXP will issue interrupts from frames as they complete, regardless of whether they were started by writing the control registers directly or using the REG_NEXT register. When software receives an interrupt, it should check/clear the PXP's status register as normal, poll the REG_PXP[ENABLED] bit, and then issue the next operation. A queued operation may be cancelled by issuing a CLEAR operation to the REG_PXP[ENABLED] register bit. The SET and TOGGLE operations should never be used with this register.

39.6 Error Handling

The PXP does minimal checking on the control registers, so it is important that these are correctly specified. The PXP does monitor the bus transactions for errors and will report errors in the status register.

Upon receipt of a bus error, the PXP will set the ERROR interrupt and abort any further operations. Bus errors can be generated from any system access that results in an error response returned from the internal SIM Bus errors in the PXP are signaled as either a read or a write error, but do not indicate the failing address. Software may deduce the failing address from the current block status indicators.

39.6.1 Known PXP Limitations/Issues

The PXP has the following known limitations:

1. When using the NEXT register, the interrupt enable setting should remain the same for all frames. If not, the PXP will change the interrupt enable register value and possible cause the loss of an interrupt.
2. Rotations of 180/270 are not supported when performing LCD handshakes

39.7 PXP Memory Map/Register Definition

PXP Hardware Register Format Summary

PXP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_C000	Control Register 0 (PXP_CTRL)	32	R/W	C000_0000h	39.7.1/2352
21C_C010	Status Register (PXP_STAT)	32	R/W	0000_0000h	39.7.2/2354
21C_C020	Output Buffer Control Register (PXP_OUT_CTRL)	32	R/W	0000_0000h	39.7.3/2356
21C_C030	Output Frame Buffer Pointer (PXP_OUT_BUF)	32	R/W	0000_0000h	39.7.4/2358
21C_C040	Output Frame Buffer Pointer #2 (PXP_OUT_BUF2)	32	R/W	0000_0000h	39.7.5/2359
21C_C050	Output Buffer Pitch (PXP_OUT_PITCH)	32	R/W	0000_0000h	39.7.6/2359
21C_C060	Output Surface Lower Right Coordinate (PXP_OUT_LRC)	32	R/W	0000_0000h	39.7.7/2360
21C_C070	Processed Surface Upper Left Coordinate (PXP_OUT_PS_ULC)	32	R/W	0000_0000h	39.7.8/2361

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_C080	Processed Surface Lower Right Coordinate (PXP_OUT_PS_LRC)	32	R/W	0000_0000h	39.7.9/2362
21C_C090	Alpha Surface Upper Left Coordinate (PXP_OUT_AS_ULC)	32	R/W	0000_0000h	39.7.10/ 2363
21C_C0A0	Alpha Surface Lower Right Coordinate (PXP_OUT_AS_LRC)	32	R/W	0000_0000h	39.7.11/ 2364
21C_C0B0	Processed Surface (PS) Control Register (PXP_PS_CTRL)	32	R/W	0000_0000h	39.7.12/ 2365
21C_C0C0	PS Input Buffer Address (PXP_PS_BUF)	32	R/W	0000_0000h	39.7.13/ 2366
21C_C0D0	PS U/Cb or 2 Plane UV Input Buffer Address (PXP_PS_UBUF)	32	R/W	0000_0000h	39.7.14/ 2367
21C_C0E0	PS V/Cr Input Buffer Address (PXP_PS_VBUF)	32	R/W	0000_0000h	39.7.15/ 2368
21C_C0F0	Processed Surface Pitch (PXP_PS_PITCH)	32	R/W	0000_0000h	39.7.16/ 2368
21C_C100	PS Background Color (PXP_PS_BACKGROUND)	32	R/W	0000_0000h	39.7.17/ 2369
21C_C110	PS Scale Factor Register (PXP_PS_SCALE)	32	R/W	1000_1000h	39.7.18/ 2370
21C_C120	PS Scale Offset Register (PXP_PS_OFFSET)	32	R/W	0000_0000h	39.7.19/ 2371
21C_C130	PS Color Key Low (PXP_PS_CLRKEYLOW)	32	R/W	00FF_FFFFh	39.7.20/ 2372
21C_C140	PS Color Key High (PXP_PS_CLRKEYHIGH)	32	R/W	0000_0000h	39.7.21/ 2373
21C_C150	Alpha Surface Control (PXP_AS_CTRL)	32	R/W	0000_0000h	39.7.22/ 2373
21C_C160	Alpha Surface Buffer Pointer (PXP_AS_BUF)	32	R/W	0000_0000h	39.7.23/ 2376
21C_C170	Alpha Surface Pitch (PXP_AS_PITCH)	32	R/W	0000_0000h	39.7.24/ 2376
21C_C180	Overlay Color Key Low (PXP_AS_CLRKEYLOW)	32	R/W	00FF_FFFFh	39.7.25/ 2377
21C_C190	Overlay Color Key High (PXP_AS_CLRKEYHIGH)	32	R/W	0000_0000h	39.7.26/ 2377
21C_C1A0	Color Space Conversion Coefficient Register 0 (PXP_CSC1_COEF0)	32	R/W	0400_0000h	39.7.27/ 2378
21C_C1B0	Color Space Conversion Coefficient Register 1 (PXP_CSC1_COEF1)	32	R/W	0123_0208h	39.7.28/ 2380
21C_C1C0	Color Space Conversion Coefficient Register 2 (PXP_CSC1_COEF2)	32	R/W	079B_076Ch	39.7.29/ 2380
21C_C1D0	Color Space Conversion Control Register. (PXP_CSC2_CTRL)	32	R/W	0000_0001h	39.7.30/ 2381

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C_C1E0	Color Space Conversion Coefficient Register 0 (PXP_CSC2_COEF0)	32	R/W	0000_0000h	39.7.31/ 2383
21C_C1F0	Color Space Conversion Coefficient Register 1 (PXP_CSC2_COEF1)	32	R/W	0000_0000h	39.7.32/ 2383
21C_C200	Color Space Conversion Coefficient Register 2 (PXP_CSC2_COEF2)	32	R/W	0000_0000h	39.7.33/ 2384
21C_C210	Color Space Conversion Coefficient Register 3 (PXP_CSC2_COEF3)	32	R/W	0000_0000h	39.7.34/ 2384
21C_C220	Color Space Conversion Coefficient Register 4 (PXP_CSC2_COEF4)	32	R/W	0000_0000h	39.7.35/ 2385
21C_C230	Color Space Conversion Coefficient Register 5 (PXP_CSC2_COEF5)	32	R/W	0000_0000h	39.7.36/ 2386
21C_C240	Lookup Table Control Register. (PXP_LUT_CTRL)	32	R/W	8001_0000h	39.7.37/ 2386
21C_C250	Lookup Table Control Register. (PXP_LUT_ADDR)	32	R/W	0000_0000h	39.7.38/ 2388
21C_C260	Lookup Table Data Register. (PXP_LUT_DATA)	32	R/W	0000_0000h	39.7.39/ 2390
21C_C270	Lookup Table External Memory Address Register. (PXP_LUT_EXTMEM)	32	R/W	0000_0000h	39.7.40/ 2390
21C_C280	Color Filter Array Register. (PXP_CFA)	32	R/W	0000_0000h	39.7.41/ 2391
21C_C290	Histogram Control Register. (PXP_HIST_CTRL)	32	R/W	0000_0020h	39.7.42/ 2391
21C_C2A0	2-level Histogram Parameter Register. (PXP_HIST2_PARAM)	32	R/W	0000_0F00h	39.7.43/ 2392
21C_C2B0	4-level Histogram Parameter Register. (PXP_HIST4_PARAM)	32	R/W	0F0A_0500h	39.7.44/ 2393
21C_C2C0	8-level Histogram Parameter 0 Register. (PXP_HIST8_PARAM0)	32	R/W	0604_4000h	39.7.45/ 2394
21C_C2D0	8-level Histogram Parameter 1 Register. (PXP_HIST8_PARAM1)	32	R/W	0F0D_0B09h	39.7.46/ 2395
21C_C2E0	16-level Histogram Parameter 0 Register. (PXP_HIST16_PARAM0)	32	R/W	0302_0100h	39.7.47/ 2396
21C_C2F0	16-level Histogram Parameter 1 Register. (PXP_HIST16_PARAM1)	32	R/W	0706_0504h	39.7.48/ 2397
21C_C300	16-level Histogram Parameter 2 Register. (PXP_HIST16_PARAM2)	32	R/W	0B0A_0908h	39.7.49/ 2398
21C_C310	16-level Histogram Parameter 3 Register. (PXP_HIST16_PARAM3)	32	R/W	0F0E_0D0Ch	39.7.50/ 2399
21C_C320	PXP Power Control Register. (PXP_POWER)	32	R/W	0000_0000h	39.7.51/ 2399
21C_C400	Next Frame Pointer (PXP_NEXT)	32	R/W	0000_0000h	39.7.52/ 2400

39.7.1 Control Register 0 (PXP_CTRL)

The CTRL register contains controls for the PXP module.

PXP_CTRL: 0x000

PXP_CTRL_SET: 0x004

PXP_CTRL_CLR: 0x008

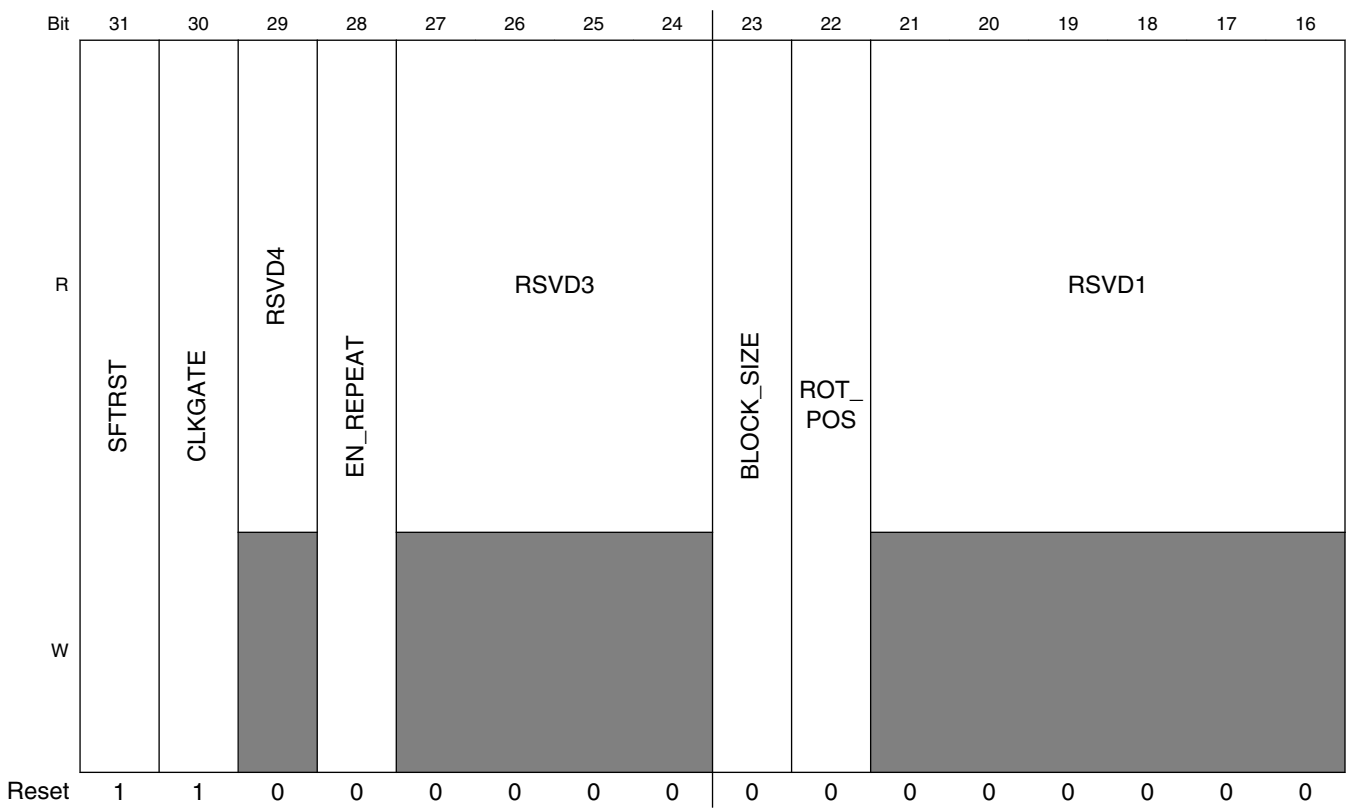
PXP_CTRL_TOG: 0x00C

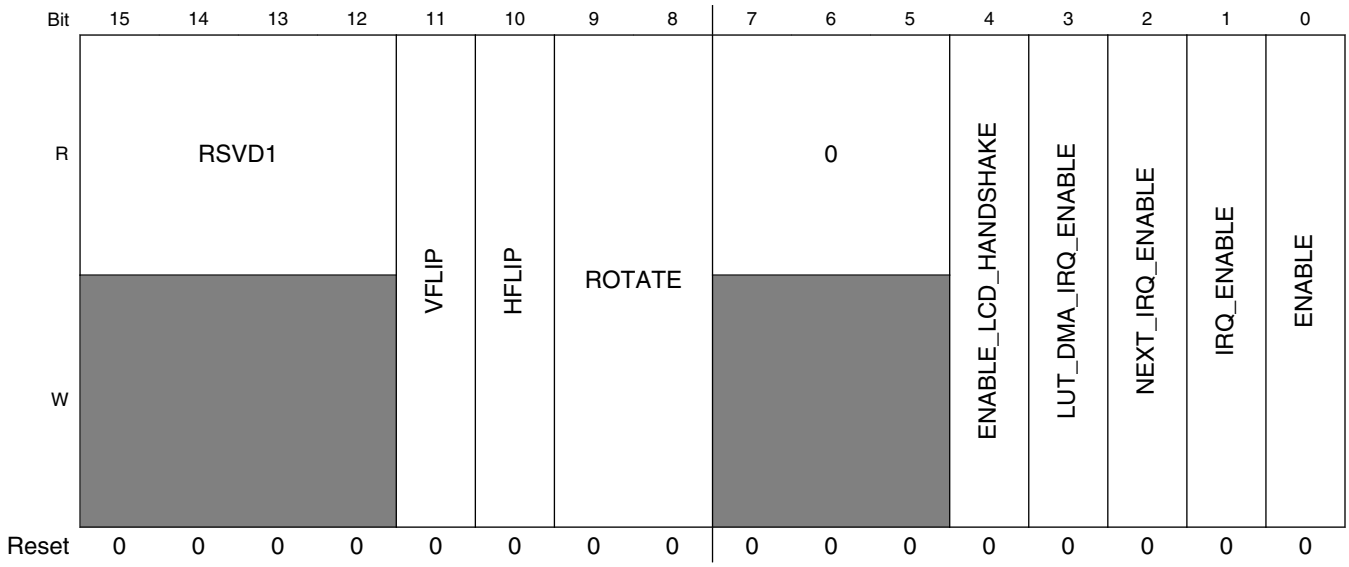
The Control register contains the primary controls for the PXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

EXAMPLE

```
PXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
PXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 21C_C000h base + 0h offset = 21C_C000h





PXP_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal PXP operation. Set this bit to one (default) to disable clocking with the PXP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the PXP block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 RSVD4	Reserved, always set to zero.
28 EN_REPEAT	Enable the PXP to run continuously. When this bit is set, the PXP will repeat based on the current configuration register settings. If this bit is not set, the PXP will complete the process and enter the idle state ready to accept the next frame to be processed. This bit should be set when the LCDIF handshake mode is enabled so that the next frame is automatically generated for the next screen refresh cycle. If it not set and the handshake mode is enabled, the CPU will have to initiate the PXP for the next refresh cycle. When the PXP NEXT feature is used, it has priority over the REPEAT mode, in that the new register settings are fetched first, and then the next PXP operation will continue.
27–24 RSVD3	Reserved, always set to zero.
23 BLOCK_SIZE	Select the block size to process. 0x0 **8X8** — Process 8x8 pixel blocks. 0x1 **16X16** — Process 16x16 pixel blocks.
22 ROT_POS	This bit controls where rotation will occur in the PXP datapath. Setting this bit to 1'b0 will place the rotation resources at the output stage of the PXP data path. Image compositing will occur before pixels are processed for rotation. Setting this bit to a 1'b1 will place the rotation resources before image composition. Only the PS can be rotated in this configuration and AS will not be rotated.
21–12 RSVD1	Reserved, always set to zero.
11 VFLIP	Indicates that the output buffer should be flipped vertically (effect applied before rotation).
10 HFLIP	Indicates that the output buffer should be flipped horizontally (effect applied before rotation).

Table continues on the next page...

PXP_CTRL field descriptions (continued)

Field	Description
9–8 ROTATE	Indicates the clockwise rotation to be applied at the output buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation. 0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 ENABLE_LCD_ HANDSHAKE	Enable handshake with LCD controller. When this is set, the PXP will not process an entire framebuffer, but will instead process rows of NxN blocks in a double-buffer handshake with the LCDIF. This enables the use of the onboard SRAM for a partial frame buffer.
3 LUT_DMA_IRQ_ ENABLE	LUT DMA interrupt enable. When set, the PXP will issue an interrupt when the LUT DMA has finished transferring data.
2 NEXT_IRQ_ ENABLE	Next command interrupt enable. When set, the PXP will issue an interrupt when a queued command initiated by a write to the PXP_NEXT register has been loaded into the PXP's registers. This interrupt also indicates that a new command may now be queued.
1 IRQ_ENABLE	Interrupt enable. NOTE: When using the PXP_NEXT functionality to reprogram the PXP, the new value of this bit will be used and may therefore enable or disable an interrupt unintentionally.
0 ENABLE	Enables PXP operation with specified parameters. The ENABLE bit will remain set while the PXP is active and will be cleared once the current operation completes. Software should use the IRQ bit in the PXP_STAT when polling for PXP completion.

39.7.2 Status Register (PXP_STAT)

The PXP Interrupt Status register provides interrupt status information.

PXP_STAT: 0x010

PXP_STAT_SET: 0x014

PXP_STAT_CLR: 0x018

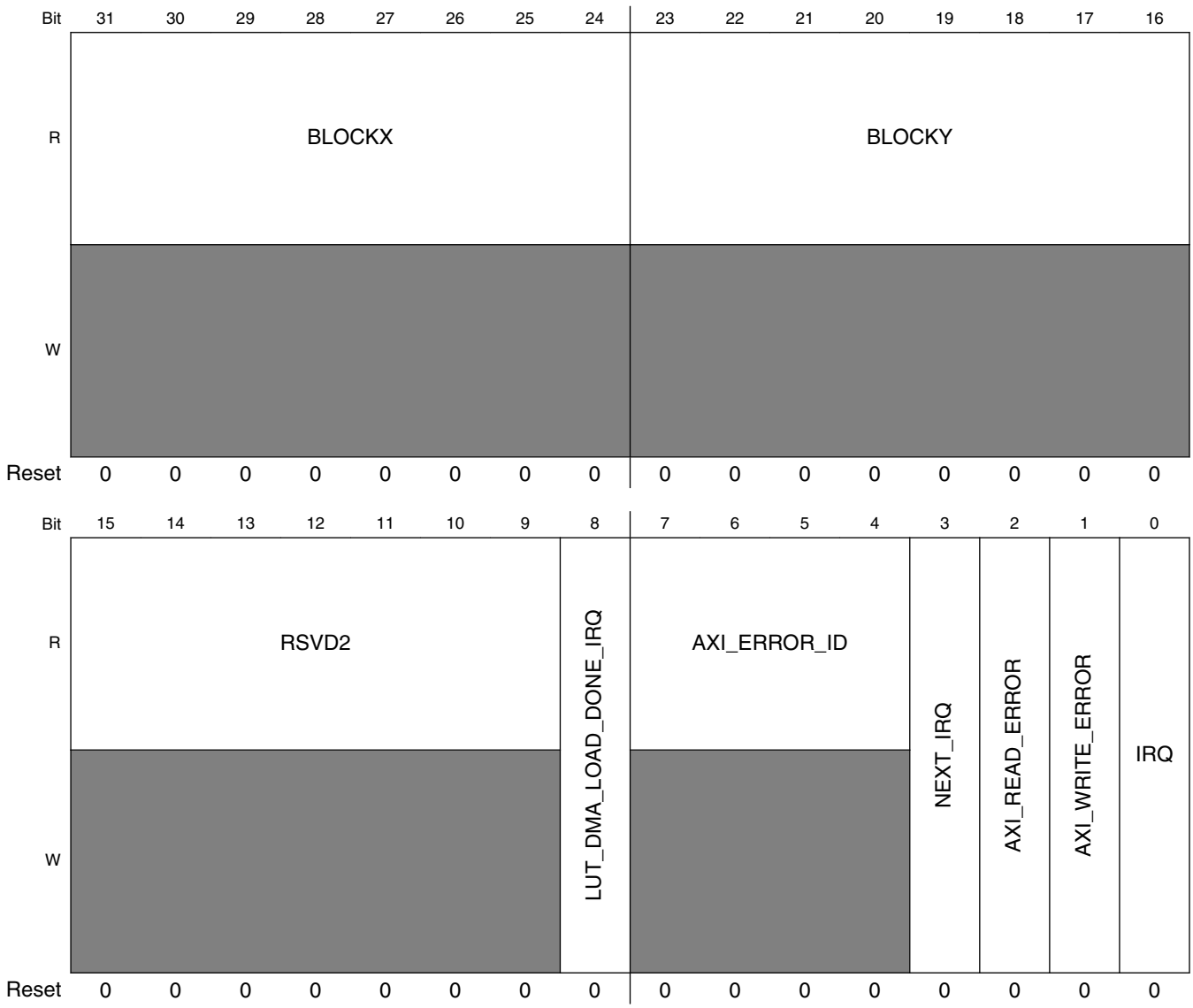
PXP_STAT_TOG: 0x01C

This register provides PXP interrupt status and the current X/Y block coordinate that is being processed.

EXAMPLE

```
PXP_STAT_CLR(BM_PXP_STAT_IRQ);  // clear CSC interrupt
```

Address: 21C_C000h base + 10h offset = 21C_C010h



PXP_STAT field descriptions

Field	Description
31–24 BLOCKX	Indicates the X coordinate of the block currently being rendered.
23–16 BLOCKY	Indicates the X coordinate of the block currently being rendered.
15–9 RSVD2	Reserved, always set to zero.
8 LUT_DMA_ LOAD_DONE_ IRQ	Indicates that the LUT DMA transfer has completed.
7–4 AXI_ERROR_ID	Indicates the AXI ID of the failing bus operation.

Table continues on the next page...

PXP_STAT field descriptions (continued)

Field	Description
3 NEXT_IRQ	Indicates that a command issued with the "Next Command" functionality has been issued and that a new command may be initiated with a write to the PXP_NEXT register.
2 AXI_READ_ ERROR	Indicates PXP encountered an AXI read error and processing has been terminated.
1 AXI_WRITE_ ERROR	Indicates PXP encountered an AXI write error and processing has been terminated.
0 IRQ	Indicates current PXP interrupt status. The IRQ is routed through the pxp_irq when the IRQ_ENABLE bit in the control register is set.

39.7.3 Output Buffer Control Register (PXP_OUT_CTRL)

The OUT_CTRL register contains controls for the Output Buffer.

PXP_OUT_CTRL: 0x020

PXP_OUT_CTRL_SET: 0x024

PXP_OUT_CTRL_CLR: 0x028

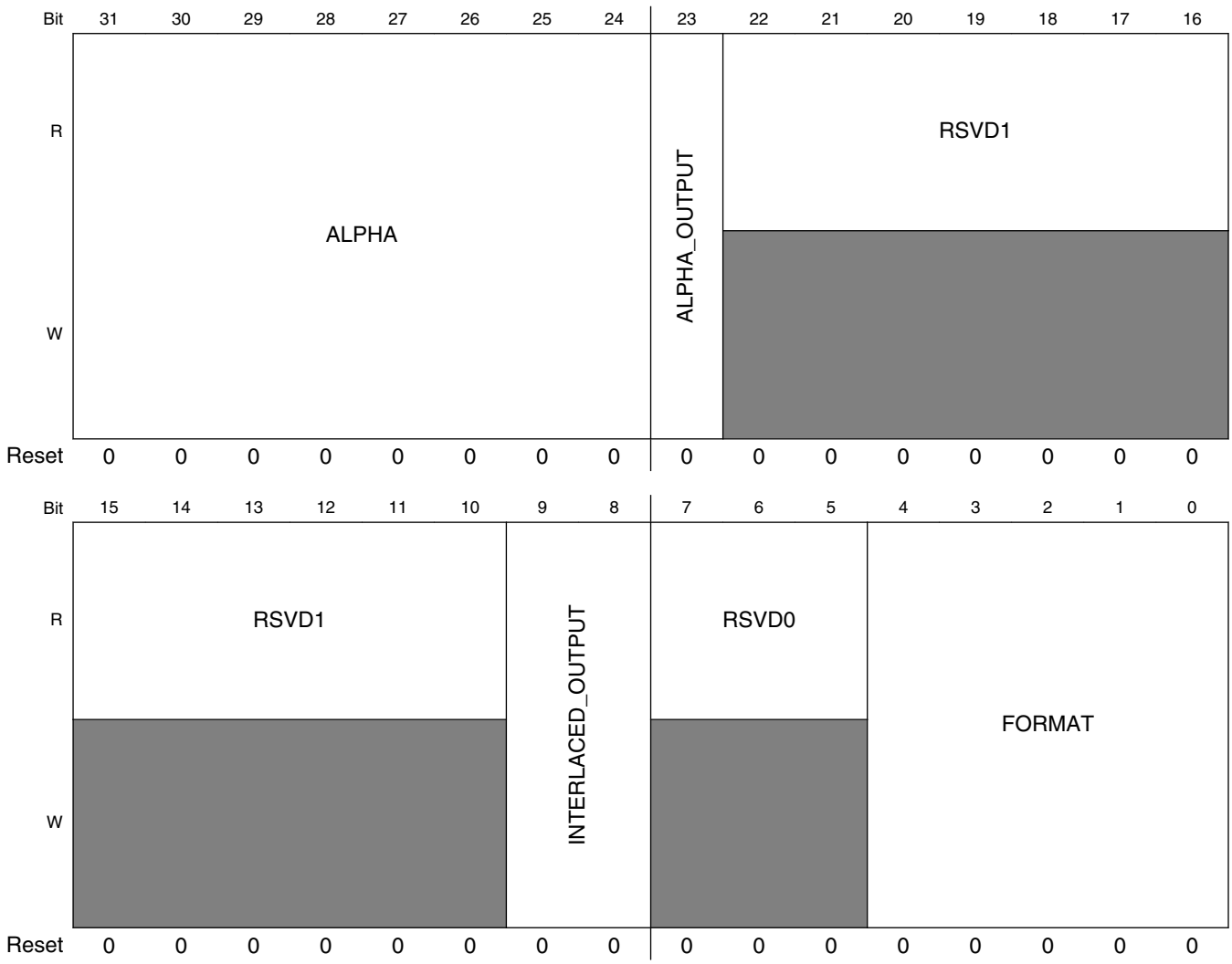
PXP_OUT_CTRL_TOG: 0x02C

The Control register contains the primary controls for the PXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

EXAMPLE

```
PXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
PXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 21C_C000h base + 20h offset = 21C_C020h



PXP_OUT_CTRL field descriptions

Field	Description
31–24 ALPHA	When generating an output buffer with an alpha component, the value in this field will be used when enabled to override the alpha passed through the pixel data pipeline.
23 ALPHA_ OUTPUT	Indicates that alpha component in output buffer pixels should be overwritten by PXP_OUT_CTRL[ALPHA]. If 0, retain their alpha value from the computed alpha for that pixel.
22–10 RSVD1	Reserved, always set to zero.
9–8 INTERLACED_ OUTPUT	Determines how the PXP writes it's output data. Output interlacing should not be used in conjunction with input interlacing. Splitting frames into fields is most efficient using output interlacing. 2-plane output formats AND interlaced output is NOT supported. 0x0 **PROGRESSIVE** — All data written in progressive format to the OUTBUF Pointer. 0x1 **FIELD0** — Interlaced output: only data for field 0 is written to the OUTBUF Pointer.

Table continues on the next page...

PXP_OUT_CTRL field descriptions (continued)

Field	Description
	0x2 **FIELD1** — Interlaced output: only data for field 1 is written to the OUTBUF2 Pointer. 0x3 **INTERLACED** — Interlaced output: data for field 0 is written to OUTBUF and data for field 1 is written to OUTBUF2.
7–5 RSVD0	Reserved, always set to zero.
FORMAT	Output framebuffer format. The UV byte lanes are synonymous with CbCr byte lanes for YUV output pixel formats. For example, the YUV2P420 format should be selected when the output is YCbCr 2-plane 420 output format. 0x0 **ARGB8888** — 32-bit pixels 0x4 **RGB888** — 32-bit pixels (unpacked 24-bit pixel in 32 bit DWORD.) 0x5 **RGB888P** — 24-bit pixels (packed 24-bit format) 0x8 **ARGB1555** — 16-bit pixels 0x9 **ARGB4444** — 16-bit pixels 0xC **RGB555** — 16-bit pixels 0xD **RGB444** — 16-bit pixels 0xE **RGB565** — 16-bit pixels 0x10 **YUV1P444** — 32-bit pixels (1-plane XYUV unpacked) 0x12 **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0x13 **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0x14 **Y8** — 8-bit monochrome pixels (1-plane Y luma output) 0x15 **Y4** — 4-bit monochrome pixels (1-plane Y luma, 4 bit truncation) 0x18 **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0x19 **YUV2P420** — 16-bit pixels (2-plane UV) 0x1A **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0x1B **YVU2P420** — 16-bit pixels (2-plane VU)

39.7.4 Output Frame Buffer Pointer (PXP_OUT_BUF)

Output Framebuffer Pointer. This register points to the beginning of the output frame buffer. This pointer is used for progressive format and field 0 when generating interlaced output.

This register is used by the logic to point to the current output location for the output frame buffer.

EXAMPLE

```
PXP_OUT_BUF_WR( buffer );
```

Address: 21C_C000h base + 30h offset = 21C_C030h

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

PXP_OUT_BUF field descriptions

Field	Description
ADDR	Current address pointer for the output frame buffer. The address can have any byte alignment. 64B alignment is recommended for optimal performance.

39.7.5 Output Frame Buffer Pointer #2 (PXP_OUT_BUF2)

Output Framebuffer Pointer #2. This register points to the beginning of the output frame buffer for either field 1 when generating interlaced output or for the UV buffer when in YUV 2-plane output modes. Both interlaced output AND 2-plane output modes are not supported in a single PXP operation. This register is not used as the pointer to the second buffer when in LCDIF_HANDSHAKE mode.

This register is used by the logic to point to the current output location for the field 1 or UV output frame buffer.

EXAMPLE

```
PXP_OUT_BUF_WR( field0 );  // buffer for interlaced field 0
PXP_OUT_BUF2_WR( field1 ); // buffer for interlaced field 1
```

Address: 21C_C000h base + 40h offset = 21C_C040h

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

PXP_OUT_BUF2 field descriptions

Field	Description
ADDR	Current address pointer for the output frame buffer. The address can have any byte alignment. 64B alignment is recommended for optimal performance.

39.7.6 Output Buffer Pitch (PXP_OUT_PITCH)

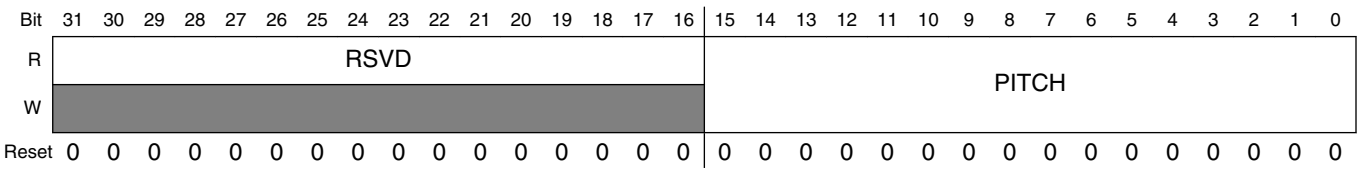
This register contains the output buffer pitch in bytes.

Any byte value will indicate the vertical pitch. This value will be used in output pixel address calculations.

EXAMPLE

```
PXP_OUT_PITCH_WR( 68 * 4 );  // The output buffer pitch is 68 pixels times 32 bits per pixel
```

Address: 21C_C000h base + 50h offset = 21C_C050h



PXP_OUT_PITCH field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

39.7.7 Output Surface Lower Right Coordinate (PXP_OUT_LRC)

This register contains the size, or lower right coordinate, of the output buffer NOT rotated. It is implied that the upper left coordinate of the output surface is always [0,0]. When rotating the framebuffer, the PXP will automatically swap the X/Y, or WIDTH/HEIGHT, to accomodate the rotated size.

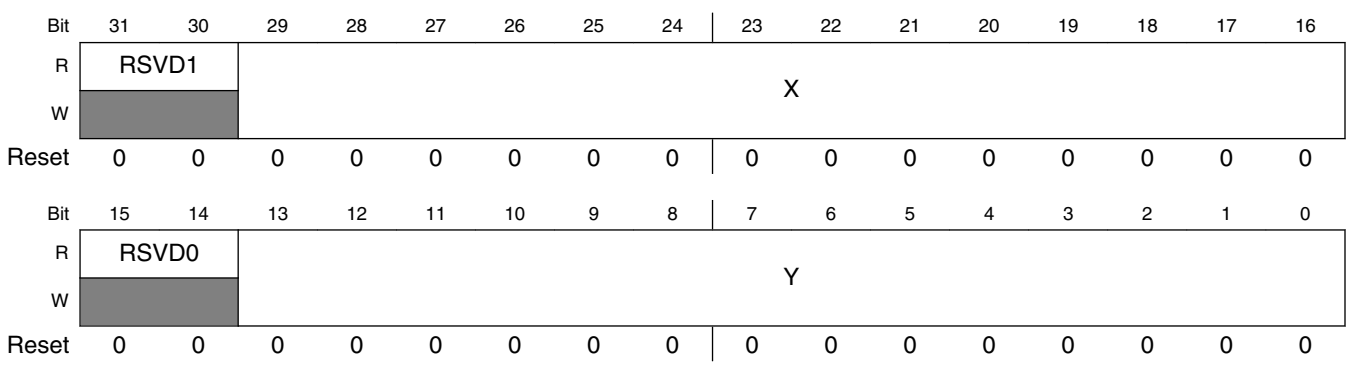
This register sets the size of the output frame buffer in pixels, not blocks. The frame buffer need not be a multiple of NxN pixels. Partial blocks will be written for output frame buffer sizes that are not divisable by N pixels in either dimension.

EXAMPLE

```
PXP_OUT_LRC[X]=319;   // set width of output frame buffer to 320 pixels
PXP_OUT_LRC[Y]=243;  // set height of output frame buffer to 244 pixels which is not
divisible by block size N

PXP_OUT_LRC_WR( BF_PXP_OUT_LRC_X(319)  |  BF_PXP_OUT_LRC_Y(243) );
```

Address: 21C_C000h base + 60h offset = 21C_C060h



PXP_OUT_LRC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	Indicates number of horizontal PIXELS in the output surface (non-rotated). The output buffer pixel width minus 1 should be programmed. The image size is not required to be a multiple of 8 pixels. The PXP will clip the pixel output at this boundary.
15–14 RSVD0	Reserved, always set to zero.
Y	Indicates the number of vertical PIXELS in the output surface (non-rotated). The output buffer pixel height minus 1 should be programmed. The image size is not required to be a multiple of 8 pixels. The PXP will clip the pixel output at this boundary.

39.7.8 Processed Surface Upper Left Coordinate (PXP_OUT_PS_ULC)

This register contains the upper left pixel coordinate for the Processed Surface in the OUTPUT buffer.

This register contains the upper left coordinate of the Processed Surface in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. If the value of the PXP_OUT_PS_ULC is greater than the PXP_OUT_LRC, then no PS pixels will be fetched from memory, but only PXP_PS_BACKGROUND pixels will be processed by the PS engine. Pixel locations that are greater than or equal to the PS upper left coordinates, less than or equal to the PS lower right coordinates, and within the PXP_OUT_LRC extents will use the PS to render pixels into the output buffer.

EXAMPLE

```
PXP_OUT_PS_ULC_WR(0,0x0002_0002); // Processed Surface upper left coordinate at (X,Y) =
2,2.  The PS surface will not effect pixels in the first and second row and column of the
output buffer.
```

Address: 21C_C000h base + 70h offset = 21C_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1		X													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0		Y													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_PS_ULC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	This field indicates the upper left X-coordinate (in pixels) of the processed surface (PS) in the output buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the upper left Y-coordinate (in pixels) of the processed surface in the output buffer.

39.7.9 Processed Surface Lower Right Coordinate (PXP_OUT_PS_LRC)

This register contains the lower right extent for the Processed Surface in the OUTPUT buffer.

This register contains the lower right coordinate of the Processed Surface in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are greater than or equal to the PS upper left coordinates, less than or equal to the PS lower right coordinates, and within the PXP_OUT_LRC extents will use the PS to render pixels into the output buffer.

EXAMPLE

```
PXP_OUT_PS_ULC_WR(0,0x03FF_03FF); // With this UL/LR pair of pixel coordinates, only one
pixel at OUT[X,Y]=1023,1023 will use the PS to contribute to its value.
PXP_OUT_PS_LRC_WR(0,0x03FF_03FF);
```

Address: 21C_C000h base + 80h offset = 21C_C080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1								X							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								Y							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_PS_LRC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	This field indicates the lower right X-coordinate (in pixels) of the processed surface (PS) in the output frame buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the lower right Y-coordinate (in pixels) of the processed surface in the output frame buffer.

39.7.10 Alpha Surface Upper Left Coordinate (PXP_OUT_AS_ULC)

This register contains the upper left location for the Alpha Surface in the output buffer.

This register contains the upper left coordinate of AS in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are greater than or equal to the upper left coordinates will use the AS to render pixels in the output buffer.

EXAMPLE

```
PXP_OUT_AS_ULC_WR(0,0x0001_0001); // Alpha Surface upper left coordinate at (X,Y) = 1,1.
The AS surface will not effect pixels in the first row or first column of the output buffer.
```

Address: 21C_C000h base + 90h offset = 21C_C090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1		X													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0		Y													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_AS_ULC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_OUT_AS_ULC field descriptions (continued)

Field	Description
29–16 X	This field indicates the upper left X-coordinate (in pixels) of the alpha surface (AS) in the output frame buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the upper left Y-coordinate (in pixels) of the alpha surface in the output frame buffer.

39.7.11 Alpha Surface Lower Right Coordinate (PXP_OUT_AS_LRC)

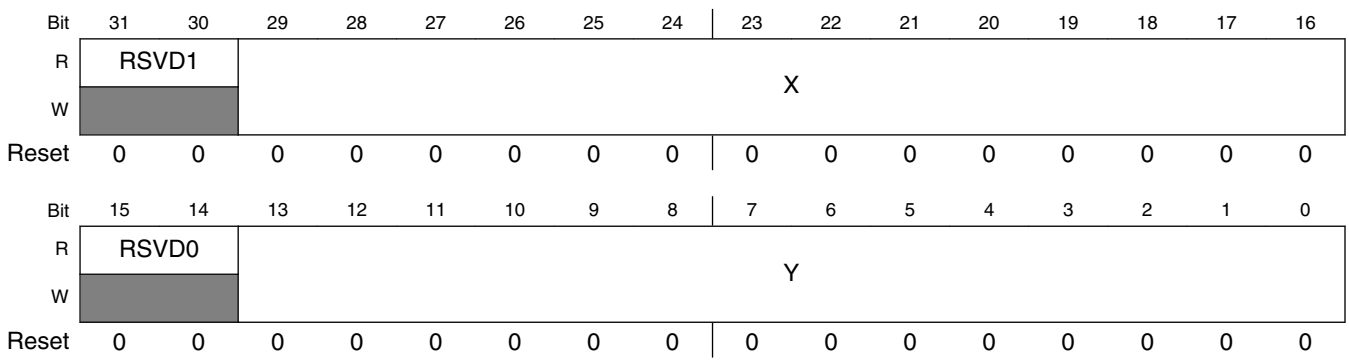
This register contains the lower right extent for Alpha Surface in the output buffer.

This register contains the lower right coordinate of AS in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are less than or equal to the lower right coordinates will use the AS to render pixels in the output buffer.

EXAMPLE

```
PXP_AS_LRC_WR(0,0x03FF_03FF); // Alpha Surface lower right coordinate at (X,Y) = 1023,1023.
```

Address: 21C_C000h base + A0h offset = 21C_C0A0h



PXP_OUT_AS_LRC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	This field indicates the lower right X-coordinate (in pixels) of the alpha surface (AS) in the output frame buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the lower right Y-coordinate (in pixels) of the alpha surface in the output frame buffer.

39.7.12 Processed Surface (PS) Control Register (PXP_PS_CTRL)

The PS_CTRL register contains controls for the Processed Surface Buffer.

PXP_PS_CTRL: 0x0B0

PXP_PS_CTRL_SET: 0x0b4

PXP_PS_CTRL_CLR: 0x0B8

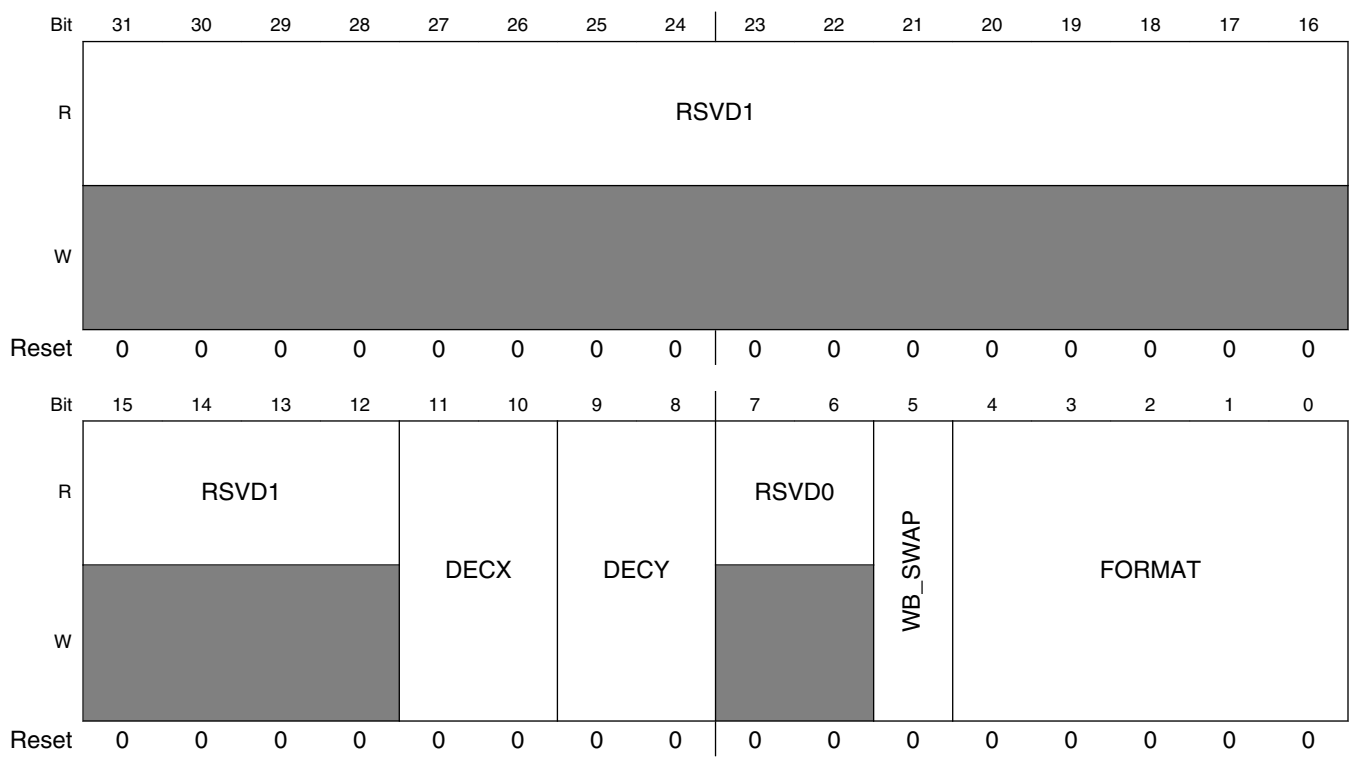
PXP_PS_CTRL_TOG: 0x0BC

The Control register contains the primary controls for the PXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

EXAMPLE

```
PXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
PXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 21C_C000h base + B0h offset = 21C_C0B0h



PXP_PS_CTRL field descriptions

Field	Description
31–12 RSVD1	Reserved, always set to zero.
11–10 DECX	Horizontal pre decimation filter control. 0x0 **DISABLE** — Disable pre-decimation filter. 0x1 **DECX2** — Decimate PS by 2. 0x2 **DECX4** — Decimate PS by 4. 0x3 **DECX8** — Decimate PS by 8.
9–8 DECY	Verticle pre decimation filter control. 0x0 **DISABLE** — Disable pre-decimation filter. 0x1 **DECY2** — Decimate PS by 2. 0x2 **DECY4** — Decimate PS by 4. 0x3 **DECY8** — Decimate PS by 8.
7–6 RSVD0	Reserved, always set to zero.
5 WB_SWAP	Swap bytes in words. For each 16 bit word, the two bytes will be swapped.
FORMAT	PS buffer format. To select between YUV and YCbCr formats, see bit 31 of the CSC1_COEF0 register. 0x4 **RGB888** — 32-bit pixels (unpacked 24-bit format) 0xC **RGB555** — 16-bit pixels 0xD **RGB444** — 16-bit pixels 0xE **RGB565** — 16-bit pixels 0x10 **YUV1P444** — 32-bit pixels (1-plane XYUV unpacked) 0x12 **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0x13 **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0x14 **Y8** — 8-bit monochrome pixels (1-plane Y luma output) 0x15 **Y4** — 4-bit monochrome pixels (1-plane Y luma, 4 bit truncation) 0x18 **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0x19 **YUV2P420** — 16-bit pixels (2-plane UV) 0x1A **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0x1B **YVU2P420** — 16-bit pixels (2-plane VU) 0x1E **YUV422** — 16-bit pixels (3-plane format) 0x1F **YUV420** — 16-bit pixels (3-plane format)

39.7.13 PS Input Buffer Address (PXP_PS_BUF)

PS Input Buffer Address. This should be programmed to the starting address of the RGB data or Y (luma) data for the PS plane.

This register contains the pointer to the Luma/RGB buffer. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
PXP_PS_BUF_WR(image_rgb); // RGB image
PXP_PS_BUF_WR(image_y);   // Y (luma) image data
PXP_PS_UBUF_WR(image_u);  // U (Cb) image data
PXP_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 21C_C000h base + C0h offset = 21C_C0C0h

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

PXP_PS_BUF field descriptions

Field	Description
ADDR	Address pointer for the PS RGB or Y (luma) input buffer.

39.7.14 PS U/Cb or 2 Plane UV Input Buffer Address (PXP_PS_UBUF)

PS Chroma (U/Cb/UV) Input Buffer Address. This register points to the beginning of the PS U/Cb input buffer. In two plane operation, this register points to the beginning of the PS UV chroma input buffer.

This register contains the pointer to the Chroma U/Cb or 2 plane UV buffer. This register is unused when processing 1-plane buffer formats. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
PXP_PS_BUF_WR(image_y);   // Y (luma) image data
PXP_PS_UBUF_WR(image_u);  // U (Cb) image data
PXP_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 21C_C000h base + D0h offset = 21C_C0D0h

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

PXP_PS_UBUF field descriptions

Field	Description
ADDR	Address pointer for the PS U/Cb or 2 plane UV Chroma input buffer.

39.7.15 PS V/Cr Input Buffer Address (PXP_PS_VBUF)

PS Chroma (V/Cr) Input Buffer Address. This register points to the beginning of the PS V/Cr input buffer. In one or two plane operation, this register is not used. In monochrome modes Y8 and Y4, the low 16 bits are used as the U/V data in the datapath instead of sourcing U/V data from external buffers. In this case, it represents a fixed value for U/V data.

This register contains the pointer to the Chroma V/Cr buffer. For Y8/Y4 modes, the low 16 bits are used as the monochrome U and V values in the data path. Bits [15:8] represent the U data byte, and bits]7:0] represent the V data byte. Other than with Y8/Y4 input buffer formats, this register is unused when processing 1 or 2-plane buffer formats. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
PXP_PS_BUF_WR(image_y);   // Y (luma) image data
PXP_PS_UBUF_WR(image_u);  // U (Cb) image data
PXP_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 21C_C000h base + E0h offset = 21C_C0E0h

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

PXP_PS_VBUF field descriptions

Field	Description
ADDR	Address pointer for the PS V/Cr Chroma input buffer.

39.7.16 Processed Surface Pitch (PXP_PS_PITCH)

This register contains the processed surface pitch in bytes.

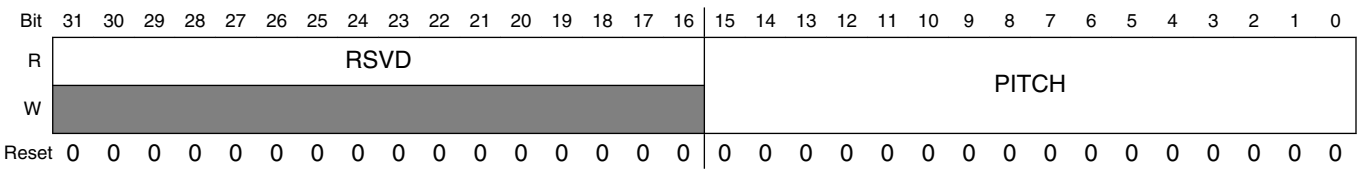
Any byte value will indicate the vertical pitch of the PS source frame buffer. This value will be used in PS pixel address calculations. This value has no relation to the UL and LR registers. It specifies how many bytes are between two vertically adjacent pixels in the input PS surface. For multi-plane formats, the Y buffer pitch should be programmed. For 2-plane YUV422, the UV pitch is the same as the Y pitch. For 3-plane YUV422, the U

and V pitch is 1/2 the Y pitch. For 2-plane YUV420, the UV pitch is 1/2 the Y pitch. For 3-plane YUV420, the U and V pitch is 1/4 the Y pitch. All source buffers should comply with these U and V resolution reductions with respect to their Y source buffers.

EXAMPLE

```
PXP_PS_PITCH_WR( 64 * 4 );  // The output buffer pitch is 64 pixels times 32 bits per pixel
```

Address: 21C_C000h base + F0h offset = 21C_C0F0h



PXP_PS_PITCH field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

39.7.17 PS Background Color (PXP_PS_BACKGROUND)

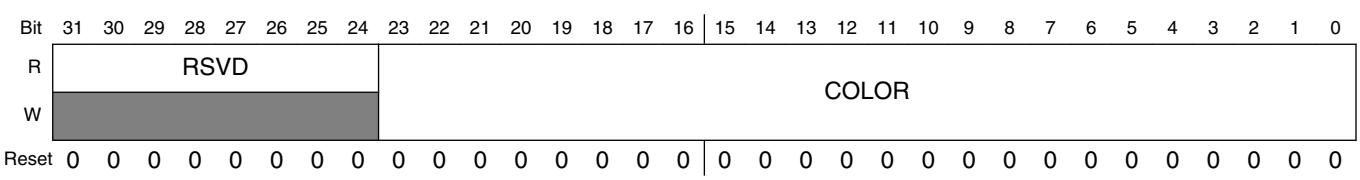
PS Background Pixel Color. This register provides a pixel value used when processing pixels outside of the region specified by the PS Coordinate registers. This value can effectively be used to set the color of the letterboxing region around the PS image.

This register contains a pixel value to be used for any PS pixels that fall outside the PS extents. This is effectively a background or letterbox color. The CSC1 control and datapath pixel format should be considered when selecting the background color.

EXAMPLE

```
PXP_PS_BACKGROUND_WR(0x00000000);  // letterbox is black
PXP_PS_BACKGROUND_WR(0x00800000);  // letterbox is dark red
PXP_PS_BACKGROUND_WR(0x00008000);  // letterbox is dark green
PXP_PS_BACKGROUND_WR(0x00000080);  // letterbox is dark blue
```

Address: 21C_C000h base + 100h offset = 21C_C100h



PXP_PS_BACKGROUND field descriptions

Field	Description
31–24 RSVD	Reserved, always set to zero.
COLOR	Background color (in 24bpp format) for any pixels not within the buffer range specified by the PS ULC/LRC.

39.7.18 PS Scale Factor Register (PXP_PS_SCALE)

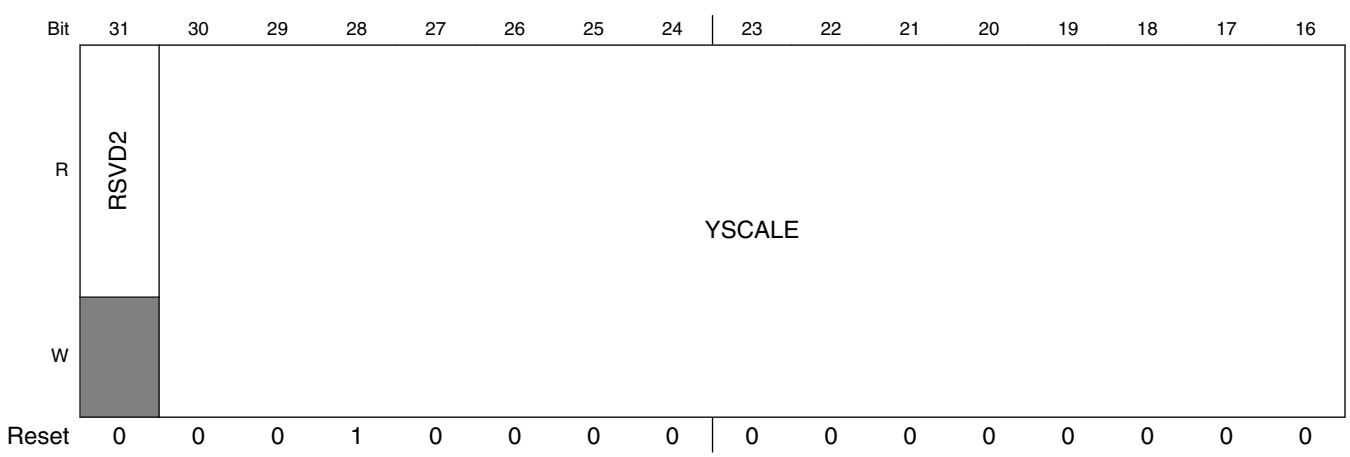
PS Scale Factor. This register provides the scale factor for the PS buffer.

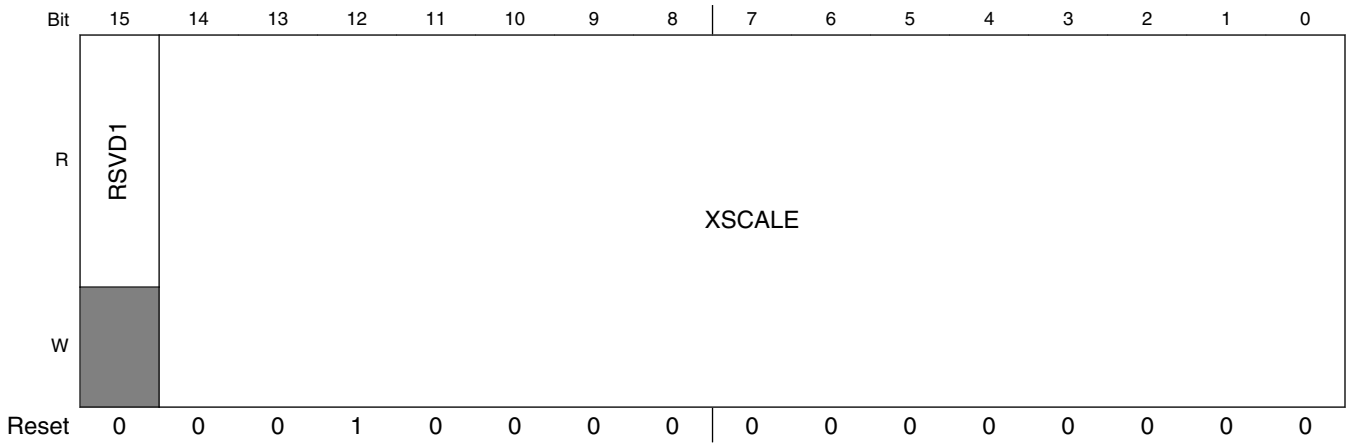
The maximum down scaling factor is 1/2 such that the output image in either axis is 1/2 the size of the source. The maximum up scaling factor is 2^12 for either axis. The reciprocal of the scale factor should be loaded into this register. To reduce the PS buffer by a factor of two in the output frame buffer, a value of 10.0000_0000_0000 should be loaded into this register. To scale up by a factor of 4, the value of 1/4, or 00.0100_0000_0000, should be loaded into this register. To scale up by 8/5, the value of 00.1010_0000_0000 should be loaded.

EXAMPLE

```
PXP_PS_SCALE_WR(0x10001000);  // 1:1  scaling (0x1.000)
PXP_PS_SCALE_WR(0x08000800);  // 2x   scaling (0x0.800)
PXP_PS_SCALE_WR(0x20002000);  // 1/2x scaling (0x2.000)
```

Address: 21C_C000h base + 110h offset = 21C_C110h





PXP_PS_SCALE field descriptions

Field	Description
31 RSVD2	Reserved, always set to zero.
30–16 YSCALE	This is a two bit integer and 12 bit fractional representation (##.####_####_####) of the Y scaling factor for the PS source buffer. The maximum value programmed should be 2 since scaling down by a factor greater than 2 is not supported with the bilinear filter. Decimation and the bilinear filter should be used together to achieve scaling by more than a factor of 2.
15 RSVD1	Reserved, always set to zero.
XSCALE	This is a two bit integer and 12 bit fractional representation (##.####_####_####) of the X scaling factor for the PS source buffer. The maximum value programmed should be 2 since scaling down by a factor greater than 2 is not supported with the bilinear filter. Decimation and the bilinear filter should be used together to achieve scaling by more than a factor of 2.

39.7.19 PS Scale Offset Register (PXP_PS_OFFSET)

PS Scale Offset. This register provides the initial scale offset for the PS buffer.

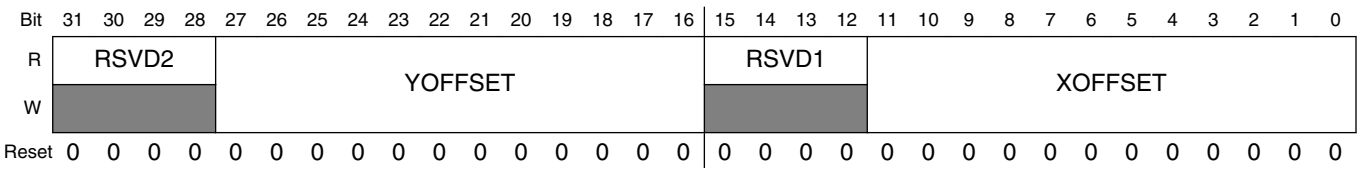
The X and Y offset provides the ability to access the source image with a per sub-pixel granularity. This provides the capability to use all source pixels to effect the output PS image. The fixed offset values can be used for sub-pixel adjustments in the bilinear scaling filter. For example, when scaling an image down by a factor of 2, an initial offset of 0x0 would result in sub-sampling every other pixel. If a fixed offset of 0x800 (1/2), all pixels are used in scaling the final output pixel value. In this case, the first output pixel would be the sum of (1/2*P0) + (1/2*P1). This fixed offset is applied after the decimation filter stage, and before the bilinear filter stage.

EXAMPLE

```
PXP_PS_SCALE_WR(0x2000_2000);   // 1/2x scaling (0x2.000)
PXP_PS_OFFSET_WR(0x0800_0800);  // half-pixel offset in both X and Y to ensure averaging
```

```
versus pixel decimation
```

Address: 21C_C000h base + 120h offset = 21C_C120h



PXP_PS_OFFSET field descriptions

Field	Description
31–28 RSVD2	Reserved, always set to zero.
27–16 YOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the Y scaling offset. This represents a fixed pixel offset which gets added to the scaled address to determine source data for the scaling engine.
15–12 RSVD1	Reserved, always set to zero.
XOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the X scaling offset. This represents a fixed pixel offset which gets added to the scaled address to determine source data for the scaling engine.

39.7.20 PS Color Key Low (PXP_PS_CLRKEYLOW)

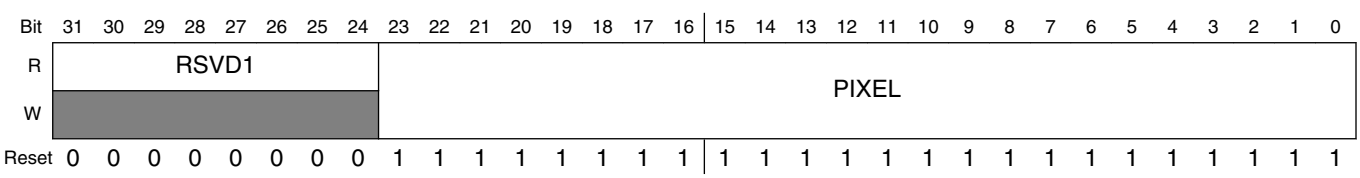
This register contains the color key low value for the PS buffer.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between PXP_PS_CLRKEYLOW and PXP_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the PXP_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                    // colorkey values between
PXP_PS_CLRKEYLOW_WR (0x008000); // medium green and
PXP_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 21C_C000h base + 130h offset = 21C_C130h



PXP_PS_CLRKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

39.7.21 PS Color Key High (PXP_PS_CLRKEYHIGH)

This register contains the color key high value for the PS buffer.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between PXP_PS_CLRKEYLOW and PXP_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the PXP_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                    // colorkey values between
PXP_PS_CLRKEYLOW_WR (0x008000); // medium green and
PXP_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 21C_C000h base + 140h offset = 21C_C140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								PIXEL																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_PS_CLRKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

39.7.22 Alpha Surface Control (PXP_AS_CTRL)

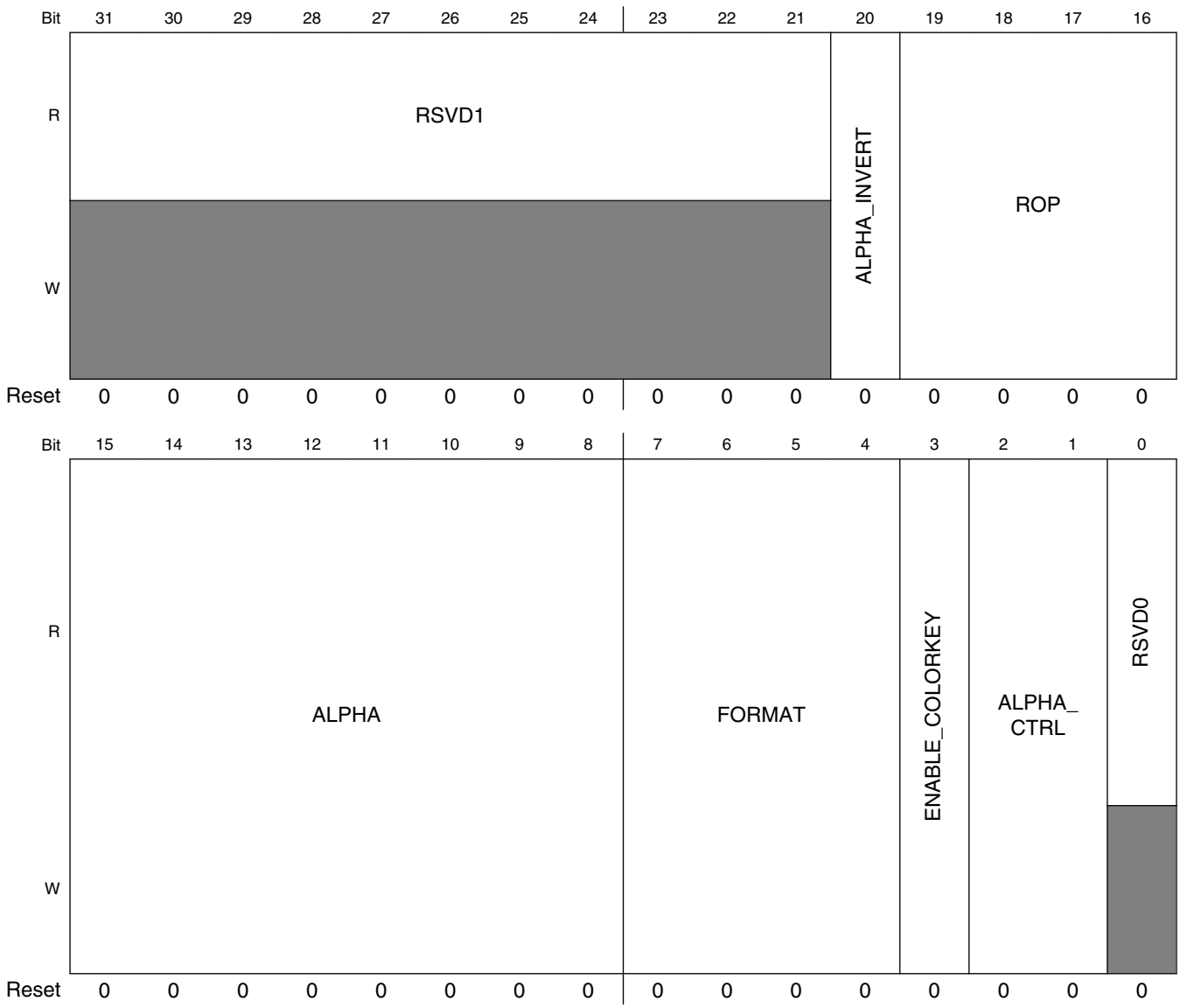
This register contains buffer control for the Alpha Surface 0 input buffer.

The Alpha Surface Parameter register provides additional controls for AS.

EXAMPLE

```
u32 asparam;
      asparam  = BF_PXP_ASPARAM_ENABLE    (1);
      asparam |= BF_PXP_ASPARAM_ALPHA_CTRL(BV_PXP_ASPARAM_ALPHA_CTRL__ROPs);
      asparam |= BF_PXP_ASPARAM_FORMAT    (BV_PXP_ASPARAM_FORMAT__ARGB8888);
      asparam |= BF_PXP_ASPARAM_ROP       (BV_PXP_ASPARAM_ROP__XORAS);
PXP_ASPARAM_WR(0,asparam);  // enable alpha surface to perform XOR ROP using RGB8888 AS
pixel format
```

Address: 21C_C000h base + 150h offset = 21C_C150h



PXP_AS_CTRL field descriptions

Field	Description
31–21 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_AS_CTRL field descriptions (continued)

Field	Description
20 ALPHA_INVERT	Setting this bit to logic 0 will not alter the alpha value. A logic 1 will invert the alpha value and apply (1-alpha) for image composition.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CTRL field. 0x0 **MASKAS** — AS AND PS 0x1 **MASKNOTAS** — nAS AND PS 0x2 **MASKASNOT** — AS AND nPS 0x3 **MERGEAS** — AS OR PS 0x4 **MERGENOTAS** — nAS OR PS 0x5 **MERGEASNOT** — AS OR nPS 0x6 **NOTCOPYAS** — nAS 0x7 **NOT** — nPS 0x8 **NOTMASKAS** — AS NAND PS 0x9 **NOTMERGEAS** — AS NOR PS 0xA **XORAS** — AS XOR PS 0xB **NOTXORAS** — AS XNOR PS
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE values are programmed in PXP_AS_CTRL[ALPHA_CTRL]. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when selected.
7–4 FORMAT	Indicates the input buffer format for AS. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x4 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x8 **ARGB1555** — 16-bit pixels with alpha 0x9 **ARGB4444** — 16-bit pixels with alpha 0xC **RGB555** — 16-bit pixels without alpha 0xD **RGB444** — 16-bit pixels without alpha 0xE **RGB565** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this alpha surface. Pixels found in the alpha surface colorkey range will be displayed as transparent (the PS pixel will be used).
2–1 ALPHA_CTRL	Determines how the alpha value is constructed for this alpha surface. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the AS pixel alpha value will be used to blend the AS with PS. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the alpha surface and PS pixels.
0 RSVD0	Reserved, always set to zero.

39.7.23 Alpha Surface Buffer Pointer (PXP_AS_BUF)

Alpha Surface 0 Buffer Address Pointer. This register points to the beginning of the Alpha Surface 0 input buffer.

This register is used to indicate the base address of the AS buffer.

EXAMPLE

```
u32* alpha_ptr;
PXP_ASn_WR(0,alpha_ptr);
```

Address: 21C_C000h base + 160h offset = 21C_C160h

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

PXP_AS_BUF field descriptions

Field	Description
ADDR	Address pointer for the alpha surface 0 buffer.

39.7.24 Alpha Surface Pitch (PXP_AS_PITCH)

This register contains the alpha surface pitch in bytes.

Any byte value will indicate the vertical pitch. This value will be used in AS pixel address calculations. This value has no relation to the UL and LR registers. It specifies how many bytes are between two vertically adjacent pixels in the input AS surface.

EXAMPLE

```
PXP_AS_PITCH_WR( 1920 * 4 );  // The output buffer pitch is HD resolution at 32 bits per
pixel
```

Address: 21C_C000h base + 170h offset = 21C_C170h

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

PXP_AS_PITCH field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

39.7.25 Overlay Color Key Low (PXP_AS_CLRKEYLOW)

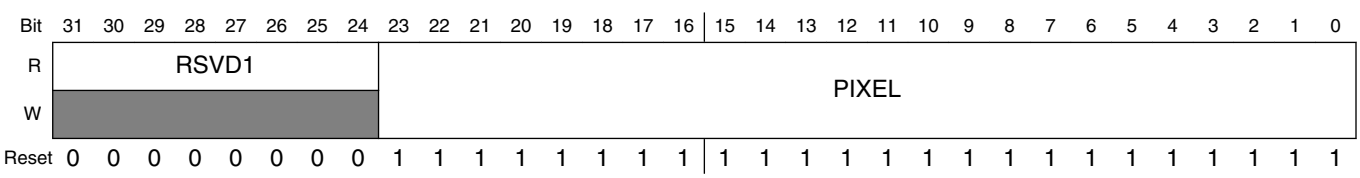
This register contains the color key low value for the AS buffer.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
PXP_AS_CLRKEYLOW_WR (0x000000); // black and
PXP_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 21C_C000h base + 180h offset = 21C_C180h



PXP_AS_CLRKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

39.7.26 Overlay Color Key High (PXP_AS_CLRKEYHIGH)

This register contains the color key high value for the AS buffer.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
PXP_AS_CLRKEYLOW_WR (0x000000); // black and
PXP_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 21C_C000h base + 190h offset = 21C_C190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								PIXEL																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_AS_CLRKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

39.7.27 Color Space Conversion Coefficient Register 0 (PXP_CSC1_COEF0)

This register contains color space conversion coefficients in two's compliment notation.

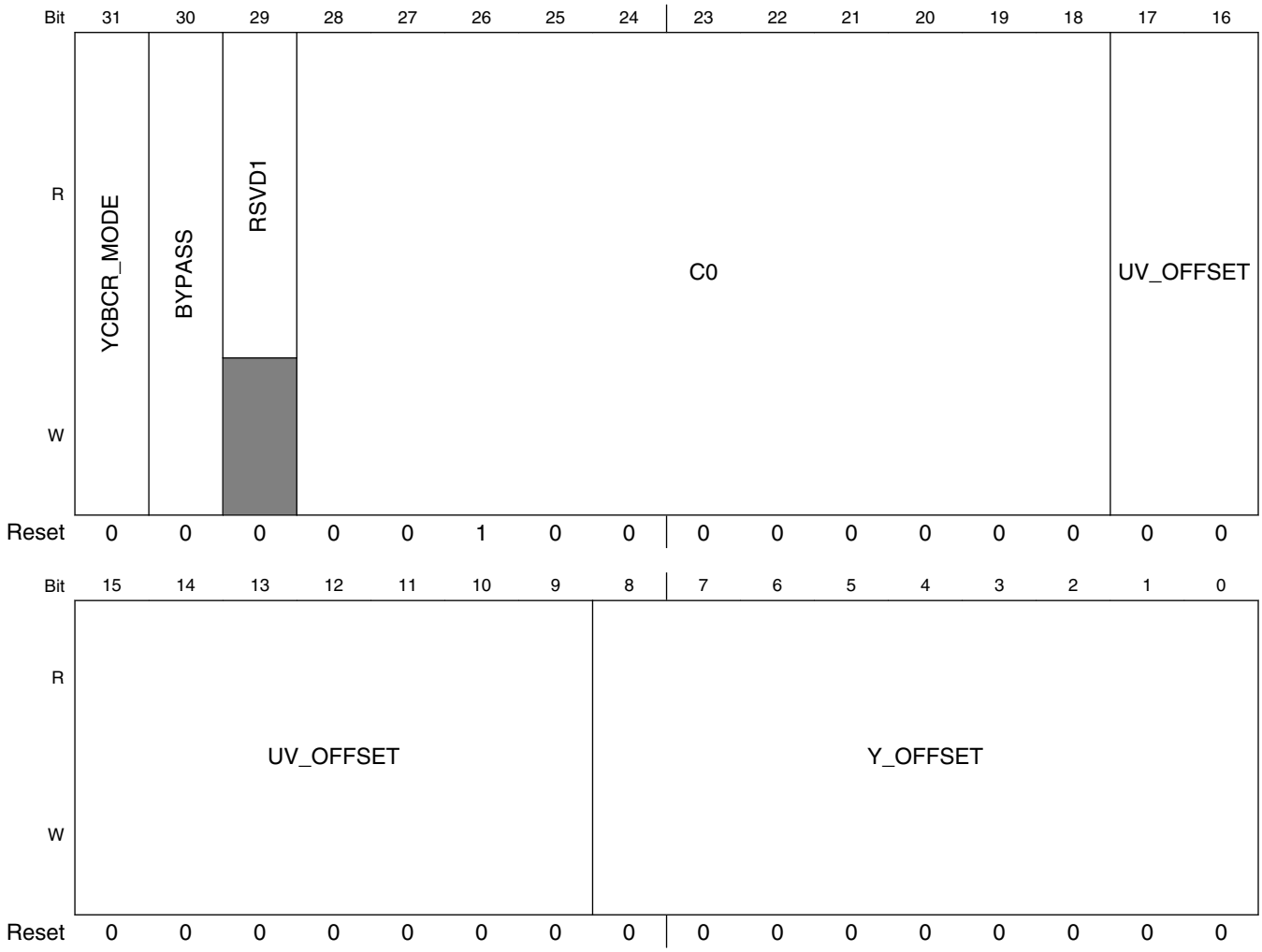
The coefficient 0 register contains coefficients used in the color space conversion algorithm. The Y and UV offsets are added to the source buffer to normalize them before the conversion. C0 is the coefficient that is used to multiply the luma component of the data for all three RGB components.

EXAMPLE

```
//  The equations used for Colorspace conversion are:
//     R = C0*(Y+YOFFSET)                    + C1(V+UV_OFFSET)
//     G = C0*(Y+YOFFSET) + C3(U+UV_OFFSET) + C2(V+UV_OFFSET)
//     R = C0*(Y+YOFFSET) + C4(U+UV_OFFSET)

   PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
   PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 21C_C000h base + 1A0h offset = 21C_C1A0h



PXP_CSC1_COEF0 field descriptions

Field	Description
31 YCBCR_MODE	Set to 1 when performing YCbCr conversion to RGB. Set to 0 when converting RGB to YUV data. This bit changes the behavior of the scaler when performing U/V scaling.
30 BYPASS	Bypass the CSC unit in the scaling engine. When set to logic 1, bypass is enabled and the output pixels will be in the YUV/YCbCr color space. When set to logic 0, the CSC unit is enabled and the pixels will be converted based on the programmed coefficients.
29 RSVD1	Reserved, always set to zero.
28–18 C0	Two's compliment Y multiplier coefficient. YUV=0x100 (1.000) YCbCr=0x12A (1.164)
17–9 UV_OFFSET	Two's compliment phase offset implicit for CbCr data. Generally used for YCbCr to RGB conversion. YCbCr=0x180, YUV=0x000 (typically -128 or 0x180 to indicate normalized -0.5 to 0.5 range)
Y_OFFSET	Two's compliment amplitude offset implicit in the Y data. For YUV, this is typically 0 and for YCbCr, this is typically -16 (0x1F0)

39.7.28 Color Space Conversion Coefficient Register 1 (PXP_CSC1_COEF1)

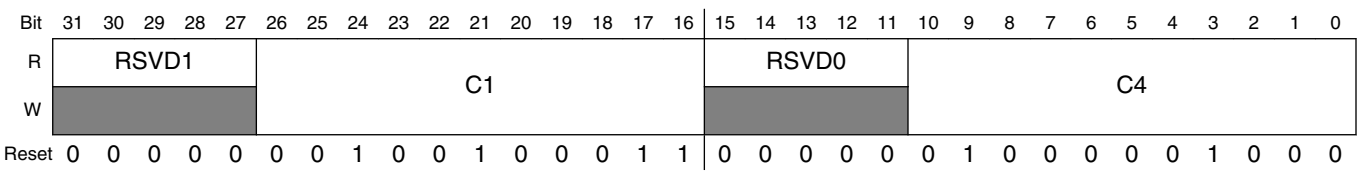
This register contains color space conversion coefficients in two's compliment notation.

The Coeffient 1 register contains coefficients used in the color space conversion algorithm. C1 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the red component. C4 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the blue component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 21C_C000h base + 1B0h offset = 21C_C1B0h



PXP_CSC1_COEF1 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C1	Two's compliment Red V/Cr multiplier coefficient. YUV=0x123 (1.140) YCbCr=0x198 (1.596)
15–11 RSVD0	Reserved, always set to zero.
C4	Two's compliment Blue U/Cb multiplier coefficient. YUV=0x208 (2.032) YCbCr=0x204 (2.017)

39.7.29 Color Space Conversion Coefficient Register 2 (PXP_CSC1_COEF2)

This register contains color space conversion coefficients in two's compliment notation.

The Coefficient 2 register contains coefficients used in the color space conversion algorithm. C2 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the green component. C3 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the green component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
// NOTE: The default values for the CSCCOEF2 register are incorrect.  C2 should be 0x76B and
C3 should be 0x79C for proper operation.

   PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
   PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 21C_C000h base + 1C0h offset = 21C_C1C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1					C2											RSVD0					C3										
W																																
Reset	0	0	0	0	0	1	1	1	1	0	0	1	1	0	1	1	0	0	0	0	0	1	1	1	0	1	1	0	1	1	0	0

PXP_CSC1_COEF2 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C2	Two's complement Green V/Cr multiplier coefficient. YUV=0x76B (-0.581) YCbCr=0x730 (-0.813)
15–11 RSVD0	Reserved, always set to zero.
C3	Two's complement Green U/Cb multiplier coefficient. YUV=0x79C (-0.394) YCbCr=0x79C (-0.392)

39.7.30 Color Space Conversion Control Register. (PXP_CSC2_CTRL)

This register contains the control registers to configure the CSC module.

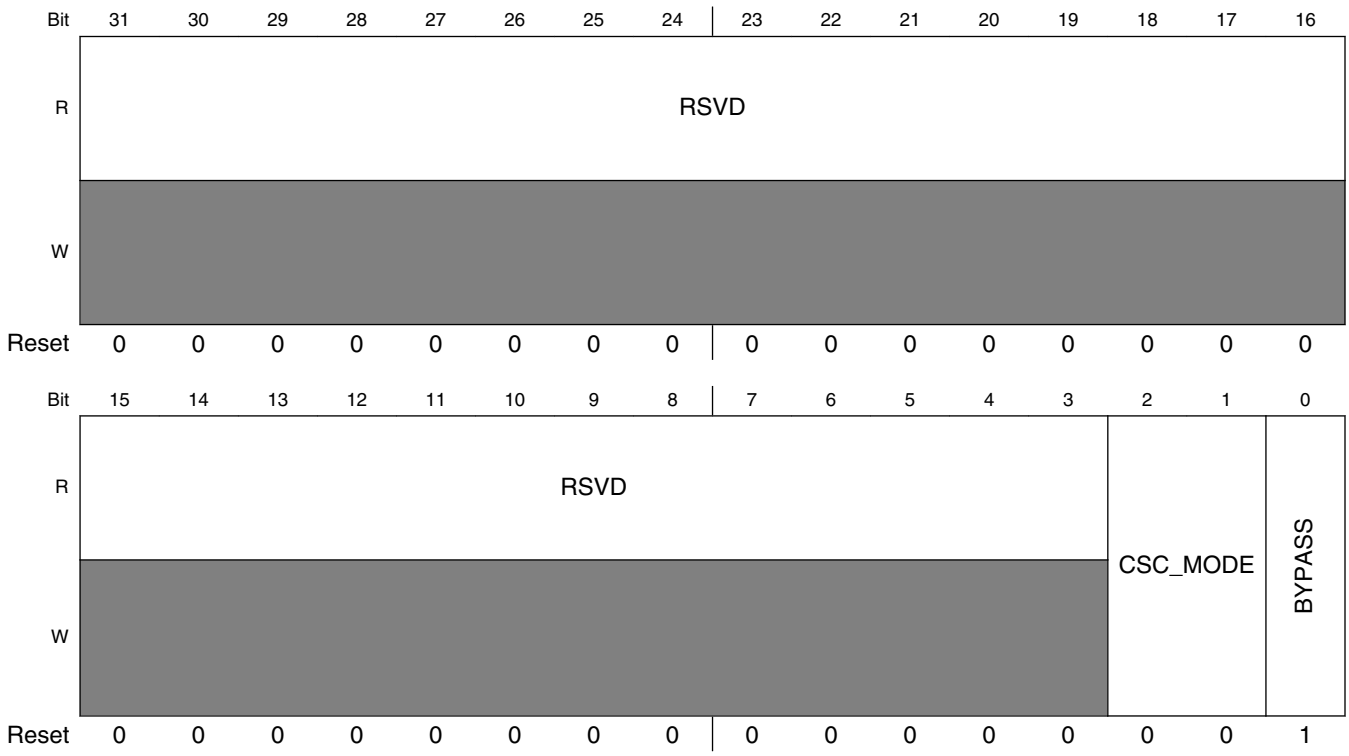
The CSC control register will configure the CSC module to perform color space conversion between the RGB/YUV/YCbCr color spaces.

EXAMPLE

```
//Converting from YUV/YCbCr color spaces to the RGB color space uses the
//following equation structure:
//
// R = A1(Y-D1) + A2(U-D2) + A3(V-D3)
```

```
// G = B1(Y-D1) + B2(U-D2) + B3(V-D3)
// B = C1(Y-D1) + C2(U-D2) + C3(V-D3)
//
//Converting from the RGB color space to YUV/YCbCr color spaces uses the
//following equation structure:
//
// Y = A1*R + A2*G + A3*B + D1
// U = B1*R + B2*G + B3*B + D2
// V = C1*R + C2*G + C3*B + D3
//
//All math is signed, so all coefficients come in as two's comp numbers
//
```

Address: 21C_C000h base + 1D0h offset = 21C_C1D0h



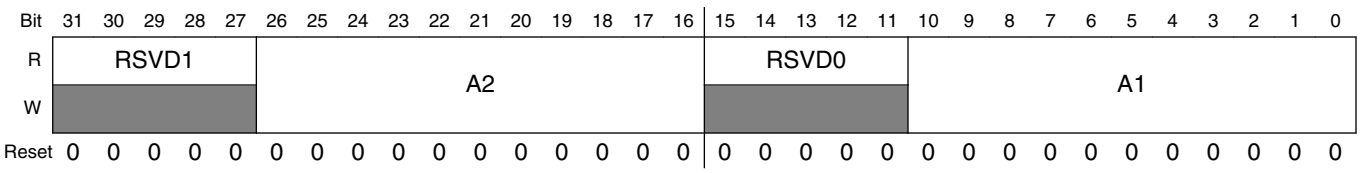
PXP_CSC2_CTRL field descriptions

Field	Description
31–3 RSVD	Reserved, always set to zero.
2–1 CSC_MODE	This field controls how the CSC unit operates on pixels when the CSC is not bypassed. 0x0 **YUV2RGB** — Convert from YUV to RGB. 0x1 **YCbCr2RGB** — Convert from YCbCr to RGB. 0x2 **RGB2YUV** — Convert from RGB to YUV. 0x3 **RGB2YCbCr** — Convert from RGB to YCbCr.
0 BYPASS	This bit controls whether the pixels entering the CSC2 unit get converted or not. When BYPASS is set, no operations occur on the pixels. When BYPASS is cleared, the selected CSC operation takes place.

39.7.31 Color Space Conversion Coefficient Register 0 (PXP_CSC2_COEF0)

This register contains color space conversion coefficients in two's complement notation.

Address: 21C_C000h base + 1E0h offset = 21C_C1E0h



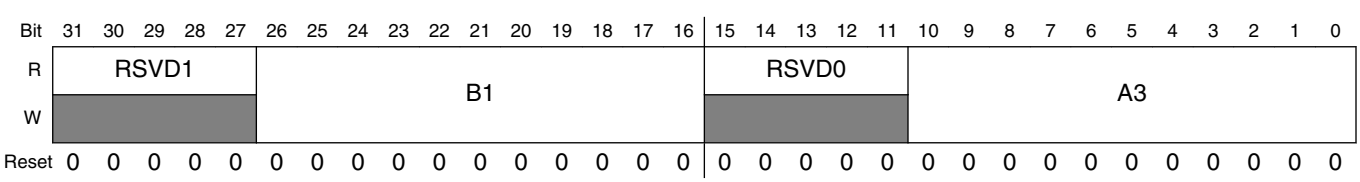
PXP_CSC2_COEF0 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 A2	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
A1	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

39.7.32 Color Space Conversion Coefficient Register 1 (PXP_CSC2_COEF1)

This register contains color space conversion coefficients in two's complement notation.

Address: 21C_C000h base + 1F0h offset = 21C_C1F0h



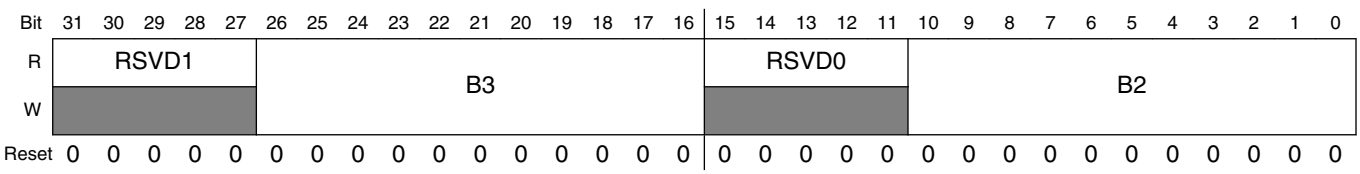
PXP_CSC2_COEF1 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 B1	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
A3	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

39.7.33 Color Space Conversion Coefficient Register 2 (PXP_CSC2_COEF2)

This register contains color space conversion coefficients in two's complement notation.

Address: 21C_C000h base + 200h offset = 21C_C200h



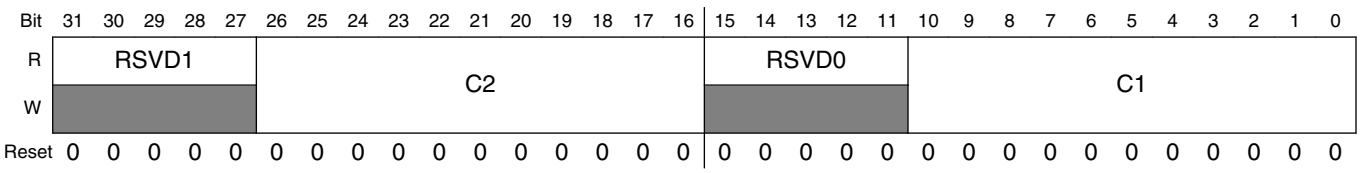
PXP_CSC2_COEF2 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 B3	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
B2	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

39.7.34 Color Space Conversion Coefficient Register 3 (PXP_CSC2_COEF3)

This register contains color space conversion coefficients in two's complement notation.

Address: 21C_C000h base + 210h offset = 21C_C210h



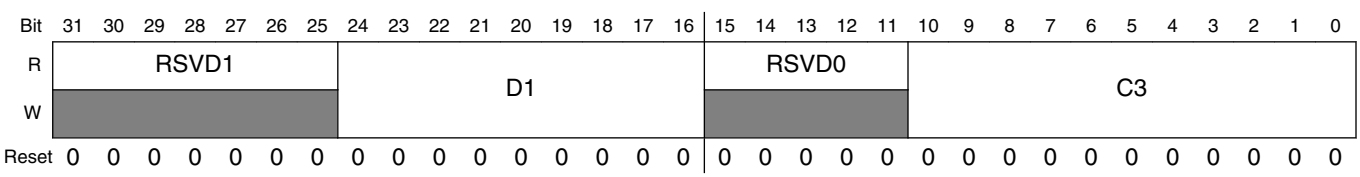
PXP_CSC2_COEF3 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C2	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
C1	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

39.7.35 Color Space Conversion Coefficient Register 4 (PXP_CSC2_COEF4)

This register contains color space conversion coefficients in two's complement notation.

Address: 21C_C000h base + 220h offset = 21C_C220h



PXP_CSC2_COEF4 field descriptions

Field	Description
31–25 RSVD1	Reserved, always set to zero.
24–16 D1	Two's complement coefficient integer offset to be added.

Table continues on the next page...

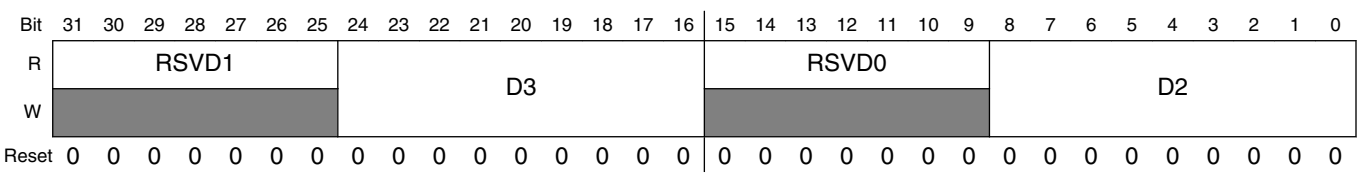
PXP_CSC2_COEF4 field descriptions (continued)

Field	Description
15–11 RSVD0	Reserved, always set to zero.
C3	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

39.7.36 Color Space Conversion Coefficient Register 5 (PXP_CSC2_COEF5)

This register contains color space conversion coefficients in two's complement notation.

Address: 21C_C000h base + 230h offset = 21C_C230h



PXP_CSC2_COEF5 field descriptions

Field	Description
31–25 RSVD1	Reserved, always set to zero.
24–16 D3	Two's complement coefficient integer offset to be added.
15–9 RSVD0	Reserved, always set to zero.
D2	Two's complement D1 coefficient integer offset to be added.

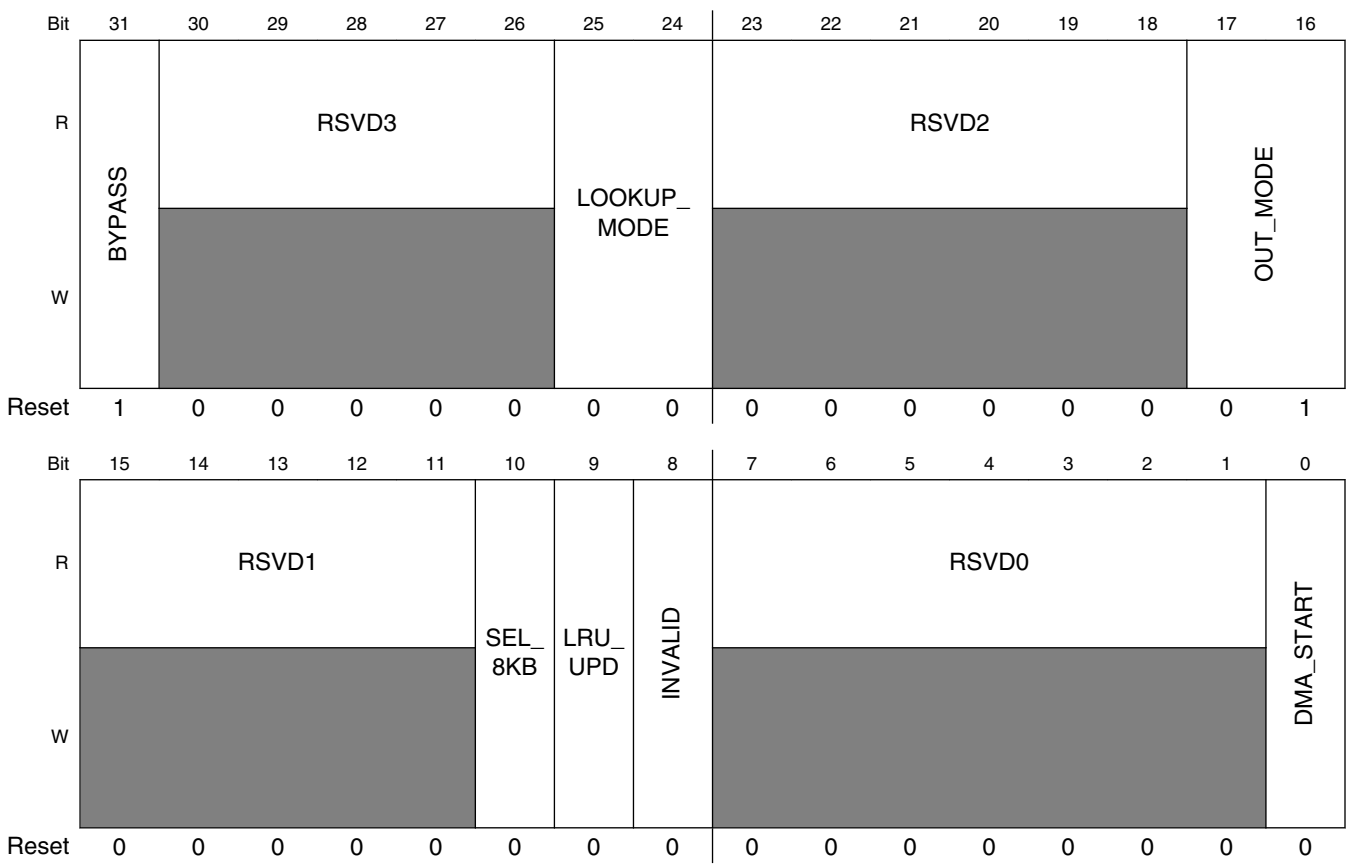
39.7.37 Lookup Table Control Register. (PXP_LUT_CTRL)

This register is used to access/control the Monochrome Lookup table.

The Y8 LUT input mode will take the high order data path byte and transform it using the LUT memory. This is an 8-bit to 8-bit transformation. The two low order bytes bypass the LUT and are not transformed, but bypassed without modification. This option can be used for monochrome gamma correction. The Direct Lookup mode will use the high nibble of each data byte and truncate the low nibble to generate the lookup address, i.e.

R[7:0]G[7:0]B[7:0] -> R[7:4]G[7:4]B[7:4]. 4K pixels (12-bit address) with 2 bytes per pixel is supported in this mode. Cached Lookup mode will use the high order bits, R[7:3],G[7:2],B[7:3] or RGB565, to address the cached LUT memory. 64KB LUT tables, using 16KB of internal LUT memory, can be indirectly transformed to 16-bit output pixels (as in RGBW4444/RGB565). This is used for 16bpp gamma correction or EPD color panel support. Cache misses are internally managed by the PXP LUT Cache controller.

Address: 21C_C000h base + 240h offset = 21C_C240h



PXP_LUT_CTRL field descriptions

Field	Description
31 BYPASS	Setting this bit will bypass the LUT memory resource completely. No pixel transfermations will occur at this stage of the PXP pixel rpocessing pipeline.
30–26 RSVD3	Reserved, always set to zero.
25–24 LOOKUP_MODE	Configure the input address for the 16KB LUT memory. The address into the LUT uses different parts of the pixel data path bytes. The data path is

Table continues on the next page...

PXP_LUT_CTRL field descriptions (continued)

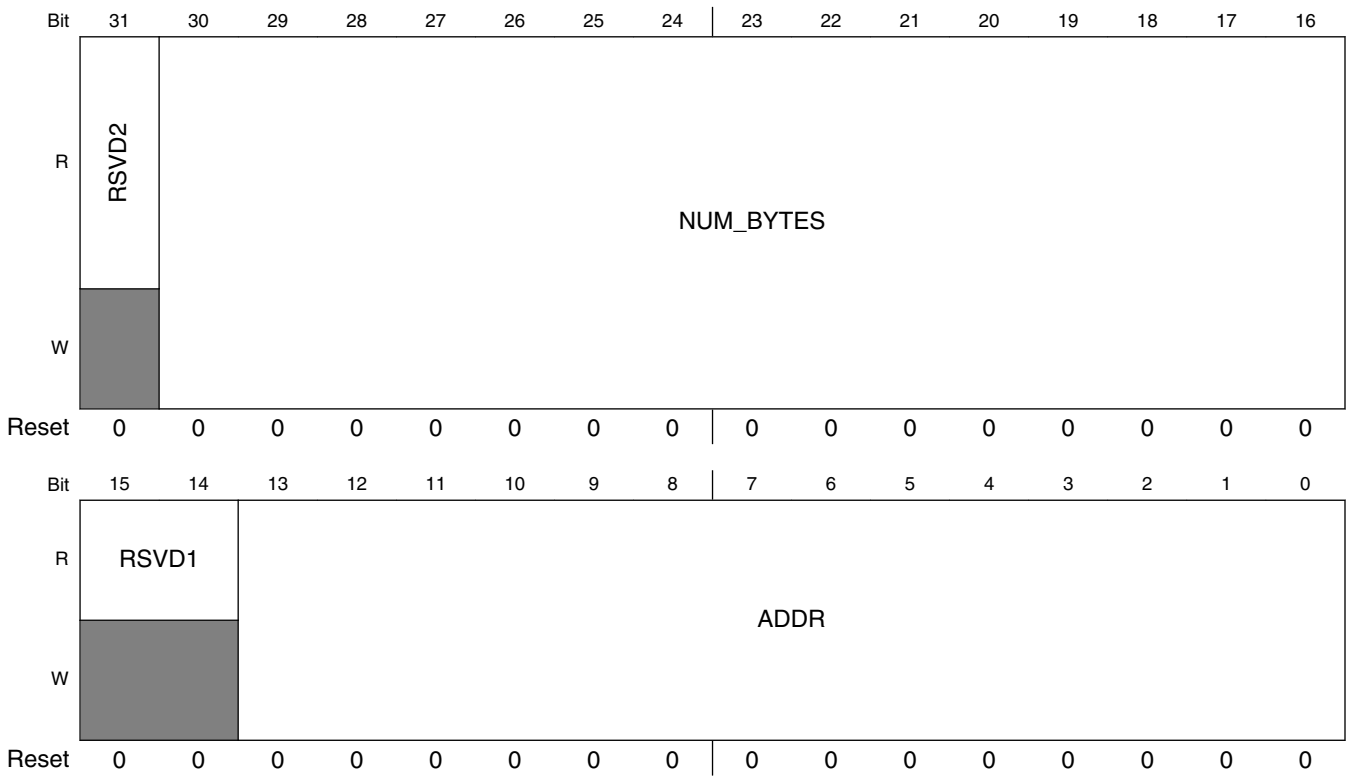
Field	Description
	defined as three bytes, conceptually as RGB/YUV/YCbCr[23:0]. Also referred to as R/Y[7:0],G/U[7:0],B/V[7:0] 0x0 **CACHE_RGB565** — LUT ADDR = R[7:3],G[7:2],B[7:3]. Use all 16KB of LUT for indirect cached 128KB lookup. 0x1 **DIRECT_Y8** — LUT ADDR = 16'b0,Y[7:0]. Use only the first 256 bytes of LUT. Only the Y, or third data path byte, is tranformed. 0x2 **DIRECT_RGB444** — LUT ADDR = R[7:4],G[7:4],B[7:4]. Use one 8KB bank of LUT selected by SEL_8KB. 0x3 **DIRECT_RGB454** — LUT ADDR = R[7:4],G[7:3],B[7:4]. Use all 16KB of LUT.
23–18 RSVD2	Reserved, always set to zero.
17–16 OUT_MODE	Select the output mode of operation for the LUT resource. There are four bytes [3-0] in the data path at the output of the LUT resource. Byte lane 3 is always bypassed and usually contains an alpha value. The LUT can be programmed to transform bytes 2,1,0 according to the options available in this field. 0x0 **RESERVED** — Reserved, not valid when using the LUT to transform pixels. 0x1 **Y8** — R/Y byte lane 2 lookup, bytes 1,0 bypassed. 0x2 **RGBW4444CFA** — Byte lane 2 = CFA_Y8, byte lane 1,0 = RGBW4444. 0x3 **RGB888** — RGB565->RGB888 conversion for Gamma correction.
15–11 RSVD1	Reserved, always set to zero.
10 SEL_8KB	Selects which 8KB bank of memory to use for direct 12bpp lookup modes. Logic 0 indicates first 8KB, logic 1 indicates second 8KB. Two direct LUT arrays can be stored and one can be selected for a given PXP operation.
9 LRU_UPD	Least Recently Used Policy Update Control: 1=> block LRU update for hit after miss. 0=> update LRU for all hits including hit after miss.
8 INVALID	Invalidate the cache LRU and valid bits. This bit will automatically reset when set to a logic 1.
7–1 RSVD0	Reserved, always set to zero.
0 DMA_START	Setting this bit will result in the DMA operation to load the PXP LUT memory based on PXP_LUT_ADDR_NUM_BYTES, PXP_LUT_ADDR_ADDR, and PXP_LUT_MEM_ADDR. This bit will automatically reset when set to a logic 1. Note: The LOOKUP_MODE must not be set to CACHE_RGB565 when starting and performing DMA transfers.

39.7.38 Lookup Table Control Register. (PXP_LUT_ADDR)

This register is used to access/control the Monochrome Lookup table.

The Y8 LUT input mode will take the high order data path byte and transform it using the LUT memory. This is an 8-bit to 8-bit transformation. The two low order bytes bypass the LUT and are not transformed, but bypassed without modification. This option can be used for monochrome gamma correction. The Direct Lookup mode will use the high nibble of each data byte and truncate the low nibble to generate the lookup address, i.e. R[7:0]G[7:0]B[7:0] -> R[7:4]G[7:4]B[7:4]. 4K pixels (12-bit address) with 2 bytes per pixel is supported in this mode. Cached Lookup mode will use the high order bits, R[7:3],G[7:2],B[7:3] or RGB565, to address the cached LUT memory. 64KB LUT tables, using 16KB of internal LUT memory, can be indirectly transformed to 16-bit output pixels (as in RGBW4444/RGB565). This is used for 16bpp gamma correction or EPD color panel support. Cache misses are internally managed by the PXP LUT Cache controller.

Address: 21C_C000h base + 250h offset = 21C_C250h



PXP_LUT_ADDR field descriptions

Field	Description
31 RSVD2	Reserved, always set to zero.
30–16 NUM_BYTES	Indicates the number of bytes to load via a DMA operation. This field must be divisable by 8 and the least significant 3 bits must be 0. The value 8 indicates load 8 bytes from the external address indicated by PXP_LUT_MEM_ADDR to the LUT memory location indicated by PXP_LUT_CTRL_ADDR.
15–14 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_LUT_ADDR field descriptions (continued)

Field	Description
ADDR	LUT indexed address pointer. This address into the LUT memory is always four byte aligned for PIO access, and eight byte aligned for DMA access. The least two significant bits are not used to drive the LUT memory array. For PIO LUT access, when the LUT data register is written, the contents of the LUT at the address specified by this address field will be loaded with a 32-bit DWORD. This address pointer will be incremented after the LUT data is written. This will provide recursive writes to the LUT data register to initialize the entire LUT array with recursive writes to the LUT data register. For DMA access, this register indicates the LUT memory address of the 8 byte QWORD to be loaded. When using the NUM_BYTES field to load more than 8 bytes, the register should be programmed with the first LUT memory location to be filled and each load of the LUT memory will increment this address field until NUM_BYTES has been loaded.

39.7.39 Lookup Table Data Register. (PXP_LUT_DATA)

This register is used to load data into the lookup table.

Address: 21C_C000h base + 260h offset = 21C_C260h

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

PXP_LUT_DATA field descriptions

Field	Description
DATA	Writing this field will load 4 bytes, aligned to four byte boundaries, of data indexed by the ADDR field of the PXP_LUT_CTRL register.

39.7.40 Lookup Table External Memory Address Register. (PXP_LUT_EXTMEM)

For DMA LUT memory loads, this is the base address from which data will be sourced to store into the LUT memory array. For Cached LUT memory pixel transformations, this register will store the base address of the full 64K pixel LUT translation table.

Address: 21C_C000h base + 270h offset = 21C_C270h

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

PXP_LUT_EXTMEM field descriptions

Field	Description
ADDR	This register contains the external memory address used for LUT memory operation. For DMA LUT memory loads, this is the base address from which data will be sourced to store into the LUT memory array. For Cached LUT memory pixel transformations, this register will store the base address of the full 64K pixel LUT translation table.

39.7.41 Color Filter Array Register. (PXP_CFA)

There are sixteen 2 bit values in this register each mapping a selected component to the output pixel. The two bit values are defined as 0=>R, 1=>G, 2=>B, and 3=>W. The first byte represents the repetitive pattern of RGBW pixels in the CFA for the first line segment of each processed PXP block. The second byte represents the pattern in the second line segment of the block, and so on. The first byte repeats two times for 8x8 macro block mode, and repeats four times for 16x16 block mode.

Address: 21C_C000h base + 280h offset = 21C_C280h

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

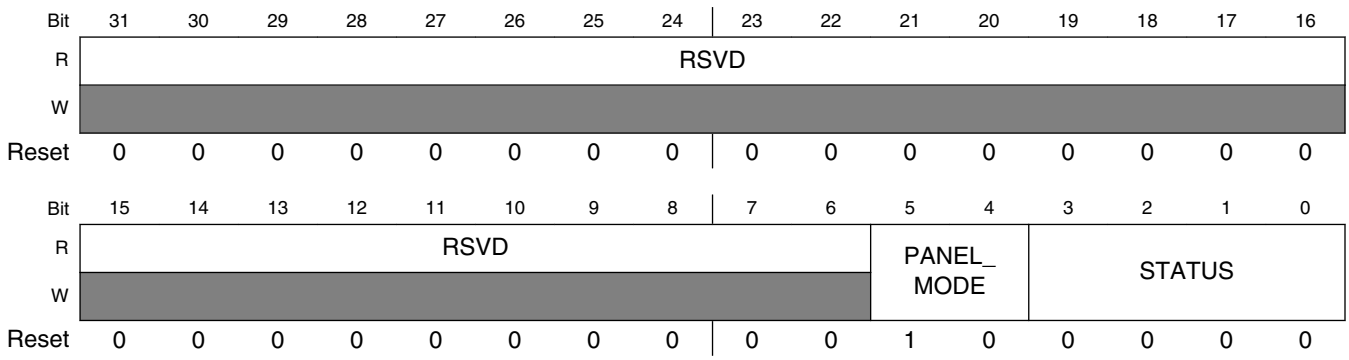
PXP_CFA field descriptions

Field	Description
DATA	This register contains the Color Filter Array pattern for decimation of RGBW4444 16 bit pixels to individual R, G, B, W values. The pattern represents a replicated 4x4 color filter array for the entire output frame buffer.

39.7.42 Histogram Control Register. (PXP_HIST_CTRL)

Provides control and status registers for the PXP's histogram classification algorithm.

Address: 21C_C000h base + 290h offset = 21C_C290h



PXP_HIST_CTRL field descriptions

Field	Description
31–6 RSVD	Reserved, always set to zero.
5–4 PANEL_MODE	This value is used to specify the number of bits used in comparisons when matching pixels to histogram bins. All comparator values MUST be programmed such that their bit width is consistent with the value of this register field. For instance, if GRAY16 is selected, comparator values must be in the range of 0x0-0xF. 0x0 **GRAY4** — 4-bit grayscale 0x1 **GRAY8** — 8-bit grayscale 0x2 **GRAY16** — 16-bit grayscale 0x3 **GRAY32** — 32-bit grayscale
STATUS	Indicates which histogram matched the processed bitmap. Bit[0] indicates that the bitmap pixels were fully contained within the HIST2 (black / white) histogram. Bit[1] indicates that the bitmap pixels were fully contained within the HIST4 (2-bit grayscale) histogram. Bit[2] indicates that the bitmap pixels were fully contained within the HIST8 (3-bit grayscale) histogram. Bit[3] indicates that the bitmap pixels were fully contained within the HIST16 (4-bit grayscale) histogram.

39.7.43 2-level Histogram Parameter Register. (PXP_HIST2_PARAM)

This register specifies the valid values for a 2-level histogram. If all pixels in a bitmap match the 2-level histogram values, STATUS[0] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 2A0h offset = 21C_C2A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RSVD1			VALUE1					RSVD0			VALUE0					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

PXP_HIST2_PARAM field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	White value for 2-level histogram
7–5 RSVD0	Reserved, always set to zero.
VALUE0	Black value for 2-level histogram

39.7.44 4-level Histogram Parameter Register. (PXP_HIST4_PARAM)

This register specifies the valid values for a 4-level histogram. If all pixels in a bitmap match the 4-level histogram values, STATUS[1] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 2B0h offset = 21C_C2B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3			VALUE3					RSVD2			VALUE2					RSVD1			VALUE1					RSVD0			VALUE0					
W																																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

PXP_HIST4_PARAM field descriptions

Field	Description
31–29 RSVD3	Reserved, always set to zero.
28–24 VALUE3	GRAY3 (White) value for 4-level histogram
23–21 RSVD2	Reserved, always set to zero.
20–16 VALUE2	GRAY2 value for 4-level histogram
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	GRAY1 value for 4-level histogram
7–5 RSVD0	Reserved, always set to zero.
VALUE0	GRAY0 (Black) value for 4-level histogram

39.7.45 8-level Histogram Parameter 0 Register. (PXP_HIST8_PARAM0)

This register specifies four of the valid values for an 8-level histogram. If all pixels in a bitmap match the 8-level histogram values, STATUS[2] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 2C0h offset = 21C_C2C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3			VALUE3					RSVD2			VALUE2					RSVD1			VALUE1					RSVD0			VALUE0				
W																																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HIST8_PARAM0 field descriptions

Field	Description
31–29 RSVD3	Reserved, always set to zero.
28–24 VALUE3	GRAY3 value for 8-level histogram

Table continues on the next page...

PXP_HIST8_PARAM0 field descriptions (continued)

Field	Description
23–21 RSVD2	Reserved, always set to zero.
20–16 VALUE2	GRAY2 value for 8-level histogram
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	GRAY1 value for 8-level histogram
7–5 RSVD0	Reserved, always set to zero.
VALUE0	GRAY0 (Black) value for 8-level histogram

39.7.46 8-level Histogram Parameter 1 Register. (PXP_HIST8_PARAM1)

This register specifies four of the valid values for an 8-level histogram. If all pixels in a bitmap match the 8-level histogram values, STATUS[2] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 2D0h offset = 21C_C2D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD7			VALUE7					RSVD6			VALUE6					RSVD5			VALUE5					RSVD4			VALUE4				
W																																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	1

PXP_HIST8_PARAM1 field descriptions

Field	Description
31–29 RSVD7	Reserved, always set to zero.
28–24 VALUE7	GRAY7 (White) value for 8-level histogram
23–21 RSVD6	Reserved, always set to zero.
20–16 VALUE6	GRAY6 value for 8-level histogram

Table continues on the next page...

PXP_HIST8_PARAM1 field descriptions (continued)

Field	Description
15–13 RSVD5	Reserved, always set to zero.
12–8 VALUE5	GRAY5 value for 8-level histogram
7–5 RSVD4	Reserved, always set to zero.
VALUE4	GRAY4 value for 8-level histogram

39.7.47 16-level Histogram Parameter 0 Register. (PXP_HIST16_PARAM0)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 2E0h offset = 21C_C2E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3			VALUE3					RSVD2			VALUE2					RSVD1			VALUE1					RSVD0			VALUE0				
W																																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

PXP_HIST16_PARAM0 field descriptions

Field	Description
31–29 RSVD3	Reserved, always set to zero.
28–24 VALUE3	GRAY3 value for 16-level histogram
23–21 RSVD2	Reserved, always set to zero.
20–16 VALUE2	GRAY2 value for 16-level histogram
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	GRAY1 value for 16-level histogram

Table continues on the next page...

PXP_HIST16_PARAM0 field descriptions (continued)

Field	Description
7–5 RSVD0	Reserved, always set to zero.
VALUE0	GRAY0 (Black) value for 16-level histogram

39.7.48 16-level Histogram Parameter 1 Register. (PXP_HIST16_PARAM1)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 2F0h offset = 21C_C2F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD7			VALUE7					RSVD6			VALUE6					RSVD5			VALUE5					RSVD4			VALUE4				
W																																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0

PXP_HIST16_PARAM1 field descriptions

Field	Description
31–29 RSVD7	Reserved, always set to zero.
28–24 VALUE7	GRAY7 value for 16-level histogram
23–21 RSVD6	Reserved, always set to zero.
20–16 VALUE6	GRAY6 value for 16-level histogram
15–13 RSVD5	Reserved, always set to zero.
12–8 VALUE5	GRAY5 value for 16-level histogram
7–5 RSVD4	Reserved, always set to zero.
VALUE4	GRAY4 value for 16-level histogram

39.7.49 16-level Histogram Parameter 2 Register. (PXP_HIST16_PARAM2)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 300h offset = 21C_C300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD11			VALUE11					RSVD10			VALUE10					RSVD9			VALUE9					RSVD8			VALUE8					
W																																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0

PXP_HIST16_PARAM2 field descriptions

Field	Description
31–29 RSVD11	Reserved, always set to zero.
28–24 VALUE11	GRAY11 value for 16-level histogram
23–21 RSVD10	Reserved, always set to zero.
20–16 VALUE10	GRAY10 value for 16-level histogram
15–13 RSVD9	Reserved, always set to zero.
12–8 VALUE9	GRAY9 value for 16-level histogram
7–5 RSVD8	Reserved, always set to zero.
VALUE8	GRAY8 value for 16-level histogram

39.7.50 16-level Histogram Parameter 3 Register. (PXP_HIST16_PARAM3)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 21C_C000h base + 310h offset = 21C_C310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD15			VALUE15					RSVD14			VALUE14					RSVD13			VALUE13					RSVD12			VALUE12					
W																																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	1	1	0	0

PXP_HIST16_PARAM3 field descriptions

Field	Description
31–29 RSVD15	Reserved, always set to zero.
28–24 VALUE15	GRAY15 (White) value for 16-level histogram
23–21 RSVD14	Reserved, always set to zero.
20–16 VALUE14	GRAY14 value for 16-level histogram
15–13 RSVD13	Reserved, always set to zero.
12–8 VALUE13	GRAY13 value for 16-level histogram
7–5 RSVD12	Reserved, always set to zero.
VALUE12	GRAY12 value for 16-level histogram

39.7.51 PXP Power Control Register. (PXP_POWER)

Address: 21C_C000h base + 320h offset = 21C_C320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CTRL																				ROT_MEM_LP_STATE			LUT_LP_STATE_WAY1_BANKN			LUT_LP_STATE_WAY0_BANKN			LUT_LP_STATE_WAY0_BANK0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_POWER field descriptions

Field	Description
31–12 CTRL	This register contains power control for the PXP.
11–9 ROT_MEM_LP_STATE	Select the low power state of the ROT memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
8–6 LUT_LP_STATE_WAY1_BANKN	Select the low power state of the LUT's WAY0-BANK0,1,2,3 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
5–3 LUT_LP_STATE_WAY0_BANKN	Select the low power state of the LUT's WAY0-BANK1,2,3 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
LUT_LP_STATE_WAY0_BANK0	Select the low power state of the LUT's WAY0-BANK0 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.

39.7.52 Next Frame Pointer (PXP_NEXT)

This register contains a pointer to a data structure used to reload the PXP registers at the end of the current frame.

To enable this functionality, software must write this register while the PXP is processing the current data frame (if the PXP is currently idle, this will also initiate an immediate load of registers from the pointer). The process of writing this register (WRITE operation) will set a semaphore in hardware to notify the control logic that a register reload operation must be performed when the current frame processing is complete. At

the end of a frame, the PXP will fetch the register settings from this location, signal an interrupt to software, then proceed with rendering the next frame of data. Software may cancel the reload operation by issuing a CLEAR operation to this register. SET and TOGGLE operations should not be used when addressing this register. All registers will be reloaded with the exception of the following: STAT, CSCCOEFn, NEXT. All other registers will be loaded in the order they appear in the register map. Once the pointer's contents have been loaded into the PXP's registers, the NEXT_IRQ interrupt will be issued (see the PXP_STATUS register).

EXAMPLE

```
// create register command structure in memory
  u32* pxp_commands0[48], pxp_commands1;
  u32  rc;

// initialize control structure for frame 0
  pxp_commands0[0] = ...; // CTRL
  pxp_commands0[1] = ...; // OUT Buffer
  ...
  pxp_commands0[47] = ..; // Overlay7 param2

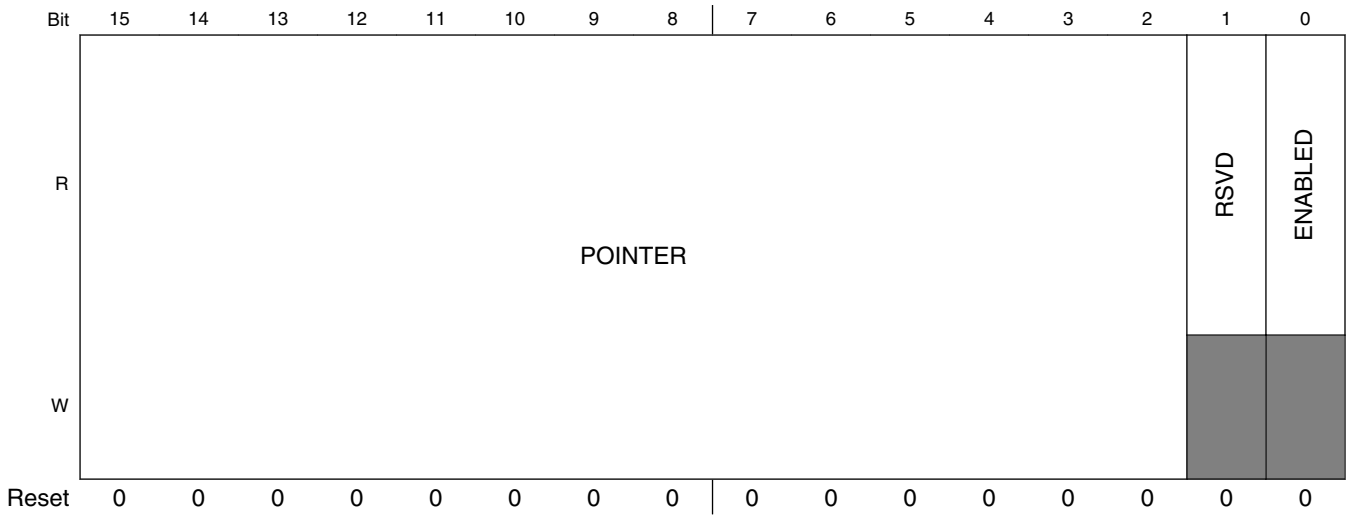
// initialize control structure for frame 1
  pxp_commands1[0] = ...; // CTRL
  pxp_commands1[1] = ...; // OUT Buffer
  ...
  pxp_commands1[47] = ..; // Overlay7 param2

// poll until a command isn't queued
  while (rc=PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  PXP_NEXT_WR(pxp_commands0);  // enable PXP operation 0 via command pointer

// poll until first command clears
  while (rc=PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  PXP_NEXT_WR(pxp_commands1);  // enable PXP operation 1 via command pointer
```

Address: 21C_C000h base + 400h offset = 21C_C400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	POINTER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PXP_NEXT field descriptions

Field	Description
31–2 POINTER	A pointer to a data structure containing register values to be used when processing the next frame. The pointer must be 32-bit aligned and should reside in on-chip or off-chip memory.
1 RSVD	Reserved, always set to zero.
0 ENABLED	Indicates that the "next frame" functionality has been enabled. This bit reflects the status of the hardware semaphore indicating that a reload operation is pending at the end of the current frame.

Chapter 40
Quad Serial Peripheral Interface (QuadSPI)

40.1 Overview

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one or two external serial flash devices, each with up to four bidirectional data lines. The following figure is a block diagram of the QuadSPI module.

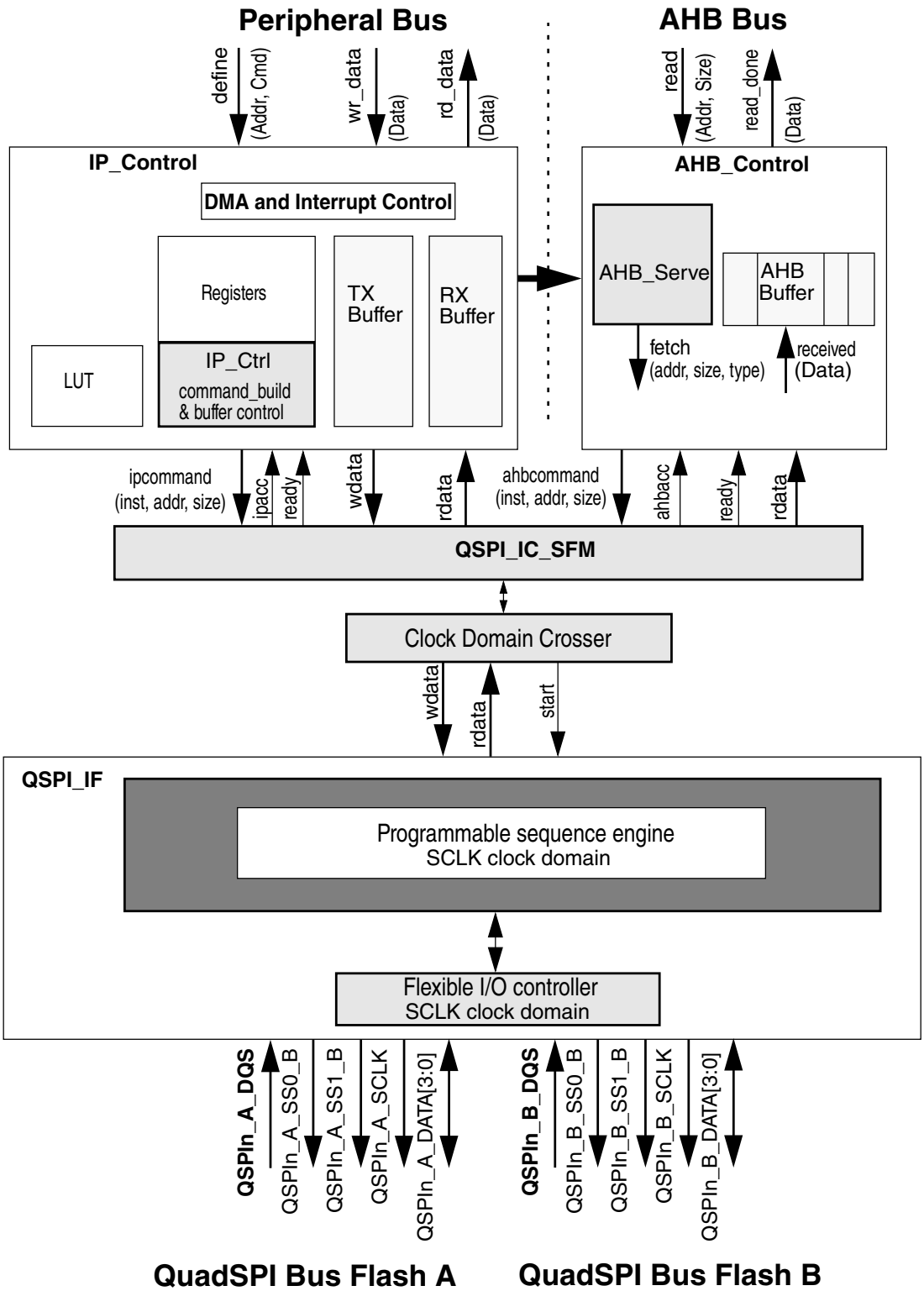


Figure 40-1. QuadSPI Block Diagram

The following figure describes the serial flash clock diagram in QuadSPI:

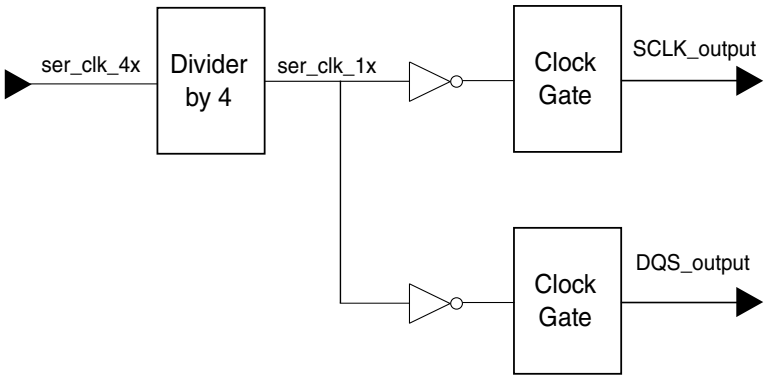


Figure 40-2. Flash Clock Diagram

ser_clk_4x is the serial clock root from CCM

ser_clk_1x' is generated clock from 'ser_clk_4x' divided by 4

SCLK_output is serial output clock generated from 'ser_clk_1x' with inverter and clock gate, it's used as Clock by external serial flash device.

DQS_output is serial flash data strobe signal generated from 'ser_clk_1x' with inverter and clock gate. This signal could be loopback from pad and used as sampling clock for serial flash data.

40.1.1 Features

The QuadSPI supports the following features:

- Flexible sequence engine to support various flash vendor devices.
- Single, dual, quad mode of operation.
- DDR/DTR mode wherein the data is generated on every edge of the serial flash clock.
- Support for flash data strobe signal for data sampling in DDR and SDR mode.
- Two identical serial flash devices can be connected and accessed in parallel for data read operations, forming one (virtual) flash memory with doubled readout bandwidth.
- DMA support to read RX Buffer data via AMBA AHB bus (64-bit width interface) or IP registers space (32-bit access).
    - Inner loop size of DMA access can be configured.
- Multi master accesses with priority

- Flexible and configurable buffer for each master

- Thirteen interrupt conditions (see Table 40-10)
- Memory mapped read access to connected flash devices.
- Programmable sequence engine to cater to future command/protocol changes and able to support all existing vendor commands and operations.
  - Supports 3-byte and 4-byte addressing.
  - TXFIFO size is 512Byte
  - RXFIFO size is 128Byte
  - AHB BUF size is 1KByte

40.1.2 QuadSPI Modes of Operation

This section provides information about the modes in which the QuadSPI module can be used.

40.1.2.1 Normal Mode

In this mode, one or two external serial flash memory devices can be accessed. Further details about this mode of operation can be found in Modes of Operation (Normal Mode).

40.1.2.2 Module Disable Mode

This mode is used for power management of the device containing the QuadSPI module. It is controlled by signals external to the QuadSPI. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode.

40.1.3 Acronyms and Abbreviations

The following table contains acronyms and abbreviations used in this document.

Table 40-1. Acronyms and Abbreviations

Terms	Description
AHB	Advanced High-performance Bus, version of AMBA
AMBA	Advanced Microcontroller Bus Architecture

Table continues on the next page...

Table 40-1. Acronyms and Abbreviations (continued)

Terms	Description
BE	Big Endian Byte Ordering
CS	Chip Select
DMA	Direct Memory Access
MB	Megabyte. Each MB is 1024 * 1024 bytes
IFM	Individual Flash Mode
PFM	Parallel Flash Mode
LSB	Least Significant Bit
MSB	Most Significant Bit
PCS	Peripheral Chip Select
QSPI, QuadSPI	Quad Serial Peripheral Interface
SCK	Serial Communications Clock
w1c	Write 1 to clear, writing a 1 to this field resets the flag

40.1.4 Glossary for QuadSPI module

Table 40-2. Glossary

Term	Definition
AHB Command	An AHB Command is a SFM Command triggered by a read access to the address range belonging to the memory mapped access defined in Table 40-7. Refer to AHB Commands for details.
Asserted	A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.
Clear	To clear a bit or bits means to establish logic level zero on the bit or bits.
Clock Phase	Determines when the data should be sampled relative to the active edge of SCK
Clock Polarity	Determines the idle state of the SCK signal.
Drain	To remove entries from a FIFO by software or hardware.
Endianness	Byte Ordering scheme.
Field	Two or more register bits grouped together.
Fill	To add entries to a FIFO by software or hardware.
Host	Refers to another functional block in the device containing the QuadSPI module
Instruction Code	8 bits defining the type of command to be executed.
IP Command	An IP Command is a SFM Command triggered by writing into the QSPI_IPCR[SEQID] field.
Logic level one	The voltage that corresponds to Boolean true (1) state.
Logic level zero	The voltage that corresponds to Boolean false (0) state.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
QSPI_AMBA_BASE	First address of QuadSPI address space on system memory map.
QSPI_ARDB_BASE	First address of QuadSPI Rx Buffer on system memory map.

Table continues on the next page...

Table 40-2. Glossary (continued)

Term	Definition
Set	To set a bit or bits means to establish logic level one on the bit or bits.
RX Buffer PUSH Event	Addition of valid entries into the RX Buffer. In the default case each Buffer PUSH Event adds 2 entries to the RX Buffer since the interface to the serial clock domain is 64 bits in width. Depending on the number of bytes read from the serial flash device it is possible for the very last Buffer PUSH Event that only one entry is added. The QSPI_RBSR[RDBFL] field is incremented by the number of entries added to the RX Buffer.
RX Buffer POP Event	Removal of valid entries from the RX Buffer. Each Buffer POP Event removes (QSPI_RBCT[WMRK] + 1) valid entries from the buffer. The QSPI_RBSR[RDBFL] field is decremented by the same number and the QSPI_RBSR[RDCTR] field is incremented accordingly.
Individual Flash Mode	Access to a single, individual serial flash device. Refer to Serial Flash Access Schemes for details.
Parallel Flash Mode	Read access to two serial flash devices attached to the QuadSPI module in parallel. Refer to Serial Flash Access Schemes for details.
SFM Command	Serial Flash Memory Command. A SFM command consists of an instruction code and all other parameters (for example, size or mode bytes) needed for that specific instruction code. Triggering a command either initiates a transaction on the external serial flash or results in an error. Refer to Table 40-20 for details on errors.
Single/Dual/Quad Instructions	Depending on the serial flash device connected to the QuadSPI module there will be instructions using a different number of data lines. • Single: Single line I/O with one data out and one data in line to/from the serial flash device. • Dual: Dual line I/O with two bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module • Quad: Quad line I/O with 4 bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI
Transaction	A transaction consists of all flags, data and signals in either direction to execute a command for an attached serial flash device. It is a combination of chip select, sclk, instruction code, address, mode- and/or dummy bytes, transmit and/or receive data.
LUT	Look-up table.

40.2 External Signals

The following table describes the external signals of QSPI:

Table 40-3. QSPI External Signals

Signal	Description	Pad	Mode	Direction
QSPI_A_DATA0	I/O data signal 1 port 0 for serial flash device A	NAND_READY_B	ALT0	I/O
QSPI_A_DATA1	I/O data signal 1 port 1 for serial flash device A	NAND_CE0_B	ALT0	I/O
QSPI_A_DATA2	I/O data signal 1 port 2 for serial flash device A	NAND_CE1_B	ALT0	I/O
QSPI_A_DATA3	I/O data signal 1 port 3 for serial flash device A	NAND_CLE	ALT0	I/O

Table continues on the next page...

Table 40-3. QSPI External Signals (continued)

Signal	Description	Pad	Mode	Direction
QSPI_A_SS0_B	Chip select 1 port 0 for serial flash device A	NAND_DQS	ALT0	O
QSPI_A_SS1_B	Chip select 1 port 1 for serial flash device A	NAND_DATA07	ALT0	O
QSPI_A_DQS	Data strobe signal 1 to serial flash device A	NAND_ALE	ALT0	I
QSPI_A_SCLK	Serial clock output 1 to serial flash device A	NAND_WP_B	ALT0	O
QSPI_B_DATA0	I/O data signal 1 port 0 for serial flash device B	NAND_DATA02	ALT0	I/O
QSPI_B_DATA1	I/O data signal 1 port 1 for serial flash device B	NAND_DATA03	ALT0	I/O
QSPI_B_DATA2	I/O data signal 1 port 2 for serial flash device B	NAND_DATA04	ALT0	I/O
QSPI_B_DATA3	I/O data signal 1 port 3 for serial flash device B	NAND_DATA05	ALT0	I/O
QSPI_B_SS0_B	Chip select 1 port 0 for serial flash device B	NAND_WE_B	ALT0	O
QSPI_B_SS1_B	Chip select 1 port 1 for serial flash device B	NAND_DATA00	ALT0	O
QSPI_B_DQS	Data strobe signal 1 to serial flash device B	NAND_DATA01	ALT0	I
QSPI_B_SCLK	Serial clock output 1 to serial flash device B	NAND_RE_B	ALT0	O

40.2.1 Driving External Signals

The different phases of serial flash access scheme are shown in the following figure.

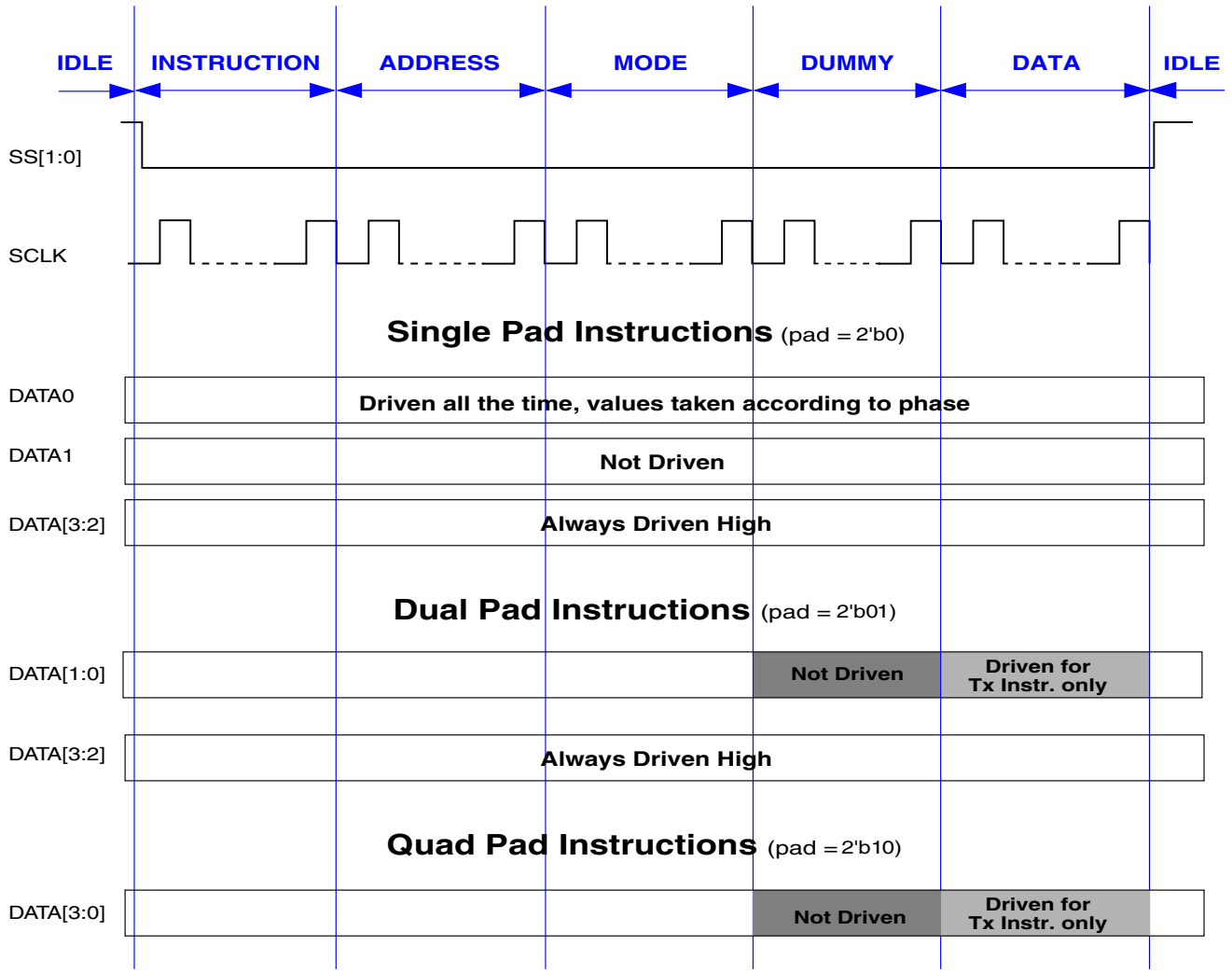


Figure 40-3. Serial Flash Access Scheme

The different phases and the I/O driving characteristics of the QuadSPI module are characterized in the following way:

- IDLE: Serial flash device not selected. No interaction with the serial flash device. All DATA signals driven.
- INSTRUCTION: Serial flash device selected. The instruction is sent to the serial flash device. All DATA signals are driven.
- ADDRESS: Serial Flash Address is sent to the device. All DATA signals are driven. Note that this phase is not applicable for all SFM Commands.
- MODE: Mode bytes are sent to the serial flash device. All DATA signals are driven. Note that this phase is not applicable for all SFM Commands.

- DUMMY: Dummy clocks are provided to the serial flash device. Refer to the Figure 40-3 for the DATA signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.
- DATA: Serial flash data are sent to or received from the serial flash device. Refer to the preceding figure for the DATA signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.

The SS[1:0] and SCLK signals are driven permanently throughout all the phases.

In Individual Flash Mode this applies to the selected flash device. In Parallel Flash Mode this applies to both serial flash devices simultaneously.

40.3 Memory Map and Register Definition

This section provides the memory map and register definitions of the QuadSPI module.

40.3.1 Register Write Access

This section describes the write access restriction terms that apply to all registers, which can be one of the following:

- **Register Write Access Restriction**

  For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the following table. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed.

  The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled.

Table 40-4. Register Write Access Restrictions

Condition	Description
Anytime	No write access restriction.
Disabled Mode	Write access only if *QSPI_MCR[MDIS] = 1*.

Table continues on the next page...

Table 40-4. Register Write Access Restrictions (continued)

Condition	Description
Normal Mode	Write access only if the module is in *Normal Mode*.

- **Register Write Access Requirements**

  All registers can be accessed with 8-bit, 16-bit, and 32-bit wide operations. For some of the registers, at least a 16/32-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each register affected.

40.3.2 Serial Flash Address Assignment

The serial flash address assignment may be modified by writing into Serial Flash A1 Top Address (QuadSPI_SFA1AD) and Serial Flash A2 Top Address (QuadSPI_SFA2AD) for device A and into Serial Flash B1Top Address (QuadSPI_SFB1AD) and Serial Flash B2Top Address (QuadSPI_SFB2AD) for device B. The following table shows how different access modes are related to the address specified for the next SFM Command. Note that this address assignment is valid for both IP and AHB commands.

Table 40-5. Serial Flash Address Assignment

Parameter	Function	Access Mode
QSPI_AMBA_BASE ((31:10) - 22 bits)	QuadSPI AHB base address	
TOP_ADDR_MEMA1(TPADA1)	Top address for the external flash A1 (first device of the dual die flash A, or the first of the two independent flashes sharing the IOFA)	Any access to the address space between TOP_ADDR_MEMA1 and QSPI_AMBA_BASE will be routed to **Serial Flash A1**
TOP_ADDR_MEMA2(TPADA2)	Top address for the external flash A2 (second device of the dual die flash A, or the second of the two independent flashes sharing the IOFA).	Any access to the address space between TOP_ADDR_MEMA2 and TOP_ADDR_MEMA1 will be routed to **Serial Flash A2**
TOP_ADDR_MEMB1(TPADB1)	Top address for the external flash B1 (first device of the dual die flash B, or the first of the two independent flashes sharing the IOFB)	Any access to the address space between TOP_ADDR_MEMB1 and TOP_ADDR_MEMA2 will be routed to **Serial Flash B1**
TOP_ADDR_MEMB2(TPADB2)	Top address for the external flash B2 (second device of the dual die flash B or the second of the two independent flashes sharing the IOFB)	Any access to the address space between TOP_ADDR_MEMB2 and TOP_ADDR_MEMB1 will be routed to **Serial Flash A2**

40.3.3 AMBA Bus Register Memory Map

QSPI_AMBA_BASE defines the address to be used as start address of the serial flash device as defined by the system memory map..

Table 40-6. QuadSPI AMBA Bus Memory Map

Address	Register Name
Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A	
QSPI_AMBA_BASE to (TOP_ADDR_MEMA2 - 0x01)	Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A Refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A for details and to Table 40-14 and Table 40-18 for information about the byte ordering.
Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B	
TOP_ADDR_MEMA2 to (TOP_ADDR_MEMB2 - 0x01)	Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B Refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B for details and to Table 40-14 and Table 40-18 for information about the byte ordering.
Parallel Flash Mode	
QSPI_AMBA_BASE to (TOP_ADDR_MEMB2 - 0x01)	Parallel Flash Mode Refer to Parallel Flash Mode for details and to Table 40-17 and Table 40-18 for information about the byte ordering.
AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)	
QSPI_ARDB_BASE to… (32 * 4 Byte) QSPI_ARDB_BASE + 0x0000_01FF	AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) Refer to Table 40-14 and Table 40-16 for information about the byte ordering.

Note

Any read access to non-implemented addresses will provide undefined results.

In case single die flash devices, TOP_ADDR_MEMA2 and TOP_ADDR_MEMB2 should be initialized/programmed to TOP_ADDR_MEMA1 and TOP_ADDR_MEMB1 respectively- in effect, setting the size of these devices to 0. This would ensure that the complete memory map is assigned to only one flash device.

Parallel Flash Mode is valid only for commands related to data read from the serial flash. The first device of flash A has to be paired with the first device of flash B and the second device of flash A has to be paired with the second device of flash B in parallel mode. Parallel mode is selected via the QSPI_BFGENCR[PAR_EN] bit for all masters in AHB driven mode and via the QSPI_IPCR[PAR_EN] in IP driven mode. In

parallel mode, the incoming address (SFAR address in case of IP initiated transactions and the incoming AHB address in case of AHB initiated transactions) is divided by 2 and sent to the two flashes connected in parallel.

Any IP Command other than data read in Parallel Flash Mode will result in the assertion of the QSPI_FR[IUEF] flag and any AHB Command other than data read in Parallel Flash Mode will result in the assertion of the QSPI_FR[ABSEF] flag.

In the Individual Flash Modes, the 3/4 address bytes (as programmed in the instruction/operand in the sequence) available for the flash address is determined by SFADR [23:0] or SFADR [31:0]as given in the table above.

In Parallel Flash Mode, both flashes are read with the same starting address of 3/4 (as programmed in the instruction/operand in the sequence) bytes in size. This address is derived from SFADR [24:1] or SFADR [31:1] as given in the table above. The LSB of the SFADR field is used to select the appropriate bits of both flash devices to combine the byte corresponding to the selected address.

40.3.4 AHB Bus Register Memory Map Descriptions

This chapter contains definitions of registers in the AMBA address space.

40.3.4.1 AHB Bus Access Considerations

It has to be noted that all logic in the QuadSPI module implementing the AHB Bus access is designed to read the content of an external serial flash device. Therefore the following restrictions apply to the QuadSPI module with respect to accesses to the AHB bus:

- Any write access is answered with the ERROR condition according to the AMBA AHB Specification. No write occurs.
- Any AHB Command resulting in the assertion of the QSPI_FR[ABSEF] flag is answered with the ERROR condition according to the AMBA_AHB specification. The resulting AHB Command is ignored.

- AHB Bus access types fully supported are NONSEQ and BUSY.
- AHB access type SEQ is treated in the same way like NONSEQ. Refer to the AMBA AHB Specification for further details.

40.3.4.2 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A

Starting with address QSPI_AMBA_BASE the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address QSPI_AMBA_BASE with increasing order. Assuming that a dual-die flash is connected on the first set of external pads, the address space is divided into two parts, one for each device of the dual die package. Refer to the following table for the address mapping. The byte ordering for 32 bit access is given in Table 40-14 and for 64 bit read access the byte ordering is given in Table 40-18.

Table 40-7. Memory Mapped Individual Flash Mode - Flash A Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash Byte Address	Flash Device
QSPI_AMBA_BASE + 0x00	QSPI_AMBA_BASE + 0x00	0x00_0000 to 0x00_0003	A1
QSPI_AMBA_BASE + 0x04		0x00_0004 to 0x00_0007	
…	…	…	
TOP_ADDR_MEMA1 - 0x08	TOP_ADDR_MEMA1 - 0x08	(TOP_ADDR_MEMA1- 0x08) to (TOP_ADDR_MEMA1 - 0x04 -0x01)	
TOP_ADDR_MEMA1 - 0x04		(TOP_ADDR_MEMA1 - 0x04) to (TOP_ADDR_MEMA1 - 0x01)	
TOP_ADDR_MEMA1 + 0x00	TOP_ADDR_MEMA1 + 0x00_0000	0x00_0000 to 0x00_0003	A2
TOP_ADDR_MEMA1 + 0x04		0x00_0004 to 0x00_0007	
…..	…	…	
TOP_ADDR_MEMA2 - 0x08	TOP_ADDR_MEMA2 - 0x08	(TOP_ADDR_MEMA2 - 0x08) to (TOP_ADDR_MEMA2 - 0x04 - 0x01)	
TOP_ADDR_MEMA2 - 0x04		(TOP_ADDR_MEMA2 - 0x04) to (TOP_ADDR_MEMA2 - 0x01)	

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

40.3.4.3 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B

Starting with address TOP_ADDR_MEMA2 the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address TOP_ADDR_MEMA2 with increasing order. Assuming that a dual-die flash is connected on the first set of external pads, the address space is divided into two parts, one for each device of the dual die package. Refer the following table for the address mapping. The byte ordering for 32 bit access is given in Table 40-14 and for 64 bit read access the byte ordering is given in Table 40-18.

Table 40-8. Memory Mapped Individual Flash Mode - Flash B Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash Byte Address	Flash Device
TOP_ADDR_MEMA2 + 0x00	TOP_ADDR_MEMA2 + 0x00	0x00_0000 to 0x00_0003	B1
TOP_ADDR_MEMA2 + 0x04		0x00_0004 to 0x00_0007	
…	…	…	
TOP_ADDR_MEMB1 - 0x08	TOP_ADDR_MEMB1 - 0x08	(TOP_ADDR_MEMB1- TOP_ADDR_MEMA2 - 0x08) to (TOP_ADDR_MEMB1 - TOP_ADDR_MEMA2 - 0x04 -0x01)	
TOP_ADDR_MEMB1 - 0x04		(TOP_ADDR_MEMB1- TOP_ADDR_MEMA2 - 0x04) to (TOP_ADDR_MEMB1- TOP_ADDR_MEMA2- 0x01)	
TOP_ADDR_MEMB1 + 0x00	TOP_ADDR_MEMB1 + 0x00_0000	0x00_0000 to 0x00_0003	B2
TOP_ADDR_MEMB1 + 0x04		0x00_0004 to 0x00_0007	
…..	…	…	
TOP_ADDR_MEMB2 - 0x08	TOP_ADDR_MEMA2 - 0x08	(TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1- 0x08) to (TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1 - 0x04 - 0x01)	
TOP_ADDR_MEMB2 - 0x04		(TOP_ADDR_MEMB2- TOP_ADDR_MEMB1 - 0x04) to (TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1 - 0x01)	

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

40.3.4.4 Parallel Flash Mode

Any of the AHB flexible-buffers can be configured to work in parallel flash mode by programming the QSPI_BFGENCR[PAR_EN] bit to '1'. When parallel mode is set, Flash A1 is paired with Flash B1 and Flash A2 is paired with Flash B2. In parallel mode, software should ensure that the size of Flash A1(A2) is equal to the size of Flash B1(B2).

Reads from any even AHB bus address provides bits [7:4] of both serial flash devices and reads from any odd AHB bus address provides bits [3:0] of both flash devices. Refer to the following table for the address mapping. The byte ordering for 32 bit access is given in Table 40-16 and for 64 bit read access the byte ordering is given in Table 40-18.

Table 40-9. Memory Mapped Parallel Flash Mode Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash A Byte Address	Serial Flash B Byte Address
QSPI_AMBA_BASE + 0x0000_0000	QSPI_AMBA_BASE + 0x00 For details, please refer to Parallel mode and Dual Die Flashes.	0x00_0000 - 0x00_0001	0x00_0000 - 0x00_0001
QSPI_AMBA_BASE + 0x0000_0004		0x00_002 - 0x00_0003	0x00_0002 - 0x00_0003
QSPI_AMBA_BASE + 0x0000_0008	QSPI_AMBA_BASE + 0x08	0x00_0004 - 0x00_0005	0x00_0004 - 0x00_0005
QSPI_AMBA_BASE + 0x0000_000C		0x00_0006 - 0x00_0007	0x00_0006 - 0x00_0007
…	…	…	…
TOP_ADDR_MEMB2 - 0x08	TOP_ADDR_MEMB2 - 0x08	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x08)/2	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x08)/2
TOP_ADDR_MEMB2 - 0x04		(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x04)/2 +0x01	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE- 0x04)/2 + 0x01

The available address range covers 27 address bits, corresponding to 128 MB per flash device. The usable space depends from the size of the external serial flash devices. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

40.4 Interrupt Signals

The interrupt request lines of the QuadSPI module are mapped to the internal flags according to the following table.

Table 40-10. Assignment of Interrupt Request Lines

IRQ/DMA line	QSPI_FR Flag	Interrupt Description
ipi_int_tfff	TBFF	TX Buffer Fill
ipi_int_tcf	TFF	Peripheral Command Transaction Finished
ipi_int_rfdf	RBDF	RX Buffer Drain
ipi_int_overrun		Buffer Overflow/Underrun Error Logical OR from:
	RBOF	RX Buffer Overrun
	TBUF	TX Buffer Underrun
	ABOF	AHB Buffer Overflow
ipi_int_cerr		Serial Flash Command Error Logical OR from:
	IPAEF	Peripheral access while AHB busy Error
	IPIEF	Peripheral Command could not be triggered Error
	IPGEF	Peripheral access while AHB Grant Error
	IUEF	Peripheral Command Usage Error
ipi_int_ored	DLPFF, TBFF, TFF, ILLINE, RBDF, RBOF, TBUF, ABSEF, ABOF, IPAEF, IPIEF, IPGEF, IUEF	Logical OR from all the QSPI_FR flags mentioned

40.5 Functional Description

This section provides the functional information of the QuadSPI module.

40.5.1 Serial Flash Access Schemes

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one single or two external serial flash devices, each with up to 4 bidirectional data lines. Depending from the serial flash devices attached to the QuadSPI module the following access schemes are possible:

Table 40-11. Access Schemes for Serial Flash Data Access

Access Scheme	One Flash Device on Port A	One Flash Device on Port B	Two identical Flash Devices connected on Port A and Port B
Individual Flash Mode: Access to Flash A	Yes	N/a	Yes
Individual Flash Mode: Access to Flash B	N/a	Yes	Yes
Parallel Flash Mode: Read from Flash A and Flash B	N/a	N/a	Yes

Note

If two flash devices are accessed in Parallel Flash Mode, they are accessed with identical control signals. Special alignment on per-flash basis is **not** possible. It is within the responsibility of the application to ensure that the identical signals are applicable to both flash devices.

In Parallel Flash Mode, both external serial flash devices appear logically as one single memory doubled in size with respect to one individual flash device.

If two different flash devices are attached, they can be operated only in Individual Flash Mode.

In the Parallel Flash Mode, only data read commands are supported. Any other IP Command will result in an error condition signaled by the assertion of the QSPI_FR[IUEF] flag and any other AHB Command will result in the assertion of the QSPI_FR[ABSEF] flag.

In the Individual Flash Mode, all supported commands are available.

Unless explicitly noted, all the following descriptions relate to the Individual Flash Mode.

40.5.2 Modes of Operation

Refer to QuadSPI Modes of Operation for an overview over the possible operational modes of the QuadSPI block.

- Normal Mode can be used for write or read accesses to an external serial flash device.

- Serial Flash Write: Data can be programmed into the flash via the IP interface only. Refer to Flash Programming for further details.
- Serial Flash Read: Read the contents of the serial flash device. Two separate read channels are available via RX Buffer and AHB Buffer, see Flash Read.

- Stop Mode: The mode is used for power management. When a request is made to enter Stop Mode, the QuadSPI block acknowledges the request and completes the SFM Command in progress, then the system clocks to the QuadSPI block may be shut off
- Module Disable Mode: The mode is used for power management. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode. The module enters the mode by setting QSPI_MCR[MDIS].

40.5.3 Normal Mode

This mode is used to allow communication with an external serial flash device. Compared to the standard SPI protocol, this communication method uses up to 4 bidirectional data lines operating at high data rates. The communication to the external serial flash device consists of an instruction code and optional address, mode, dummy and data transfers. The flexible programmable core engine described below is immune to a wide variety of command/protocol differences in the serial flash devices provided by various flash vendors.

40.5.3.1 Programmable Sequence Engine

The core of the QuadSPI module is a programmable sequence engine that works on "instruction-operand" pairs. The core controller executes each programmed instruction sequentially. The complete list of instructions and the corresponding operands is given in the following table.

Table 40-12. Instruction set

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
CMD	6'd1	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads	8 bit command value	Provide the serial flash with operand on the number of pads specified

Table continues on the next page...

Table 40-12. Instruction set (continued)

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
ADDR	6'd2	2'd2 - Four pads	Number of address bits to be sent (for example, 8'd24 => 24 address bits required)	Provide the serial flash with address cycles according to the operand on the number of pads specified. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of SFAR in case of IPS initiated transactions .
DUMMY	6'd3		Number of dummy clock cycles (should be <= 64 cycles)	Provide the serial flash with dummy cycles as per the operand. The PAD information defines the number of pads in input mode. (for example, one pad implies that pad 1 is not driven, rest all are driven)
MODE	6'd4		8 bit mode value	Provide the serial flash with 8 bit operand on the number of pads specified
MODE2	6'd5	N=2'd{0,1}	2 bit mode value	Provide the serial flash with 2 bit operand on the number of pads[1] specified
MODE4	6'd6	N=2'd{0,1,2}	4 bit mode value	Provide the serial flash with 4 bit operand on the number of pads[2] specified
READ	6'd7	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Read data size in bytes (for AHB transactions, the user's application should ensure that data size is a multiple of 8 bytes)	Read data from flash on the number of pads specified. The data size may be overwritten by writing to the ADATSZ field of the QSPI_BUFxCR registers for AHB initiated transactions and IDATSZ field of IP Configuration Register (QuadSPI_IPCR) for IP initiated transactions.
WRITE	6'd8		Write data size in bytes	Write data on number of pads sepcified. The data size may be overwritten by writing to the IDATSZ field of IP Configuration Register (QuadSPI_IPCR) register
JMP_ON_CS	6'd9	NA	Instruction number	Every time the CS is deasserted, jump to the instruction pointed to by the operand. This instruction allows the programmer to specify the behavior of the controller when a new read transaction is initiated following a CS deassertion.
ADDR_DDR	6'd10	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Number of address bits to be sent (for example, 8'd24 => 24 address bits required)	Provide the serial flash with address cycles according to the operand on the number of pads specified at each clock edge of serial flash clock. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of QSPI_SFAR in case of IPS initiated transactions .
MODE_DDR	6'd11		8 bit mode value	Provide the serial flash with 8 bit operand on the number of pads specified at each clock edge of serial flash.
MODE2_DDR	6'd12	N=2'd{0}	2 bit mode value	Provide the serial flash with 2 bit operand on the number of pads specified at each clock edge of serial flash[3]
MODE4_DDR	6'd13	N=2'd{0,1}	4 bit mode value	Provide the serial flash with 4 bit operand on the number of pads specified at each clock edge of serial flash[4].

Table continues on the next page...

Table 40-12. Instruction set (continued)

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
READ_DDR	6'd14	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Read data size in bytes (for AHB transactions, the user's application should ensure that data size is in multiple of 8 bytes)	Read data from flash on the number of pads specified at each clock edge of serial flash. The data size may be overwritten by writing to the ADATSZ field of the QSPI_BUFxCR registers for AHB initiated transactions and IDATSZ field of IP Configuration Register (QuadSPI_IPCR) for IP initiated transactions
WRITE_DDR	6'd15		Write data size in bytes	Write data on the number of pads specified at each clock edge of serial flash. The data size may be overwritten by writing to the IDATSZ field of IP Configuration Register (QuadSPI_IPCR) register
DATA_LEARN[5]	6'd16		8 bit Data learning pattern	Find the correct sampling point with the data learning pattern. When this instruction is encountered, the QSPI_SMPR[DDRSMP] values are ignored and the controller finds the correct sampling point on its own by sampling the data learning pattern.[6]
STOP	8'd0	NA	NA	Stop execution; deassert CS

1. For a one pad instruction, MODE2 will take 2 serial flash clock cycles on the flash interface.
2. For a one pad instruction, MODE4 will take 4 serial flash clock cycles on the flash interface. For a 4 pad instruction, MODE4 will take 1 serial flash clock cycle on the flash interface.
3. For a one pad instruction, MODE2_DDR will take 1 serial flash clock cycle on the flash interface.
4. For a one pad instruction, MODE4_DDR will take 2 serial flash clock cycles on the flash interface. For a 4 pad instruction MODE4_DDR will take half a cycle on the serial flash interface.
5. Data learning is not implemented on this chip.
6. t is not recommended to have 0x00 or 0xFF as the data learning pattern.

A sequence of such instruction-operand pairs may be pre-populated in the LUT according to the device connected on board. Each instruction-operand pair is of 16 bits (2 bytes) each. Every sequence pre-programmed in the LUT is referred to by its index.

The programmable sequence engine allows the user to configure the QuadSPI module according to the serial flash connected on board. The flexible structure is easily adaptable to new command/protocol changes from different vendors.

40.5.3.2 Flexible AHB buffers

In order to reduce the latency of the reads for AHB masters, the data read from the serial flash is buffered in flexible AHB buffers. There are four such flexible buffers. The size of each of these buffers is configurable with the minimum size being 0 Bytes and maximum size being the size of the complete buffer instantiated.The size of buffer 0 is defined as being from 0 to QSPI_BUF0IND. The Size of buffer 1 is from QSPI_BUF0IND to

QSPI_BUF1IND, buffer2 is from QSPI_BUF1IND to QSPI_BUF2IND and buffer 3 is from QSPI_BUF2IND to the size of the complete buffer, which is given in the chip-specific QuadSPI information.

Each flexible AHB buffer is associated with the following

1. An AHB master. Optionally, buffer3 may be configured as an "all master" buffer by setting the QSPI_BUF3CR[ALLMST] bit. When buffer3 is configured in such a way, any access from a master not associated with any other buffer is routed to buffer3.
2. A datasize field representing the amount of data to be fetched from the flash on every "missed" access.

The master port number of every incoming request is checked and the data is returned/ fetched into the corresponding associated buffer. Every "missed" access to the buffer causes the controller to clear the buffer and fetch QSPI_BUFxCR[ADATSZ] amount of data from the serial flash.As such, there is no benefit in configuring a buffer size of greater than ADATSZ, as the locations greater than ADATSZ will never be used. For any AHB access, the sequence pointed to by the QSPI_BFGENCR[SEQID] field is used for the flash transaction initiated. The data is returned to the master as soon as the requested amount is read from the serial flash. The controller however, continues to prefetch the rest of the data in anticipation of a next consecutive request. Figure 40-4 shows the flexible AHB buffers.

The QSPI_BFGENCR[SEQID] field points to an index of the LUT. Refer to Look-up Table for details.

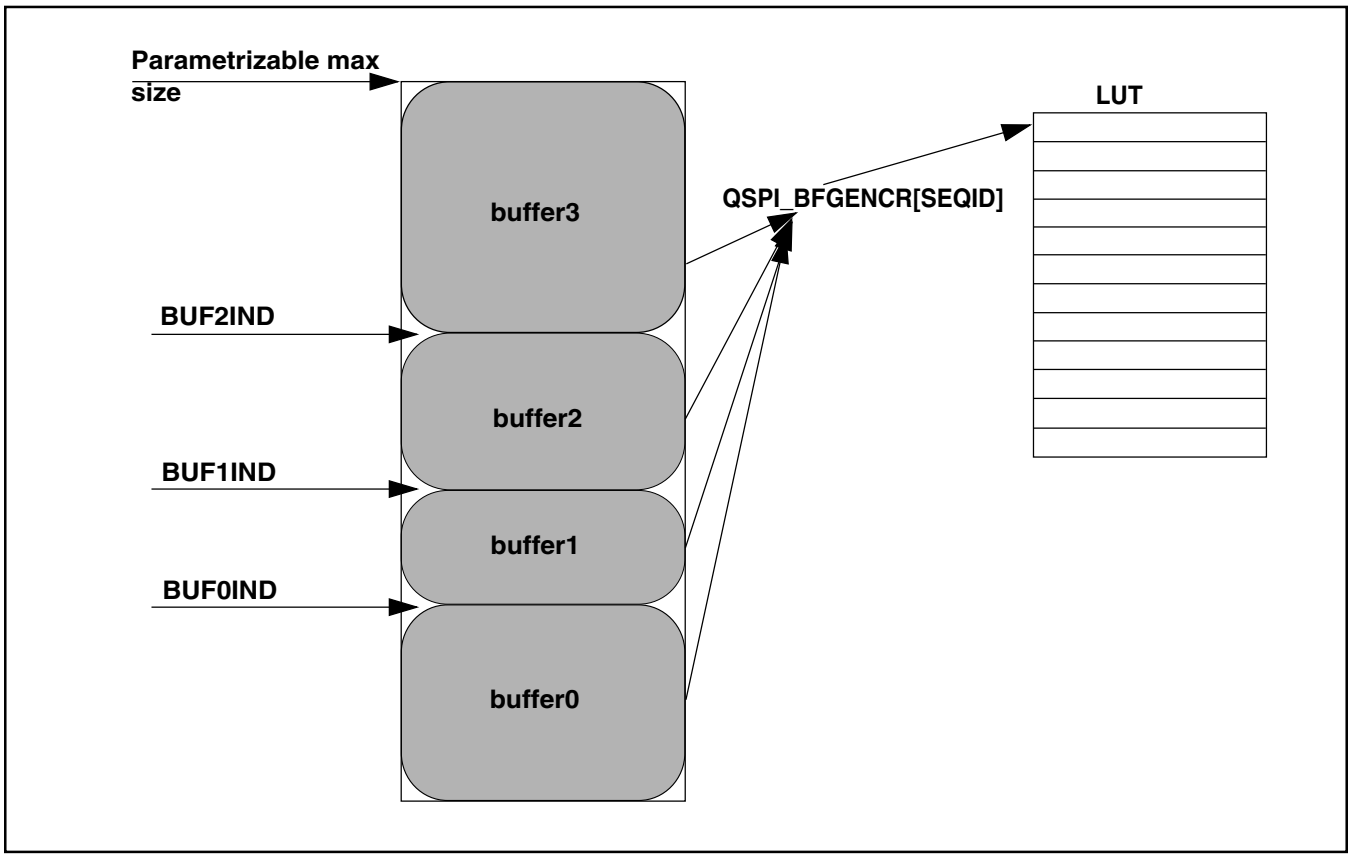


Figure 40-4. Flexible AHB Buffers

Buffer0 may optionally be configured to be associated with a high priority master by setting the QSPU_BUF0CR[HP_EN] bit. An access by a high priority master suspends any ongoing prefetch to any of the other buffers. The ongoing prefetch is suspended and the high priority master is serviced first. Once the high priority masters access completes, the suspended transaction is resumed (before any other AHB access is entertained). The status of the suspended buffer can be read from Sequence Suspend Status Register (QuadSPI_SPNDST).

40.5.3.3 Suspend-Abort Mechanism

Any low priority AHB access can be suspended by a high priority AHB master request. The ongoing transaction is suspended at 64 bit boundary. The suspended transaction is restarted after the high priority master is served and the high priority transaction including data prefetch is completed. While a transaction is in suspended state, it may be aborted if a transaction by the same suspended master is made to a location which is different from the location of the suspended transaction.

Any ongoing transaction is aborted if a request from the same master arrives for a location other than the location at which the transaction is going on. The abort can happen at any point of time.

40.5.3.4 Look-up Table

The Look-up-table or LUT consists of a number of pre-programmed sequences. Each sequence is basically a sequence of instruction-operand pairs which when executed sequentially generates a valid serial flash transaction. Each sequence can have a maximum of 8 instruction-operand pairs. The LUT can hold a maximum of sequences. The figure below shows the basic structure of the sequence in the LUT.

At reset, the index 0 of the look-up-table (LUT[0..3]) is programmed with a basic read sequence as given in Table 40-13. After reset the complete LUT may be reprogrammed according to the device connected on board.In order to protect its contents during a code runover the LUT may be locked, after which a write to the LUT will not be successful until it has been unlocked again.The key for locking or unlocking the LUT is **0x5AF05AF0**. The process for locking and un-locking the LUT is as follows:

Locking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register (QuadSPI_LUTKEY).
2. Write 0b01 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register locks the LUT.

Unlocking the LUT

1. Write the key **(0x5AF05AF0)** into the LUT Key Register (QuadSPI_LUTKEY)
2. Write 0b10 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register unlocks the LUT.

The lock status of the LUT can be read from QSPI_LCKCR[UNLOCK] and QSPI_LCKCR[LOCK] bit.

Some example sequences are defined in Example Sequences. The reset sequence at LUT index 0 is given in the following table.

Table 40-13. Reset sequence

Instruction	Pad	Operand	Comment
CMD	0x00	0x03	Read Data byte command on one pad
ADDR	0x00	0x18	24 Addr bits to be sent on one pad
READ	0x00	0x08	Read 64 bits
JMP_ON_CS	0x00	0x00	Jump to instruction 0 (CMD)

40.5.3.5 Issuing SFM Commands

Each access to the external device follows the same sequence:

1. The user must pre-populate the LUT with the serial flash command sequences that are required for the flash device being used.
2. The QuadSPI module starts executing the instructions in the sequence one by one. The transaction starts and the status bit QSPI_SR[BUSY] is set.
3. Communication with the external serial flash device is started and the transaction is executed.
4. When the transaction is finished (all transmit- and receive operations with the external serial flash device are finished) the status bit QSPI_SR[BUSY] is reset. In case of an IP Command the QSPI_FR[TFF] flag is asserted.

Further details are given in below in Flash Programming and Flash Read.

You can trigger the processing of SFM commands in the QuadSPI module in one of the following ways:

- **Using IP commands**

  For IP Commands the required components need to be written into the following registers:

  - Write the serial flash address to be used by the instruction into QSPI_SFAR, refer to Serial Flash Address Register (QSPI_SFAR). For IP Commands not related to specific addresses, the base address of the related flash need to be programmed.For example, for an instruction which does not require an address (i.e. write enable instruction) the SFAR should be programmed with the base address of the memory the command is to be sent to.

- Write the sequence ID and data size details in the IP Configuration Register (QSPI_IPCR).
- Note that the write into the QSPI_IPCR[SEQID] field must be the last step of the sequence. It is possible to combine all fields of the QSPI_IPCR into one single write. Refer to IP Configuration Register (QSPI_IPCR) for details.

Note that there are some conditions where no IP Command is executed after writing the QSPI_IPCR[SEQID] field and the write operation itself is ignored. They are described in Command Arbitration.

- **Using AHB commands**

  Any AHB memory mapped access is routed to one of the buffers depending on the master port number of the request. If the access is a "miss", a new serial flash transaction is started. The transaction is based on the sequence pointed to by the BFGENCR[SEQID] field as described in Flexible AHB buffers.

  An AHB access is termed memory mapped when the access is to the memory mapped serial flashes, as described in Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A and Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B.

  Again the possible error conditions are described in Command Arbitration .

40.5.3.6 Flash Programming

In all cases the memory sector to be written needs to be erased first. The programming sequence itself is then initiated in the following way:

1. Check that the TX Buffer is empty. If the QSPI_SR[TXEDA] bit is set then the TX Buffer must be cleared by writing 1 into the QSPI_MCR[CLR_TXF] bit.
2. Program the address related to the command in the QSPI_SFAR register.
3. Provide initial data for the program command into the circular buffer via register TX Buffer Data Register (QSPI_TBDR) . At least four word of data must be written into the TX Buffer up to a maximum of 32.
4. Program the QSPI_IPCR register to trigger the command. The QSPI_IPCR[SEQID] should point to an index of the LUT which has the flash program sequence pre-programmed.The IDATSZ field should be set to denote the size of the write.

5. Depending on the amount of data required, step 3 must be repeated until all the required data have been written into the QSPI_TBDR register.The QSPI_SR[TXFULL] can be used to check if the buffer is ready to receive more data. At any time, the QSPI_TBSR[TRCTR] field can be read to check how many words have been written actually into the TX Buffer.

Upon writing the QSPI_IPCR[SEQID] field (refer to step 4) the QuadSPI module will start to execute the programmed sequence. It is the responsibility of the software to ensure that a correct sequence is programmed into the LUT in accordance with the flash memory connected to the module. The data is fetched from the TX Buffer. It consists of **32** entries of 32-bits and is organized as a circular FIFO, whose read pointer is incremented by four after each fetch. When all data are transmitted, the QuadSPI module will return from 'busy' to 'idle'. However, this is not true for the external device since the internal programming is still ongoing. It is up to the user to monitor the relevant status information available from the serial flash device and to ensure that the programming is finished properly.

40.5.3.7 Flash Read

Host access to the data stored in the external serial flash device is done in two steps. First, the data must be read into the internal buffers and in the second step these internal buffers can be read by the host.

1. **Reading Serial Flash Data into the QuadSPI Module Internal Buffers**

   A read access to the external serial flash device can be triggered in two different ways:

   - **IP Command Read**: For **reading flash data into the RX Buffer** the user must provide the correct sequence ID in the QuadSPI_IPCR[SEQID] register. The sequence ID points to a sequence in the LUT. It is the responsibility of the software to ensure that a correct read sequence is programmed in the LUT in accordance with the serial flash device connected on board. The user should program the Serial Flash Address Register (QSPI_SFAR) and the IP Configuration Register (QSPI_IPCR) registers. All available read commands supported by the external serial flash are possible.

     Optionally it is possible to clear the RX Buffer pointer prior to triggering the IP Command by writing a 1 into the QuadSPI_MCR[CLR_RXF] field.

     From these inputs, the complete transaction is built when the QSPI_IPCR[SEQID] field is written. The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash

device into the RX Buffer. Since the read access is triggered by an IP command, the IP_ACC status bit and the BUSY bit are both set (both are located in the Status Register (QSPI_SR)). A count of the number of entries currently in the Rx Buffer can be obtained from QSPI_RBCT[RXBRD].

The communication with the external serial flash is stopped when the specified number of bytes has been read (successful completion of the transaction).

- **AHB Command Read**: For **reading flash data into the AHB Buffer** the user must set up a read access by a master to the address range in the system memory map which the external serial flash devices are mapped to. The user should also program the buffer registers corresponding to the AHB master initiating the request, this is depends on the configuration of the QSPI_RBCT[RXBRD]. The user should provide the correct sequence ID into the buffer generic configuration register (QSPI_BFGENCR). It is the responsibility of the software to ensure that a correct read sequence in programmed in the LUT in accordance with the serial flash device connected on board. Flash device selection and access mode are determined by the address accessed in the AHB address space associated to the QuadSPI module (refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A, Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B and Parallel Flash Mode.)

  On each AHB read access to the memory mapped area the valid data in the AHB Buffer is checked against the address requested in the actual read. When the AHB read request can't be served from the content of the AHB Buffer, the buffer is flushed and the sequence pointed to by the sequence ID is executed by the controller. The requested number of buffer entries defined in the QSPI_BUFxCR[ADATSZ] field is then fetched from the external serial flash device into the internal AHB Buffer. Since the read access is triggered via the AHB bus, the QSPI_SR[AHB_ACC] status bit is set driving in turn the QSPI_SR[BUSY] bit until the transaction is finished. The communication with the external serial flash is stopped when the specified number of entries has been filled.

2. **Data Transfer from the QuadSPI Module Internal Buffers**

   The data read out from the external serial flash device by the QuadSPI module is stored in the internal buffers.The means of accessing the data from the buffer differs depending on which buffer the data has been loaded to. Refer to Block Diagram for details about the two available buffers, the RX Buffer and the AHB Buffer, in the QuadSPI module:

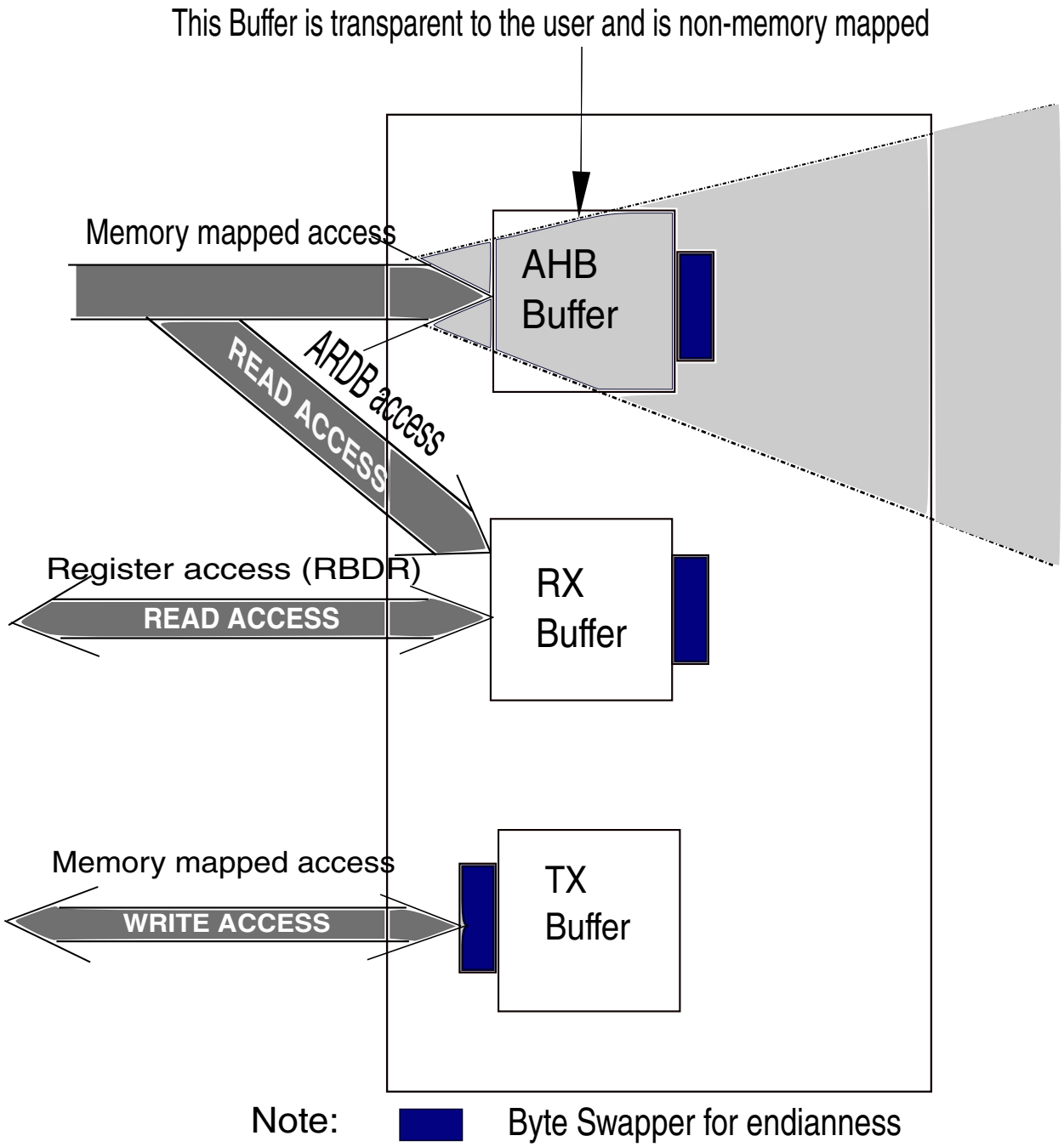


Figure 40-5. QuadSPI memory map

- The RX Buffer is implemented as FIFO of depth 32 entries of 4 bytes. Its content is accessible in two different address areas both referring to the identical data and the same physical memory.

  In the IPS address space in the area associated to QSPI_RBDR0 to QSPI_RBDR31

  In the AHB address space in the area associated to QSPI_ARDB0 to QSPI_ARDB31. Two successive entries are accessed with one single 64 bit AHB read operation.

RX Buffer operation can be summarized as follows: The QSPI_RBCT[WMRK] field determines at which fill level the RXWE bit is asserted and how many entries are removed from the RX Buffer on each Buffer POP operation. So the QSPI_SR[RXWE] bit indicates that the configured number of data entries is available in the RX Buffer and the QSPI_RBSR[RDBFL] field indicates how many valid entries are available in total. Note that the first entry (QSPI_RBDR0 or QSPI_ARDB0) always corresponds to the first valid entry in the RX Buffer. The software needs to manage the number of valid data bytes itself.

Further details can be found in RX Buffer Data Register (QuadSPI_RBDR*n*) and in AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31).

- **Flag-based Data Read of the RX Buffer** is done by polling the QSPI_SR[RXWE] bit. When it is asserted the valid entries can be read either via the IPS address space (QSPI_RBDRn) or the AHB address space (QSPI_ARDBn). A Buffer POP operation must be triggered by the application by writing a 1 into the QSPI_FR[RBDF] bit - this automatically updates the FIFO to point to the next entry as defined by RBCT[WMRK]. For example, if WMRK is set to 3, then the buffer will discard 16 bytes of data.
- **DMA controlled Data Read of the RX Buffer** is done by using the DMA module.The application must ensure that the DMA controller of the related device is programmed appropriately like it is described in DMA Usage.

  DMA controlled read out is triggered fully automatically by the assertion of the QSPI_SR[RXWE] bit. The related Buffer POP operation is also handled completely inside the QuadSPI module. Like in the case above, accessing the RX Buffer content either on QSPI_RBDRn or QSPI_ARDBn related addresses is equivalent.

- **AHB Buffer data read via memory mapped access:** This kind of access is done by reading one of the addresses assigned to the external serial flash device(s) within the range given in Table 40-6 table *under the condition that the data requested are already present in the AHB Buffer or it is currently being read from the serial flash device by the instruction in progress.* If this is not the case a memory mapped AHB command read is triggered as described above. If the requested data is already available in the AHB Buffer they are provided directly to the host.

  When AHB access are made to the flash memory mapped address, the data will be fetched and returned to the AHB interface. Till the data is being fetched the AHB interface would be stalled. As soon as the data from the requested address has been read by the QuadSPI module the AHB read access is served. So it is possible to run sequential reads from the AHB buffer at arbitrary speed without

the need to monitor any information about the availability of the data. Nevertheless this access scheme stalls the AHB bus for the time required to read the data from the serial flash device. A better way is (when it is known the access is sequential) to have a prefetch enabled (by programming the ADATSZ field) such that before the next sequential AHB access come, the data is already fetched into the buffer.

As long as the host restricts its accesses to the data already in the buffer and the data currently fetched from the serial flash, it is possible to run the host read from the AHB Buffer in parallel to the serial flash read into the AHB Buffer.

40.5.3.8 Byte Ordering of Serial Flash Read Data

In this paragraph the byte ordering of the serial flash data is given. The basic scheme is that the **first** byte read out of the serial flash device - which is addressed by the QSPI_SFAR[SFADR] field - corresponds to bit position QSPI_RBDR0[31:24] register for IP Command read. In contrast to that for AHB Command read the bytes are always positioned according to the byte ordering of the AHB bus.

- **Byte Ordering in Individual Flash Mode**

  The following table gives the byte ordering scheme of how the byte oriented data space of the serial flash device is mapped into one single 32 bit entry of the RX Buffer or the AHB Buffer. The table is valid within the following context:

  - Flash A or Flash B in Individual Flash Mode
  - All AHB data read commands with access size of 32 bit

Table 40-14. Byte Ordering in Individual Flash Mode

Serial Flash Byte Numbering	3	2	1	0
Buffer Entry Bit Position [31:0] (32 Bit data width)	[31:24]	[23:16]	[15:8]	[7:0]

Note

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4, then the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

For AHB Commands, reads, starting from an address not aligned to 32 bit boundaries, the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- **Byte Ordering in Parallel Flash Mode**

  In Parallel Flash Mode each byte is combined out of 2 half bytes which are read in parallel from the two serial flash devices. The following tables shows how the flash content is separated into the half bytes and how the half bytes are assembled to the content of the QSPI_RBDR0 register.

Table 40-15. Serial Flash Device Half Byte Ordering

Serial Flash Device Byte #	Flash A Bit Position		Flash B Bit Position	
	[7:4]	[3:0]	[7:4]	[3:0]
0	fah0	fal0	fbh0	fbl0
1	fah1	fal1	fbh1	fbl1
2	fah2	fal2	fbh2	fbl2
3	fah3	fal3	fbh3	fbl3
4	fah4	fal4	fbh4	fbl4
5	fah5	fal5	fbh5	fbl5
6	fah6	fal6	fbh6	fbl6
7	fah7	fal7	fbh7	fbl7
8	fah8	fal8	fbh8	fbl8

The table entry naming reflects the half byte positioning in the serial flash devices:

- **<fa>**h0 means **Flash A**, <fb>h0 means Flash B.
- fa**<h>**0 means half byte in **high position**, fa<l>o means half byte in low position.
- fah**<0>** means **physical byte address 0** in the serial flash device, fal<1> means physical byte address 1 in the serial flash device.

Table 40-16. Byte Ordering in Parallel Flash Mode - RX Buffer

QSPI_SFAR[SFADR] set to 0x000_0000

Table continues on the next page...

Table 40-16. Byte Ordering in Parallel Flash Mode - RX Buffer (continued)

QSPI_RBDR0 **QSPI_ARDB0**	fal1	fbl1	fah1	fbh1	fal0	fbl0	fah0	fbh0
QSPI_RBDR1 **QSPI_ARDB1**	fal3	fbl3	fah3	fbh3	fal2	fbl2	fah2	fbh2
QSPI_SFAR[SFADR] set to 0x000_0001								
QSPI_RBDR0 **QSPI_ARDB0**	fal2	fbl2	fah2	fbh2	fal1	fbl1	fah1	fbh1
QSPI_RBDR1 **QSPI_ARDB1**	fal4	fbl4	fah4	fbh4	fal3	fbl3	fah3	fbh3

Note

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4 the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

* Appicable only for single io mode.

Table 40-17. Byte Ordering in Parallel Flash Mode - AHB Buffer

AHB Address **(32 Bit Access)**	fal1	fbl1	fah1	fbh1	fal0	fbl0	fah0	fbh0
AHB Address 0x800_0004 **(32 Bit Access)**	fal3	fbl3	fah3	fbh3	fal2	fbl2	fah2	fbh2

Note

For AHB Command read starting from an address not aligned to 32 bit boundaries or AHB access size smaller than 32 bit the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- **Buffer Entry Ordering for 64 Bit Read Access**

For read access via the AHB interface 64 bit access is possible. Each 64 bit access reads 2 32 bit entries simultaneously. The ordering of these 32 bit entries within the 64 bit word is given in the following table.

Table 40-18. 64 Bit Read Access Buffer Entry Ordering

AHB Read Data Bit Position [63:0]	[63:32]	[31:0]
Buffer Entry #	Odd (1, 3, 5, ...)	Even (0, 2, 4, ...)

40.5.3.9 Normal Mode Interrupt and DMA Requests

The QuadSPI module has different flags that can only generate interrupt requests and one flag that can generate interrupt as well as DMA requests. The following table lists the eight conditions. Note that the flags mentioned in the table are related to the Flag Register (QSPI_FR).

Table 40-19. Interrupt and DMA Request Conditions

Condition	Flag(QSPI_FR)	DMA
Data Learn patttern Failure	DLPFF	-
TX Buffer Fill	TBFF	-
TX Buffer Underrun	TBUF	-
Illegal Instruction Error	ILLINE	-
RX Buffer Drain	RBDF	X
RX Buffer Overflow	RBOF	-
AHB Buffer Overflow	ABOF	-
AHB Sequence Error	ABSEF	-
IP Command Usage Error	IUEF	-
IP Command Trigger during AHB Access Error	IPAEF	-
IP Command Trigger could not be executed Error	IPIEF	-
IP Access during AHB Grant Error	IPGEF	-
IP Command related Transaction Finished	TFF	-

Each condition has a flag bit in the Flag Register (QSPI_FR) and a Request Enable bit in the DMA Request Select and Enable Register (QSPI_RSER) . The RX Buffer Drain Flag (RBDF) has separate enable bits for generating IRQ and DMA requests. Note that not all flags have an individual IRQ line. Check the devices Interrupt Vector Table for more details.

- Transmit Buffer Fill Interrupt Request:

  The Transmit Buffer Fill IRQ indicates that the TX Buffer can accept new data. It is asserted if the QSPI_FR[TBFF] flag is asserted and if the corresponding enable bit (QSIP_RSER[TBFIE]) is set. Refer to TX Buffer Operation, for details about the assertion of the QSPI_FR[TBFF] flag.

- Receive Buffer Drain Interrupt or DMA Request:

  The Receive Buffer Drain IRQ derived from the QSPI_FR[RBDF] flag indicates that the RX Buffer of the QuadSPI module has data available from the serial flash device to be read by the host. It remains set as long as the QSPI_RBSR[RXWE] bit is set. The QSPI_RSER[RBDIE] bit enables the related IRQ.

  Aside from the IRQ it is possible to handle RX Buffer drain by DMA. If the QSPI_RSER[RBDDE] bit is set, a DMA request will be triggered when the RX Buffer contains more than QSPI_RBCT[WMRK] valid entries.The application must set the environment appropriately (for example, the DMA controller) for the DMA transfers.

- Buffer Overflow/Underrun Interrupt Request:

  The Buffer Overflow/Underrun IRQ is a combination of the following flags (all located in the QSPI_FR register with the related enable bits in the QSPI_RSER register):

  - TBUF - TX Buffer Underrun, enabled by TBUIE
  - RBOF - RX Buffer Overflow, enabled by RBOIE
  - ABOF - AHB Buffer Overflow, enabled by ABOIE

    The Transmit Buffer Underrun indicates that an underrun condition in the TX Buffer has occurred. It is generated when a write instruction is triggered whilst the Tx Buffer is empty and the QSPI_RSER[TFUFIE] bit is set.

    The Receive Buffer Overflow indicates that an overflow condition in the RX Buffer has occurred. It is generated when the RX Buffer is full, an additional read transfer attempts to write into the RX Buffer and the QSPI_RSER[RBOIE] bit is set.

    The AHB Buffer Overflow indicates that an overflow condition in the AHB Buffer has occurred. It is generated when the AHB Buffer is full, an additional read transfer attempts to write into the AHB Buffer and the QSPI_RSER[ABOIE] bit is set.

    The data from the transfers that generated the individual overflow conditions is ignored.

- Serial Flash Command Error Interrupt Request

  If the IPAEF, IPIEF, IPGEF or IUEF flags in the QSPI_FR are set, and the related interrupt enable bits in the QSPI_RSER are also set, then an interrupt is requested.

- Transaction Finished Interrupt Request

  The IP Command Transaction Finished IRQ indicates the completion of the current IP Command.It is triggered by the QSPI_FR[TFF] flag and is masked by the QSPI_RSER[TFIE] bit.

40.5.3.10 TX Buffer Operation

The TX Buffer provides the data used for page programming. For proper operation it is required to provide at least four entry in the TX Buffer prior to starting the execution of the page programming command. The application must ensure that the required number of data bytes is written into the TX Buffer fast enough as long as the command is executed without a TX Buffer overflow or underrun.

The QuadSPI module sets the QSPI_FR[TBFF] flag so long as the TX Buffer is not full and can accept more data.

When the QuadSPI module tries to pull data out of an empty TX Buffer the TX Buffer underrun is signaled by the QSPI_FR[TBUF] flag.The TX buffer underrun flag is also asserted when TX buffer contains less than 128 bits of data and QuadSPI module tries to pull out data from it. The current IP Command leading to the underrun condition is continued until the specified number of bytes has been sent to the serial flash device, in the underrun condition when QuadSPI module tries to pull out data of empty TX buffer, the data transferred is all F's i.e. once the underrun flag is set under this condition, it will return F's until the required number of bytes are not sent. This has been done to ensure that the software need not to erase whole sector after underrun, just reprogramming from failure point will serve the purpose. When this Sequence Command is finished, the QSPI_FR[TBFF] flag is asserted indicating that the Tx Buffer is ready to be written again.

The TX Buffer overflow isn't signaled explicitly, but the TX Buffer fill level can be monitored by the QSPI_TBSR[TRBFL] field.

Refer to TX Buffer Status Register (QuadSPI_TBSR) and Flag Register (QuadSPI_FR) for details about the TX Buffer related registers.

40.5.3.11 Address scheme

Earlier serial flash memories supported only 24-bit address space hence restricting the maximum memory size of the serial flash as 16 MB. The new memory specification supports two types of 32-bit addressing mode in addition to legacy 24-bit address mode.

- **Extended Address Mode**

  In this mode, the legacy 24-bit commands are converted to accept 32-bit address commands. The flash memory needs to be configured for 32-bit address mode. Also, while programming the LUT sequence in QuadSPI for 32-bit mode, the ADDR and ADDR_DDR command should be programmed with 8'd32 as the operand value. By default, the QuadSPI is in 24-bit legacy address mode. Each of the memory vendors have a different way of enabling this mode (Refer to the memory specification from memory vendors). For example, the command B7h sent to Macronix flash will enable it for 32-bit address mode.

- **Extended Address register**

  In this mode, the upper 8-bit of the 32-bit address is provided by the Extended address register in the memory itself. The memory provides a specific register which is updated according to the address to be accessed. This effectively converts the legacy 24-bit address command into 32-bit address commands. The memories greater in size than 16 MB, consists of banks of 16 MB. The 8-bit written in the extended address register effectively enables a bank. For example in Spansion memory, when the extended address register is updated with a value of 0x01 with the help of the command 17h, it will open Bank1 of the memory. The consequent 24-bit address commands will lead to Bank1. The extended address register needs to be update with the respective value for access to other banks. This effectively converts the legacy 24-bit address command into 32-bit address commands.

40.6 Initialization/Application Information

This section provides the initialization and application information of the QuadSPI module.

40.6.1 Power Up and Reset

Note that the serial flash devices connected to the QuadSPI module may require special voltage characteristics of their inputs during power up or reset. It is the responsibility of the application to ensure this.

40.6.2 Available Status/Flag Information

This paragraph gives an overview of the different status and flag information available and their interdependencies for different use cases. Related registers are QSPI_SR and QSPI_FR. Refer to the related descriptions how to set up the QuadSPI module appropriately.

40.6.2.1 IP Commands

Refer to IP Configuration Register (QuadSPI_IPCR) for additional details not explicitly covered in this paragraph.

- **IP Commands - Normal Operation**

  Writing the QSPI_IPCR[SEQID] field triggers the execution of a new IP Command. Given that this is a legal command the QSPI_SR[IPACC] and the QSPI_SR[BUSY] bits are asserted simultaneously, immediately after the execution is started.

  When the instruction on the serial flash device has been finished these bits are de-asserted and the QSPI_FR[TFF] flag is set.

- **IP Commands - Error Situations**

  Refer to Table 40-20 below.

40.6.2.2 AHB Commands

Refer to Section 1, Reading Serial Flash Data into the QuadSPI Module, in Flash Read for additional details not explicitly covered in this paragraph.

- **AHB Commands - Normal Operation**

  Memory mapped read access to a serial flash address not contained in the AHB Buffer, triggers the execution of an AHB Command. Given that this is a legal command the QSPI_SR[AHBACC] and the QSPI_SR[BUSY] bits are asserted simultaneously immediately after the execution is started. When the instruction on the serial flash device has been finished these bits are de-asserted.

- **IP Commands - Error Situations**

  Refer to Table 40-20 below.

40.6.2.3 Overview of Error Flags

The following table gives an overview of the different error flags in the QSPI_FR register and additional error-related details.

Table 40-20. Overview of QSPI_FR Error Flags

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
AHB Error Flag	ABSEF	Flash transaction is aborted	AHB sequence contains • WRITE instruction • WRITE_DDR instruction
AHB Error Flag	ABOF	Flash transaction continues until it finishes	Set when the module tried to push data into the AHB buffer that exceeded the size of the AHB buffer. Only occurs due to wrong programming of the QSPI_BUFxCR[ADATSZ].
Miscellaneous Error Flag	DLPFF[1]	Flash transaction continues until it finishes	Set when DATA_LEARN instruction was encountered in a sequence but no sampling point was found for the data learning pattern.
Miscellaneous Error Flag	ILLINE	Flash transaction aborted	Illegal instruction Error Flag – Set when an illegal instruction is encountered by the controller in any of the sequences.
Command Arbitration Error	IPIEF	TFF not asserted in conjunction with that command	IP Command Error - caused when IP access is currently in progress (IP_ACC set) and • write attempt to QSPI_IPCR register. • write attempt to QSPI_SFAR register. • write attempt to QSPI_RBCT register.
Command Arbitration Error	IPAEF		• AHB Command already running, another IP Command could not be executed. • AHB Command already running, write attempt to QSPI_IPCR[SEQID] field.
Command Arbitration Error	IPGEF		• Exclusive access to the serial flash granted for AHB Commands, write attempt to QSPI_IPCR[SEQID] field.
IP Command Error	IUEF	—	• IP Command Usage Error

Table continues on the next page...

Table 40-20. Overview of QSPI_FR Error Flags (continued)

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
Buffer Related Error	RBOF	TFF is asserted on completion	• RX Buffer Overrun
Buffer Related Error	TBUF		• TX Buffer Underrun

1. Data learning is not implemented on this chip.

Note that only the buffer related errors are related to a transaction on the external serial flash. All the other errors do not trigger an actual transaction.

40.6.2.4 IP Bus and AHB Access Command Collisions

There are two flags related to this topic, the QSPI_FR[IPAEF] and QSPI_FR[IPIEF]. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for a description of the flags and Command Arbitration , for details about possible command collisions.

40.6.3 Exclusive Access to Serial Flash for AHB Commands

It is possible that several masters need to access the serial flash device connected to the QuadSPI module separately, one master by triggering IP Commands and reading the RX Buffer (via RBDR*n* register) and the other masters by triggering AHB Commands (via ARDB*n* Registers). These two set of buffer (RBDR and ARDB Buffer) points to the same physical buffer.Refer to Figure 40-5 To avoid command collisions resulting in excessive latencies the QuadSPI module implements a request-handshake mechanism between the master triggering AHB Commands and the QuadSPI module allowing this specific master to request exclusive access to the serial flash device for AHB Commands. If this exclusive access is granted the execution of IP Commands is blocked. This resolves command collisions and excessive times where the AHB interface may be blocked.

If this capability is used in the device there is additional status and flag information available related to this mechanism. The QSPI_SR[AHBGNT] bit reflects the module-internal state that the exclusive access mentioned above is granted, any attempt to trigger an IP Command is rejected and results in the assertion of the QSPI_FR[IPGEF] flag. Refer to the descriptions of the related bit and flag for details.

It is within the responsibility of the application to set up the master using this mechanism appropriately, if used incorrectly no IP Commands at all can be triggered.

Two different cases can be distinguished:

40.6.3.1 RX Buffer Read via QSPI_ARDB Registers

In this case all masters share the AHB bus for RX Buffer as well as for AHB Buffer read. In this case the access to the AHB interface by the master triggering AHB Commands must be deferred until any pending IP Command has been finished **and** the RX Buffer readout has been finished as well. The QSPI ARDB Buffers access the Rx buffer i.e the data from the Rx Buffer is returned and no data from AHB Buffer is touched. This is the conservative use case, corresponding to the reset value 0 of the QSPI_RBCT[RXBRD] bit.

In this case the QSPI_SR[AHBGNT] bit is asserted not earlier than any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0 and Rx Buffer readout is via AHB(QSPI_RBCT[RXBRD]) equal to 1.

40.6.3.2 RX Buffer Read via QSPI_RBDR Registers

This is the preferred use case as an access to the AHB buffer (memory mapped flash) does not interfere with any IPS access to read the RBDR buffer. It is not possible that a pending AHB bus access triggered by an AHB Command stalls the AHB bus and blocks the RX Buffer readout since the RX Buffer is read via the IP bus based registers QSPI_RBDR0 to QSPI_RBDR31.

For this case it is recommended to program the QSPI_RBCT[RXBRD] bit to 1. The QSPI_SR[AHBGNT] bit is asserted immediately after any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0, allowing the master triggering AHB Commands to trigger AHB Commands as soon as possible without the need to wait for the RX Buffer readout to be finished.

40.6.4 Command Arbitration

In case of overlapping commands, the arbitration scheme is described in the following paragraphs under the assumption that the priority mechanism described in Exclusive Access to Serial Flash for AHB Commands is **not** used:

- During the execution of an IP Command, the running IP Command can't be terminated by issuing another IP Command or AHB Command. The QSPI_FR[IPIEF] flag is asserted when the host tries to write into the QSPI_IPCR register. When the host triggers an AHB Command (refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for details), this command is stalled until the currently running IP Command is finished.
- During the execution of an AHB Command, the running AHB Command can't be terminated by issuing an IP Command. The command is ignored and the QSPI_FR[IPAEF] flag is asserted. Refer to Flag Register (QuadSPI_FR) for the description of these flags.

  When another AHB Command is triggered the address of the memory mapped access is considered. If the requested address is currently read from the serial flash device, the running command is continued. If this is not the case the currently running command is terminated and another AHB Command related to the requested address is executed. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for further details.

In case of coinciding commands the IP Command is triggered and the AHB Command is stalled until the IP Command has been finished (QSPI_SR[IP_ACC] has been deasserted).

The IP Commands ignored in case of command collision will not result in the assertion of the QSPI_FR[TFF] flag.

40.6.5 Flash Device Selection

Regardless of the SFM Command (IP or AHB) the access mode is selected by specifying the 32 bit address value for the following SFM Command.

For IP Commands the access mode is selected with the address programmed into the QSPI_SFAR register. Refer to Serial Flash Address Register (QuadSPI_SFAR) for details.

For AHB Commands the access mode is determined by the memory mapped address which is accessed Refer to AMBA Bus Register Memory Map for details.

40.6.6 DMA Usage

For the complete description of the DMA module refer to the related DMA Controller chapter. In this paragraph only the details specific to the DMA usage related to the QuadSPI module are given.

40.6.6.1 DMA Usage in Normal Mode

40.6.6.1.1 Bandwidth considerations

Careful consideration of the throughput rate of the entire chain (serial flash -> AHB bus / IP Bus -> DMA controller) involved in the read data process is essential for proper operation. Such analysis must take into account not only the data rate provided by the serial flash but also the data rate of the AHB bus and the performance of the DMA controller in reading data from the RX buffer.

Two figures must match for proper operation, that means that the data rate provided by the serial flash device must not exceed the average RX Buffer readout data rate. Otherwise, the longer this state persists, a RX Buffer overflow will result.

AHB Bus Side (data read):

The total number of bus cycles for each DMA Minor Loop completion is added from the following components:

- Overhead for each minor loop, given by DMA controller: Assume 10 cycles
- Overhead due to clock domain crossing: Assume 2 cycles
- Number of bus clock cycles required for 8 bytes (64 bit read size): Assume 2 cycles (read/write sequence of DMA controller)

Note that the size of the minor loop is determined by the size of the QSPI_RBCT[WMRK] field, therefore the overhead given above distributes among (QSPI_RBCT[WMRK]+1)/2 read accesses of 64 bit each.

The following table gives some examples for typical use cases:

Table 40-21. Access Duration Examples - Bus Clock Side

QSPI_RBCT[WMRK]	Number of Bytes per DMA Loop [1]	Number of Bus Clock cycles for DMA Minor Loop	Time Duration of DMA Minor Loop for 120Mhz Bus clock Frequency
0	4	12+2 = 14	~117ns

Table continues on the next page...

Table 40-21. Access Duration Examples - Bus Clock Side (continued)

QSPI_RBCT[WMRK]	Number of Bytes per DMA Loop [1]	Number of Bus Clock cycles for DMA Minor Loop	Time Duration of DMA Minor Loop for 120Mhz Bus clock Frequency
1	8	12+2 = 14	~117ns
3	16	12+4 = 16	~133ns
7	32	12+8 = 20	~167ns
11	48	12+12 = 24	~200ns

1. DMA Loop means one Minor Loop Completion which is equivalent to one.

NOTE

The table figure represents ideal scenario, actual performance will depend on how the system is integrated.

Serial Flash Device Side (data read):

The number of serial flash cycles can be determined in the following way:

- Number of serial flash clock cycles required to read 4 bytes, corresponding to one RX Buffer entry (setup of command and address not considered): 2 cycles for Quad DDR mode instructions in Parallel Flash Mode, 4 cycles for Quad (SDR) mode instruction in parallel flash mode or Dual IO DDR mode instruction in parallel flash mode, 8 cycles for Quad Mode (SDR) instructions in Individual Flash Mode etc.
- Overhead due to clock domain crossing : 1 cycle.

The following table lists the number of clock cycles required to read the data from the serial flash corresponding to the different settings of the QSPI_RBCT[WMRK] field:

Table 40-22. Access Duration Examples - Serial Flash side

QSPI_RBCT[WMRK] setting	Num Bytes per DMA Loop [1]	Num SCKFx for 60MHz SCKFx			Time duration of Flash data readout for 60MHz SCKFx (~16.6ns period)		
		IFM [2] Quad	IFM Quad DDR	PFM [3] Quad DDR	IFM Quad	IFM Quad DDR	PFM Quad DDR
0	4	9	5	3	~150ns	~83ns	~50ns
1	8	17	9	5	~282ns	~150ns	~83ns
3	16	33	17	9	~548ns	~282ns	~150ns
7	32	65	33	17	~1079ns	~548ns	~282ns
11	48	97	49	25	~1610ns	~813ns	~415ns

1. DMA Loop means one Minor loop completion which is equivalent to one Major Loop iteration.
2. Individual flash mode.
3. Parallel flash mode.

From the examples given in the two tables above, it can be seen that depending on the relationship between the Bus clock and Serial flash clock frequencies, there are settings possible where the serial flash provides the read data faster than the AHB bus can read out the RX buffer. In the above tables, it is the case of PFM Quad DDR mode with Watermark up to 3 and other cases. In these cases, the RX buffer data keeps accumulating over time and will eventually overflow. To avoid RX Buffer overflow, the data transaction size should be small enough.

A complementary example would be when the watermark is set to be too high. In such a case, the time taken by the DMA to read out the RX buffer entries should be smaller than the time taken by the controller to push in the remaining entries in the buffer.

NOTE

The tables mentioned above are only examples which must be correlated with the DMA in the system.

40.6.7 Parallel mode

QuadSPI can access two flashes in parallel. This increases the throughput of the QuadSPI by two times. Only read operations are allowed in parallel mode. In case a write transaction is initiated in parallel mode, QSPI_FR[IUEF] is set. When dual die flashes are accessed in parallel mode, it is mandatory for flash A1 to be of the same size as B1 and A2 to be of the same size as B2. The following figure shows how QuadSPI maps the incoming addresses to the different flashes connected on board.

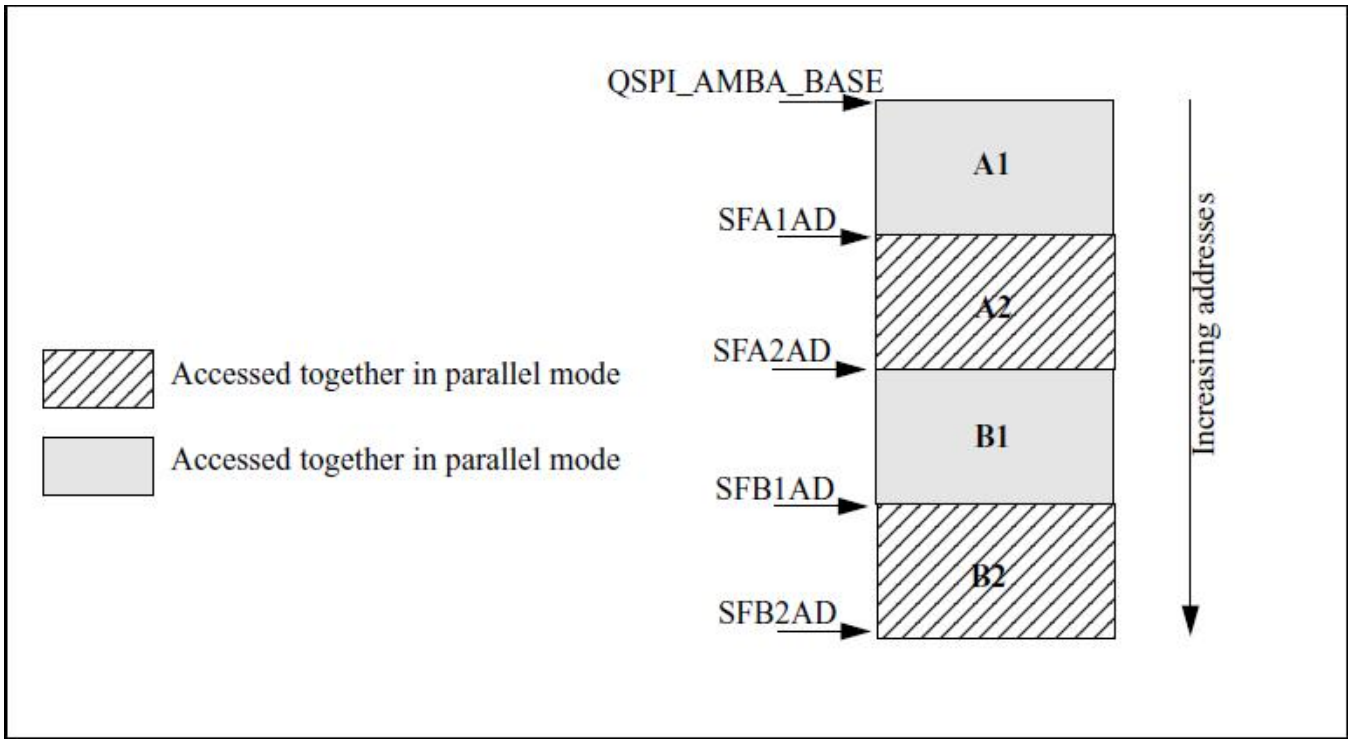


Figure 40-6. Flash addressing

An example programming for parallel mode access is given below (flash sizes are assumed to be 256MB):

- QSPI_AMBA_BASE - 0x10000000
- QSPI_SFA1AD[TPADA1] - 0x20000000
- QSPI_SFA2AD[TPADA2] - 0x30000000
- QSPI_SFB1AD[TPADB1] - 0x40000000
- QSPI_SFB2AD[TPADB2] - 0x50000000

In order to access the first location of A1/B1 pair, the incoming address should be 0x10000000. QSPI_AMBA_BASE is subtracted from this address and the result is divided by two. Therefore, address provided to flash A1 and B1

Flash Address = (Memory mapped address - QSPI_AMBA_BASE)/2

For Memory Mapped address:

- 0x10000000, flash address: 0x0 (Or, the first address of flash A1 and B1)
- 0x10000004, flash address: 0x2
- 0x10000008, flash address: 0x4 etc.

Similarly, in order to access the first location of A2/B2 pair, the incoming address should be 0x30000000.

Flash Address = (Memory mapped address - SFA2AD)/2

For Memory Mapped address:

- 0x30000000, flash address: 0x0 (Or, the first address of flash A2 and B2)
- 0x30000004, flash address: 0x2
- 0x30000008, flash address: 0x4 etc.

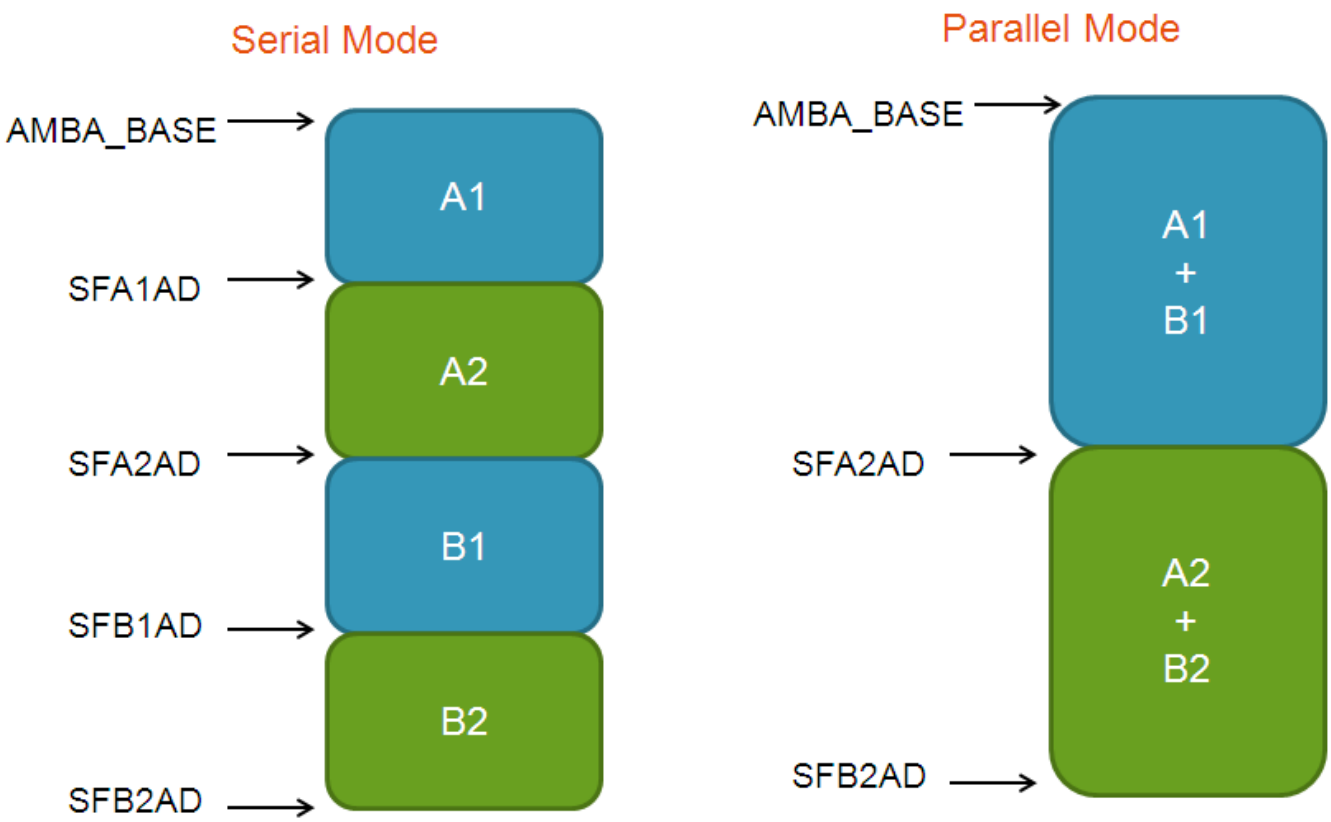


Figure 40-7. Memory map - Serial and Parallel

40.7 Byte Ordering - Endianness

QuadSPI provides support for swapping the flash read/write data based on the configuration of the QSPI_MCR[END_CFG]. By default the data is always returned in 64 bit LE format on the AHB bus and 32 bit LE format on the IPS interface when read via the RX buffer and written in 32 bit LE format when written via the TX buffer.

The table(QSPI_MCR[END_CFG]) below shows the complete bit ordering. BE signifies Big Endian which means the high order bits of the associated data vectors are associated with low order address positions. LE signifies Little Endian which means the lower order bits of the associated data vectors are associated with low order address positions.Refer to figure Figure 40-5

Table 40-23. QSPI_MCR[END_CFG]

00	64 bit BE
01	32 bit LE
10	32 bit BE
11	64 bit LE

The tables below (Byte ordering configuration in AHB) and (Byte ordering configuration in IPS) show how this configuration is implemented in QSPI AHB and IPS interfaces respectively. B in the table signifies Byte and the index 1-8 refers to the byte position i.e. 1 refer to bits[7:0], 8 refer to bits[63:56] and so on.

Table 40-24. Byte ordering configuration in AHB

64 bit BE	B1	B2	B3	B4	B5	B6	B7	B8
64 bit LE	B8	B7	B6	B5	B4	B3	B2	B1
32 bit BE	B5	B6	B7	B8	B1	B2	B3	B4
32 bit LE	B4	B3	B2	B1	B8	B7	B6	B5

Table 40-25. Byte ordering configuration in IPS

32BE	B1	B2	B3	B4
32LE	B4	B3	B2	B1

The examples below show the byte ordering in 64 bit BE configuration for AHB Buffer and 32 bit BE for TX/RX Buffer:

40.7.1 Programming Flash Data

CPU write instructions to the QSPI_TBDR register like

- Write QSPI_TBDR -> 0x01_02_03_04
- Write QSPI_TBDR -> 0x05_06_07_08

result in the following content of the TX Buffer:

Table 40-26. Example of QuadSPI TX Buffer

TX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

Programming the TX Buffer into the external serial flash device results in the following byte order to be sent to the serial flash:

- 01…02…03…04…05…06…07…08

40.7.2 Reading Flash Data into the RX Buffer

Reading the content from the same address provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the RX Buffer filled with:

Table 40-27. Resulting RX Buffer Content

RX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

40.7.2.1 Readout of the RX Buffer via QSPI_RBDRn

The RX Buffer content appears at CPU read access via the Peripheral bus interface in the following order:

- Read QSPI_RBDR0 <- 0x01_02_03_04
- Read QSPI_RBDR1 <- 0x05_06_07_08

40.7.2.2 Readout of the RX Buffer via ARDBn

The RX Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04
- (2a): 32 Bit Access: Read QSPI_ARDB1 <- 0x05_06_07_08
- (1b/2b): 64 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04_05_06_07_08

40.7.3 Reading Flash Data into the AHB Buffer

Reading the content from the same address as it was written to provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the AHB Buffer filled with:

Table 40-28. Resulting AHB Buffer Content

AHB Buffer Entry	Content
0	64'h01_02_03_04_05_06_07_08

40.7.3.1 Readout of the AHB Buffer via Memory Mapped Read

The AHB Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Read Access: <- 0x01_02_03_04
- (2a): 32 Bit Read Access: <- 0x05_06_07_08
- (1/2): 64 Bit Read Access: <- 0x01_02_03_04_05_06_07_08

40.8 Serial Flash Devices

Several different vendors make flash devices with a QuadSPI interface. At present there is no set standard for the QuadSPI instruction set. Most common commands currently have the same instruction code for all vendors, however some commands are unique to specific vendors. Some example sequences are provided below.

40.8.1 Example Sequences

This section provides the example sequences of the QuadSPI module.

Table 40-29. Exit 4 x I/O Read Enhance Performance Mode (XIP) (Macronix) and Read Status

INSTR	PAD	OPERAND	COMMENT
CMD	0x0	0xEB	4xIO Read Command
ADDR	0x2	0x18	24 Bit address to be send on 4 pads
MODE	0x2	0x00	2 mode cycles (exit XIP)
DUMMY	0x0	0x04	4 dummy cycles
READ	0x2	0x08	Read 64 bits
CMD	0x0	0x05	Read Status register
READ	0x0	0x01	Status register data
STOP	0x0	0x00	STOP, Instruction over

40.8.1.1 Fast Read Sequence (Macronix/Numonyx/Spansion/ Winbond)

The following table shows the fast read sequence for Macronix/Numonyx/Spansion/ Winbond flashes.

Table 40-30. Fast Read sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x0B	Fast Read command = 0x0B
ADDR	0x0	0x18	24 Addr bits to be sent on one pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x0	0x04	Read 32 Bits on one pad
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

40.8.1.2 Fast Dual I/O DT Read Sequence (Macronix)

The following table shows the Fast Dual I/O DT read sequence for Macronix flashes.

Table 40-31. Fast Dual I/O DT Read sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xBD	Fast Dual I/O DT read command = 0xBD

Table continues on the next page...

Table 40-31. Fast Dual I/O DT Read sequence (continued)

Instruction	Pad	Operand	Comment
ADDR_DDR	0x1	0x18	24 Addr bits to be sent on 2 pads in DDR mode
MODE4_DDR	0x1	0x00	P2=P0 or P3=P1 is necessary. Refer to Macronix datasheet for details. One clock cycle for mode.
DUMMY	0x0	0x06	6 Dummy cycles
READ_DDR	0x1	0x04	Read 32 Bits on 2 pads in DDR mode
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

40.8.1.3 Fast Read Quad Output (Winbond)

The following table shows the Fast read quad output sequence for Winbond memories

Table 40-32. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x6B	Fast read quad output command = 0x6B
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

40.8.1.4 4 x I/O Read Enhance Performance Mode (XIP) (Macronix)

The following table shows the 4 x I/O Read Enhance Performance Mode for Macronix flashes. The enhanced performance mode is also known as XIP mode.

Table 40-33. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xEB	4xI/O Read command = 0xEB
ADDR	0x2	0x18	24 Addr bits to be sent on 4 pads
MODE	0x2	0xA5	2 mode cycles
DUMMY	0x0	0x04	4 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x01	Jump to instruction 1 (ADDR)

When in XIP mode the software should ensure that all the flashes connected to the controller are in XIP mode. As a part of initializing the controller, all the flashes may be enabled with XIP by carrying out dummy reads.

40.8.1.5 Dual Command Page Program (Numonyx)

The following table shows the Dual command page program sequence for Numonyx flashes.

Table 40-34. Dual Command Page Program sequence

Instruction	Pad	Operand	Comment
CMD	0x1	0x02	Dual command page program = 0x02 on 2 pads
ADDR	0x1	0x18	24 Addr bits to be sent on 2 pads
WRITE	0x1	0x20	Write 32 Bytes on 2 pads
STOP	0x0	0x00	STOP, Instruction over

40.8.1.6 Sector Erase (Macronix/Spansion/Numonyx)

The following table shows the Sector erase sequence for Macronix/Spansion/Numonyx flashes

Table 40-35. Sector Erase sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xD8	Sector erase command = 0xD8
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
STOP	0x0	0x00	STOP, Instruction over

40.8.1.7 Read Status Register (Macronix/Spansion/Numonyx/ Winbond)

The following table shows the Read status register sequence for Macronix/Spansion/ Numonyx/Winbond flashes.

Table 40-36. Read Status Register Sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x05	Read status register command = 0x05
READ	0x0	0x01	Read status register data
STOP	0x0	0x00	STOP, Instruction over

40.8.2 Dual Die Flashes

Certain serial flash vendors provide dual-die packages which are essentially two devices (dies) stacked within the same package to increase the memory capacity of a single package. These two devices within a package share the same data and clock pins, but have individual Chip Selects. QuadSPI controller provides support for two dual-die packages to be connected simultaneously. The figure below shows the two dual-die packages and the naming conventions used in this document. For simplicity, the data pins are shown to be unidirectional.

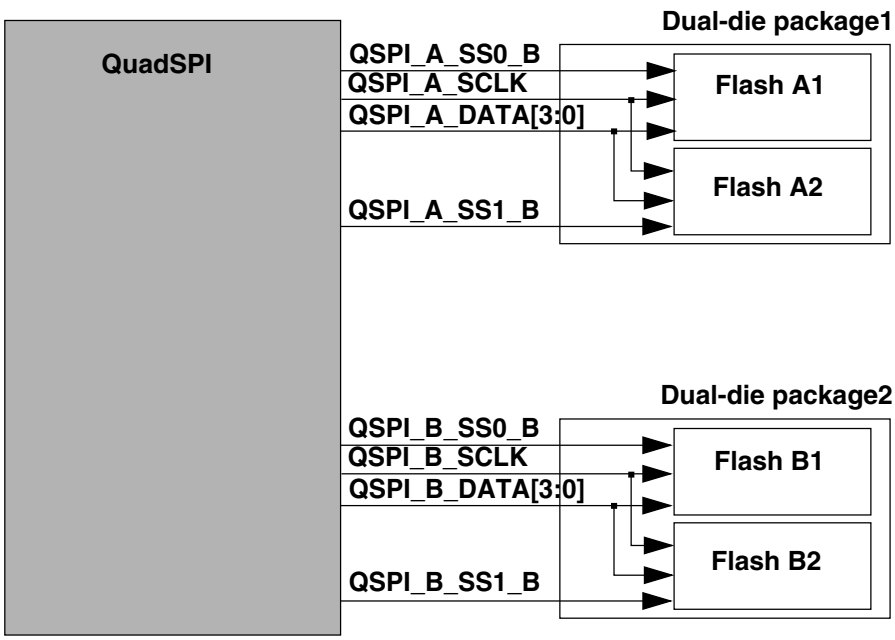


Figure 40-8. Dual-die support

Since the two devices within one package share the same i/o pads, they cannot function in parallel mode. Software should ensure that when QuadSPI is configured in parallel mode the two selected flash devices are from different dual-die packages.

40.8.3 Boot initialization sequence

The following are the recommended sequence of steps for booting from QuadSPI:

- System out of reset and flash available (300us)

- Clocks still at very low frequency. Clock tree configured, I/O pins configured. First request sent to QuadSPI for address 0x0 of flash.
- The reset command sequence in QuadSPI has 0x03 (basic read command) which is applicable to all flashes at < 50MHz serial flash clock
- The first few bytes of data is read from the flash which contains the following information:
  - The total sizes of all the flashes connected on board
  - Whether DDR mode supported
  - Frequency of DDR operation
  - Continuous mode entry sequence
  - 24bit or 32bit addressing (assuming 24bit for first accesses)
- All the serial flashes are configured
  - Quad Mode enabled
  - Dummy reads to enter into XIP
- QuadSPI is configured
  - Parallel enable set
  - LUT configured for highest performance reads
  - DDR mode enabled (if applicable)
  - Buffers configured
- Serial flash clock frequency increased.
- Boot reads happen in parallel, DDR enabled, quad output mode @66MHz.

40.9 Sampling of Serial Flash Input Data

40.9.1 Internal Sampling of Serial Flash Input Data

Depending from the actual implementation there is a delay between the internal clocking in the QuadSPI module and the external serial flash device.Refer to the following figure for an overview of this scheme.

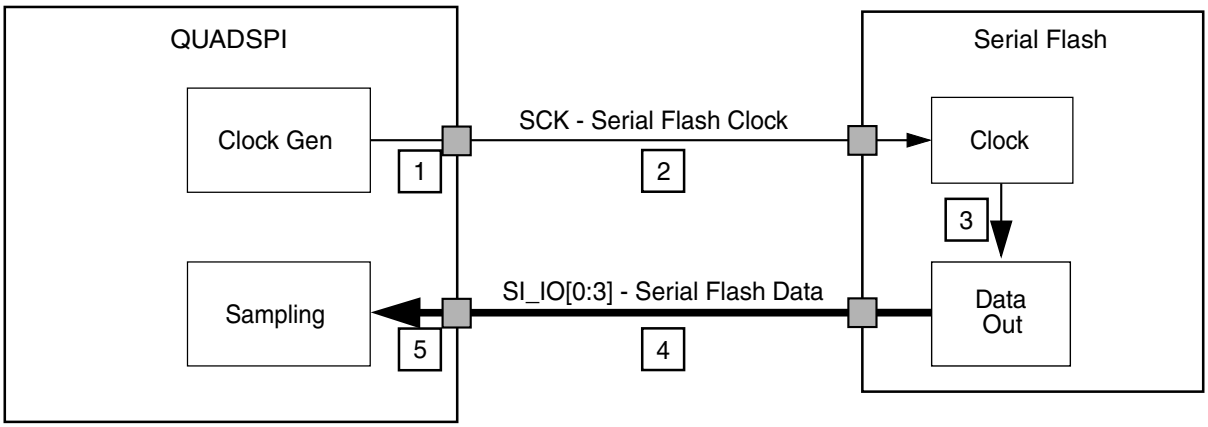


Figure 40-9. Serial Flash Sampling Clock Overview

Note

The arrival of the serial flash data in the sampling stage of the QuadSPI module are given in the following figure. Note that the amount of the total delay $t_{Del,total}$ is very specific to the characteristics of the actual implementation.

Note also that the serial flash device clock SCK is inverted with respect to the QuadSPI internal reference clock.

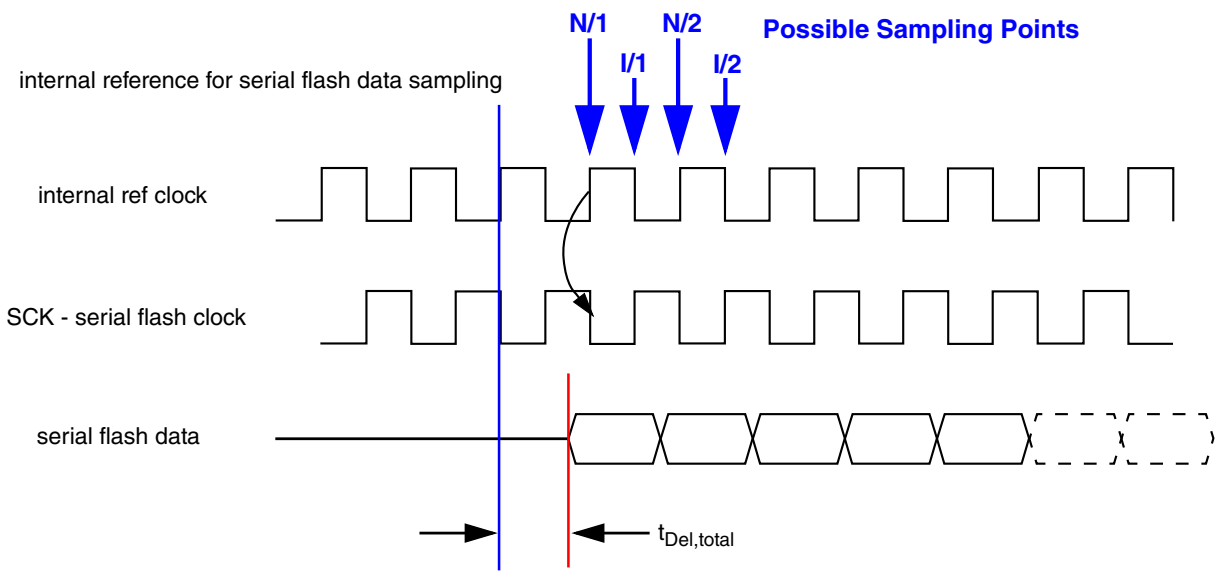


Figure 40-10. Serial Flash Sampling Clock Timing

The rising edge of the internal reference clock is taken as timing reference for the data output of the serial flash. After a time of $t_{Del,total}$ the data arrive at the internal sampling stage of the QuadSPI module.

According to the Serial Flash Sampling Clock Overview figure, the following parts of the delay chain contribute to $t_{Del,total}$:

1. Output delay of the serial flash clock output of the device containing the QuadSPI module
2. Wire delay of application/PCB from the device containing the QuadSPI module to the external serial flash device
3. Clock to data out delay of the external serial flash device, including input and output delays
4. Wire delay of application/PCB from the external serial flash device to the device containing the QuadSPI module
5. Input delay belonging to the data in input

The possible points in time for the sampling of the incoming data are denoted as N/1, I/1, N/2 and I/2 above. The sampling point relevant for the internal sampling is configured in the QSPI_SMPR register, refer to Sampling Register (QuadSPI_SMPR) for details. Note that the falling edges of the reference clock are not actually used, instead the inverted clock is used for sampling at these positions. The following table gives an overview of the available configurations for the commands running at regular (full) speed:

Table 40-37. Sampling Configuration

Sampling Point	Description	Delay [FSDLY] [HSDLY]	Phase [FSPHS] [HSPHS]	QSPI_SMPR for Full Speed Setting[1]
N/1	sampling with non-inverted clock, 1 sample delay	0	0	0x0000000x
I/1	sampling with inverted clock, 1 sample delay	0	1	0x0000002x
N/2	sampling with non-inverted clock, 2 samples delay	1	0	0x0000004x
I/2	sampling with inverted clock, 2 samples delay	1	1	0x0000006x

1. 'x' is not considered here

Depending from the actual delay and the serial flash clock frequency the appropriate sampling point can be chosen. The following remarks should be considered when selecting the appropriate setting:

- Theoretically there should be 2 settings possible to capture the correct data since the serial flash output is valid for 1 clock cycle, disregarding rise and fall times and timing uncertainties.
- Depending from the timing uncertainties it may turn out in actual applications that only one possible sample positions remains. This is subject to careful consideration depending from the actual implementation.

- The delay $t_{Del,total}$ is an absolute size to shift the point in time when the serial flash date get valid at the QuadSPI input.
- For decreasing frequency of the serial flash clock the distance between the edges increases. So for large differences in the frequency the required setting may change.
- For commands running at half of the regular serial flash clock (QSPI_SMPR[HSENA] bit set) the sampling point must be figured separately to allow for the compensation of the absolute shift in time with respect to the sample-relative setting in the QSPI_SPMPR register.

40.9.2 DDR Mode

The increasing requirement of improved throughput has introduced the double data rate (DDR) mode. In DDR mode, the data is transferred on both the rising and falling edges of the serial flash clock. The DDR serial flashes sample as well as drive the data on both rising and falling edges of serial flash clock.

40.10 Serial Flash Data Input Timing

There are sampling modes for input flash data:

- Internal sampling - Input serial flash data is captured by internal serial clock. In SDR mode, serial data is sampled by serial clock 1x (ser_clk_1x) rise edge. In DDR mode, serial data is sampled by serial clock 4x (ser_clk_4x) rise edge.
- Loopback DQS sampling - Soc will output serial data strobe with internal serial clock. This serial data strobe would be loopback from pad and used to sample input serial data. In SDR mode, serial data is sampled by loopback DQS rise edge. In DDR mode, serial data is sampled by loopback DQS both edge

NOTE

DQS pad need to be set to force input for loopback.

- Flash DQS sampling - Some serial Flash device provide the data strobe output together with serial data. This strobe signal is used to sample input serial data directly. In SDR mode, serial data is sampled by Flash DQS rise edge. In DDR mode, serial data is sampled by Flash DQS both edge

Following table shows selection for sampling mode:

Sampling mode	DQS_EN	DQS_LOOPBACK_EN
Internal sampling	0	doesn't matter

Table continues on the next page...

Sampling mode	DQS_EN	DQS_LOOPBACK_EN
Loopback DQS sampling	1	1
Flash DQS sampling	1	0

Serial flash data and clock path for input timing is shown in the following figure.

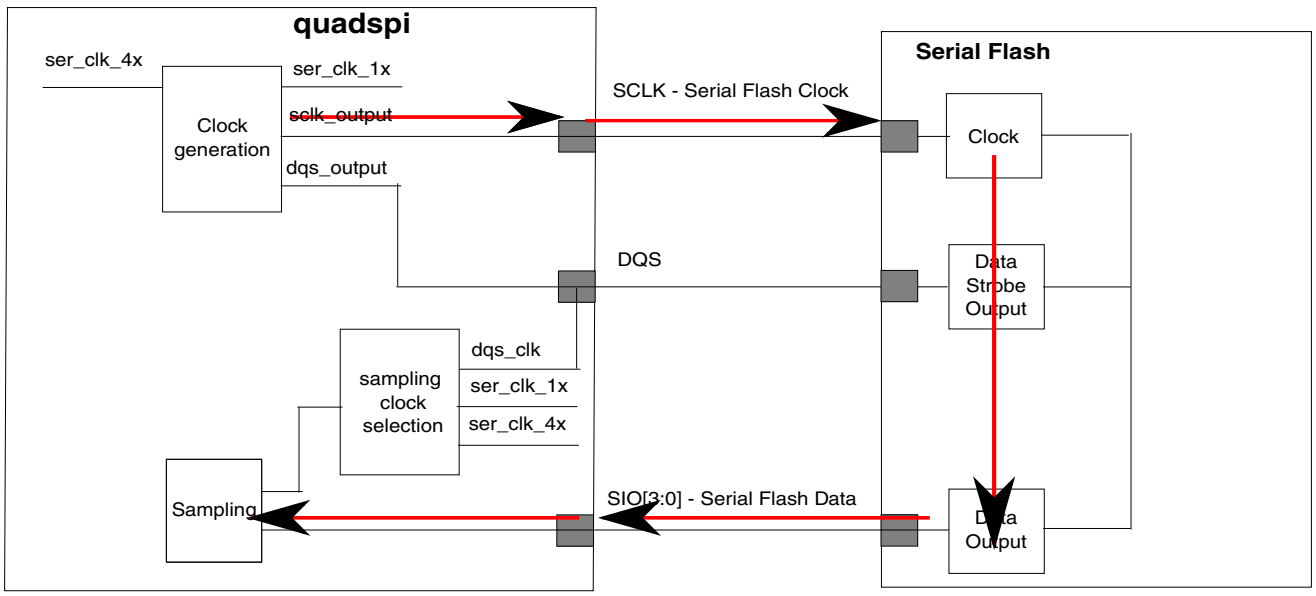


Figure 40-11. Serial flash data and clock path

NOTE

The red line is for data input path, the blue line is for DQS loopback path, and the purple line is for DQS input path from Serial Flash.

Total delay (Ttotal_delay_data) for serial flash data input is the sum of following delay:

1. Output delay of serial flash clock from internal serial clock to SCLK pad inside SOC
2. Wire delay of serial flash clock (SCLK) from SOC to external Serial Flash Device
3. Clock to Output Valid time of external Serial Flash Device
4. Wire delay of serial flash data (SIO) from external serial Flash Device to SOC
5. Input delay of serial flash data from SIO pad to internal sample register

Total delay (Ttotal_delay_loopback_dqs) for loopback DQS clock is the sum of following delay:

1. Output delay of DQS clock from internal serial clock to DQS pad inside SOC
2. Input delay of DQS clock from DQS pad to internal sample register

Total delay (Ttotal_delay_flash_dqs) for Flash DQS clock is the sum of following delay:

1. Output delay of serial flash clock from internal serial clock to SCLK pad inside SOC
2. Wire delay of serial flash clock (SCLK) from SOC to external Serial Flash Device

3. Clock to DQS Output time of external Serial Flash Device
4. Wire delay of serial flash data strobe (DQS) from external serial Flash Device to SOC
5. Input delay of DQS clock from DQS pad to internal sample register

40.10.1 Input timing in SDR mode with internal sampling

Input Timing diagram in SDR mode with internal sampling is show in the figure below.

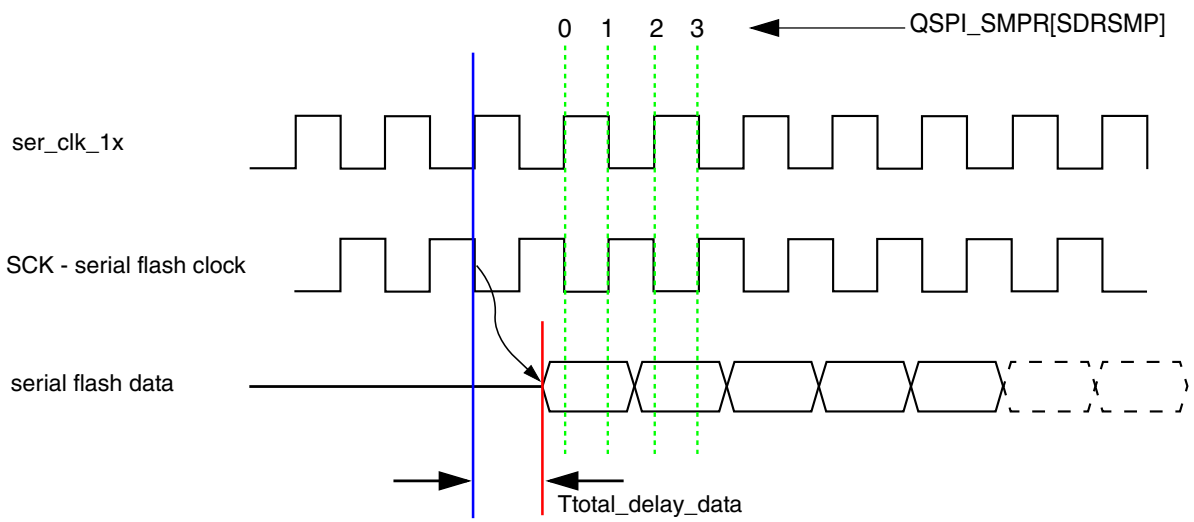


Figure 40-12. Internal sample SDR

There are four sample points for this sampling mode, which is determined by register field QSPI_SMPR.SDRSMP.

Sampling point need to be select correctly to meet both Setup and Hold timing for internal sample registers:

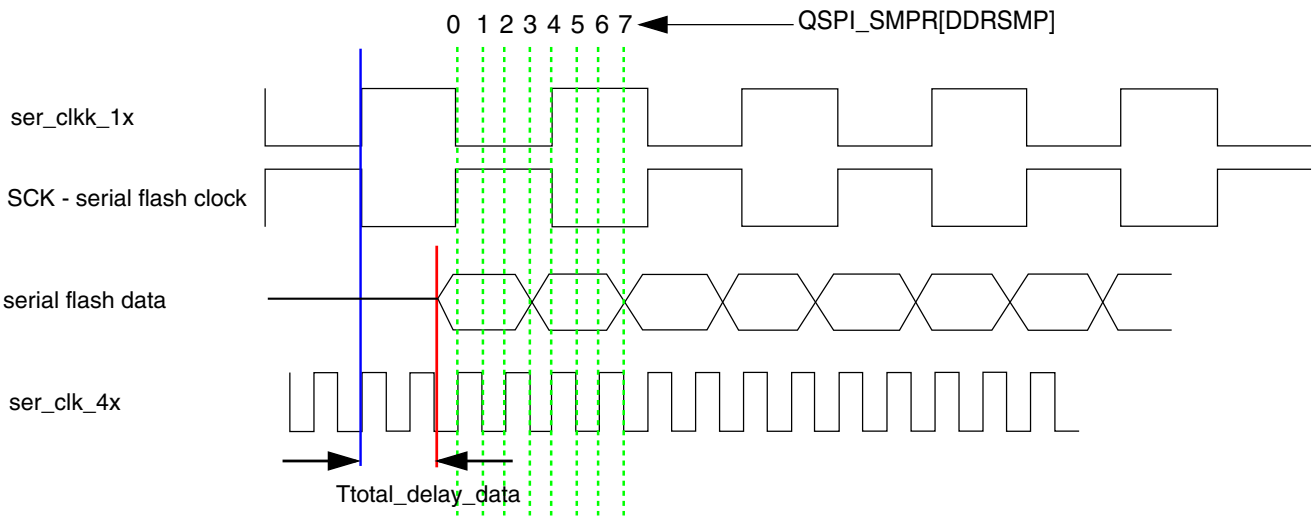
- For sample point N, Setup requirement is: Tcycle*(N+2)/2>Ttotal_delay_data,max
- For sample point N, Hold requirement is: Ttotal_delay_data,min>Tcycle*N/2

NOTE

Tcycle is the cycle of ser_clk_1x. Ttotal_delay_data,max is maximum delay of serial data input path. Ttotal_delay_data,min is the minimum delay of serial data input path. N=0,1,2,3

40.10.2 Input timing in DDR mode with internal sampling

Input Timing diagram in SDR mode with internal sampling is show in the following figure.



There are 8 sample points for this sampling mode, which is determined by register field DDRSMP.

Sampling point need to be select correctly to meet both Setup and Hold timing for internal sample registers:

- For sample point N, Setup requirement is: Tcycle * (N+2)/8 >Ttotal_delay_data,max
- For sample point N, Hold requirement is: Ttotal_delay_data,min> Tcycle * N/8

40.10.3 Input timing in SDR mode with loopback DQS sampling

Input Timing diagram in SDR mode with internal sampling is show in the following figure.

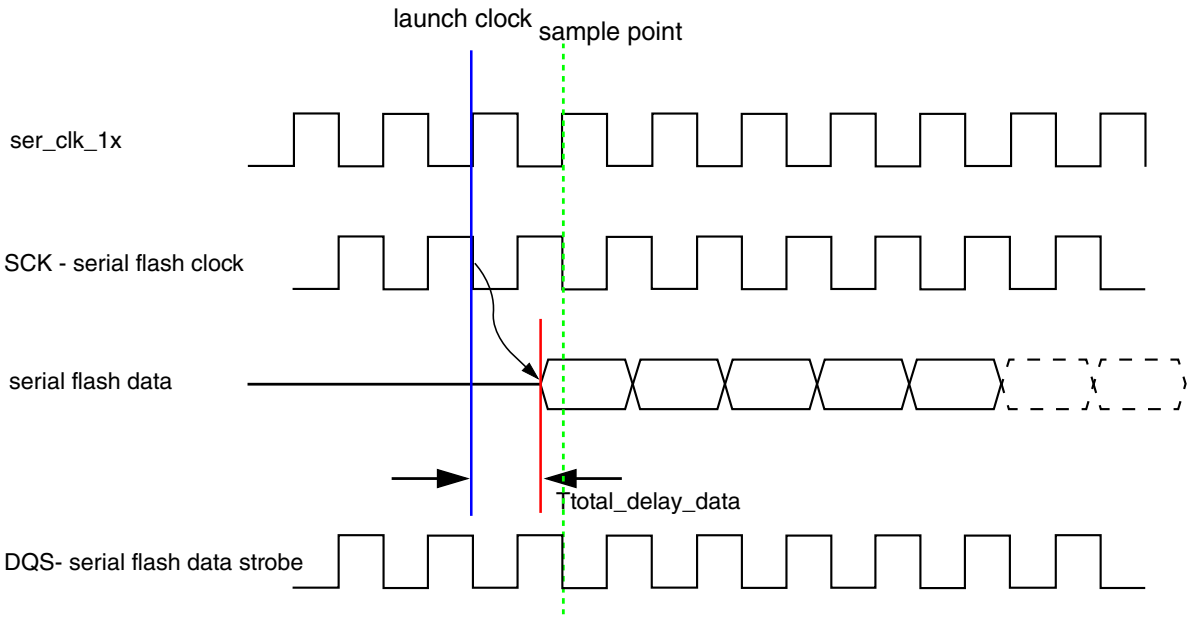


Figure 40-13. Input timing in SDR mode with loopback DQS sampling

In SDR mode and loopback sampling mode, DQS_PHASE_EN should be set 1.

For this sample point, the Setup requirement is: Tcycle >max(Ttotal_delay_data-Ttotal_delay_loopback_dqs). The Hold requirement is: min(Ttotal_delay_data-Ttotal_delay_loopback_dqs)> 0.

40.10.4 Input timing in DDR mode with loopback DQS sampling

Input Timing diagram in DDR mode with internal sampling is show in the following figure.

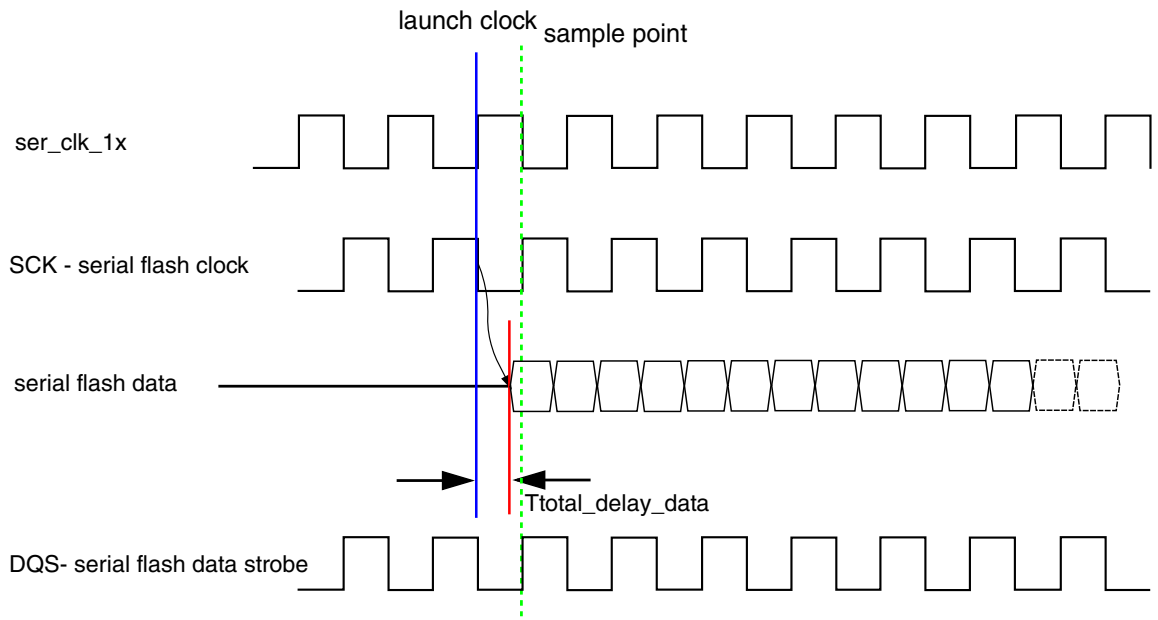


Figure 40-14. Input timing in DDR mode with loopback DQS sampling

In DDR mode and loopback sampling mode, DQS_PHASE_EN should be set 0.

For this sample point, the Setup requirement is: Tcycle >max(Ttotal_delay_data-Ttotal_delay_loopback_dqs). The Hold requirement is: min(Ttotal_delay_data-Ttotal_delay_loopback_dqs)> 0.

40.10.5 Input timing in SDR mode with flash DQS sampling

Input Timing diagram in SDR mode with internal sampling is show in the following figure.

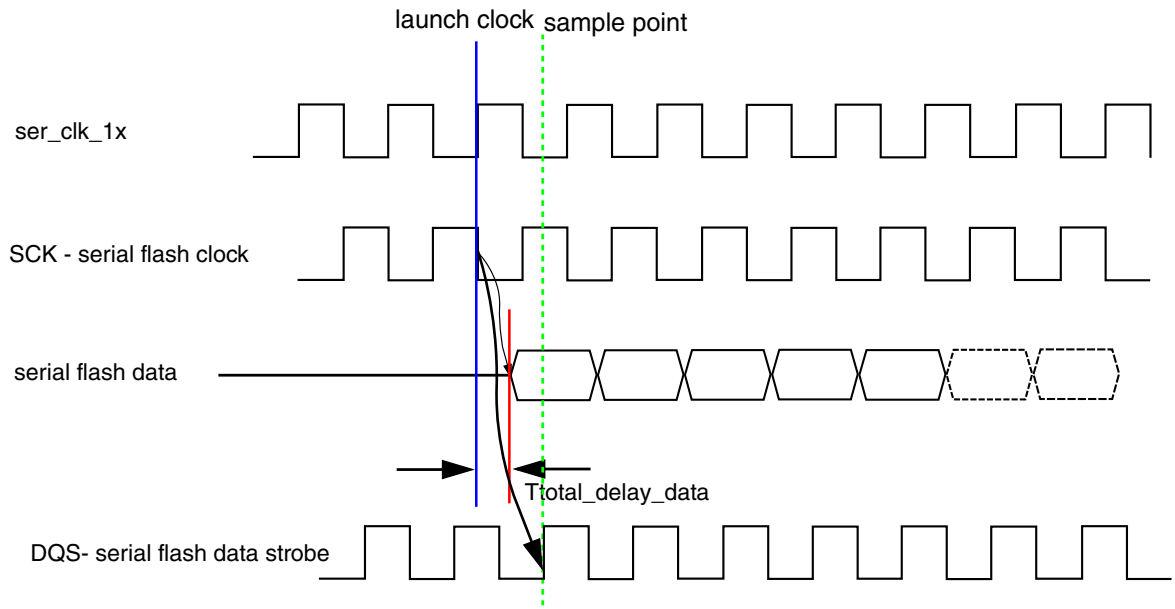


Figure 40-15. Input timing in SDR mode with flash DQS sampling

In this sampling mode, there will be a setup/hold requirement on Flash serial Data and Flash serial Data Strobe. This value will be specified in the data sheet.

40.10.6 Input timing in DDR mode with flash DQS sampling

Input Timing diagram in DDR mode with internal sampling is show in the following figure.

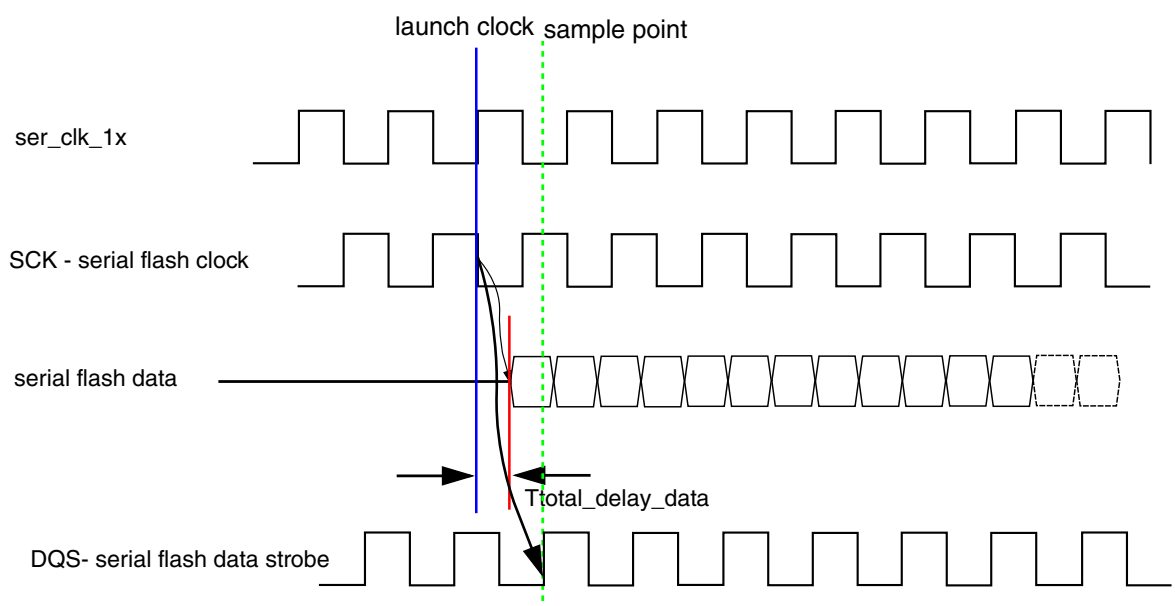


Figure 40-16. Input timing in DDR mode with flash DQS sampling

In this sampling mode, there will be a setup/hold requirement on Flash serial Data and Flash serial Data Strobe. This value will be specified in the data sheet.

40.10.7 Data Strobe Signal functionality

Some external serial flashes provide the data strobe (DQS/RDS) output which is fed directly to the QuadSPI module. The strobe (DQS/RDS) signal needs to be delayed to have the edges aligned to the data valid period. QuadSPI internally samples the incoming data at posedge of the strobe signal for SDR and on both the edges of the strobe signal for DDR. Refer to the figure for more detail.

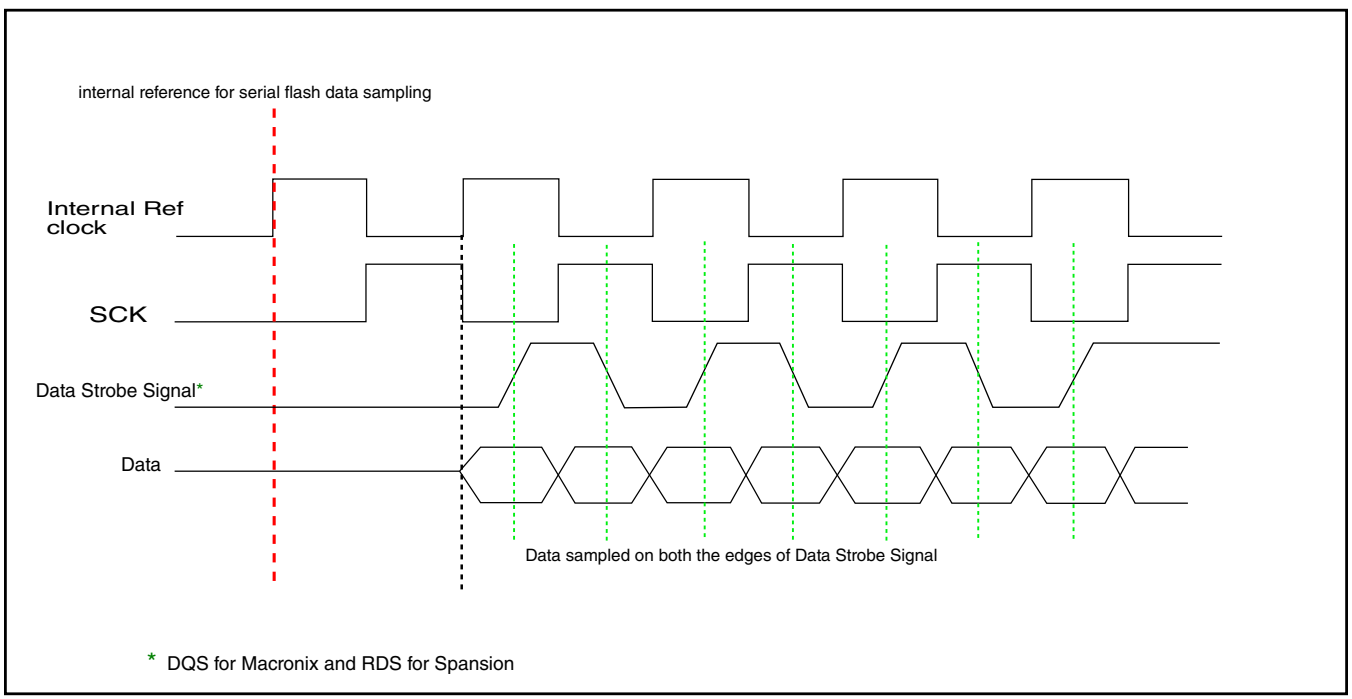


Figure 40-17. Data strobe signal functionality

40.11 Output timing in SDR mode

Output timing diagram in SDR mode is show in the following figure.

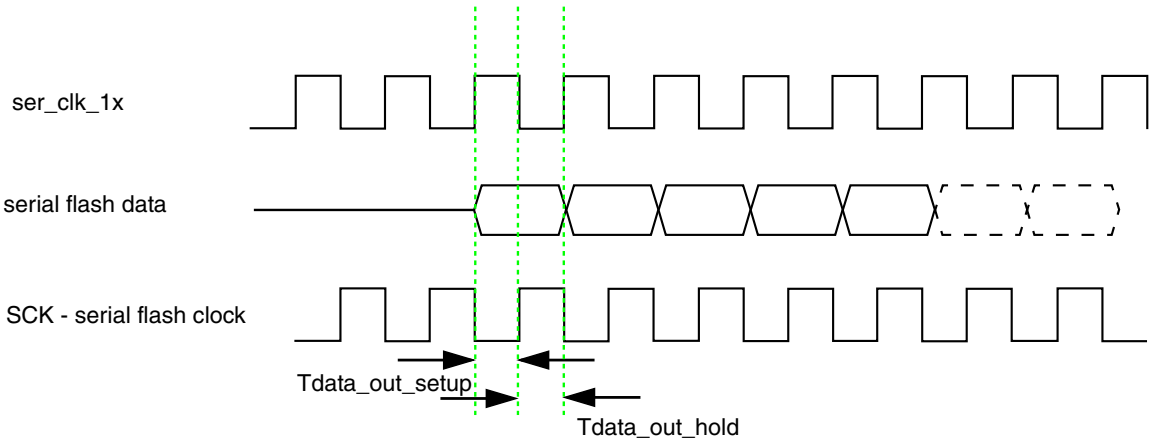


Figure 40-18. Output timing in SDR mode

In SDR mode, quadspi output serial data with internal serial clock rise edge 1x (ser_clk_1x). Flash Device has requirement on Data and Clock setup and hold timing.

40.12 Output timing in DDR mode

Output Timing diagram in DDR mode is show in the following figure.

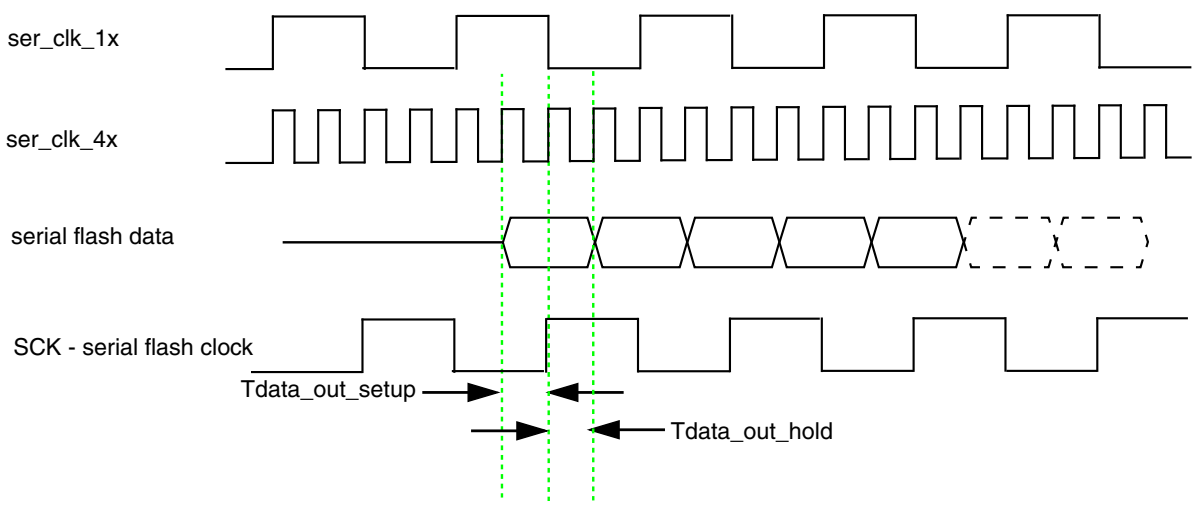


Figure 40-19. Output timing in DDR mode

In DDR mode, quadspi output serial data with internal serial clock both edge 1x (ser_clk_1x) and then delay one ser_clk_4x cycle for hold timing.

NOTE

TX_DDR_DELAY_EN should be set to 1 for DDR mode.

40.13 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)

40.13.1 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)

NOTE

See the System Memory map in this document for the base address of the QSPI AHB RX Data Buffer.

memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	AHB RX Data Buffer register (ARDB0)	32	R/W	0000_0000h	40.13.1.1/ 2468
4	AHB RX Data Buffer register (ARDB1)	32	R/W	0000_0000h	40.13.1.1/ 2468
8	AHB RX Data Buffer register (ARDB2)	32	R/W	0000_0000h	40.13.1.1/ 2468
C	AHB RX Data Buffer register (ARDB3)	32	R/W	0000_0000h	40.13.1.1/ 2468
10	AHB RX Data Buffer register (ARDB4)	32	R/W	0000_0000h	40.13.1.1/ 2468
14	AHB RX Data Buffer register (ARDB5)	32	R/W	0000_0000h	40.13.1.1/ 2468
18	AHB RX Data Buffer register (ARDB6)	32	R/W	0000_0000h	40.13.1.1/ 2468
1C	AHB RX Data Buffer register (ARDB7)	32	R/W	0000_0000h	40.13.1.1/ 2468
20	AHB RX Data Buffer register (ARDB8)	32	R/W	0000_0000h	40.13.1.1/ 2468
24	AHB RX Data Buffer register (ARDB9)	32	R/W	0000_0000h	40.13.1.1/ 2468
28	AHB RX Data Buffer register (ARDB10)	32	R/W	0000_0000h	40.13.1.1/ 2468
2C	AHB RX Data Buffer register (ARDB11)	32	R/W	0000_0000h	40.13.1.1/ 2468
30	AHB RX Data Buffer register (ARDB12)	32	R/W	0000_0000h	40.13.1.1/ 2468

Table continues on the next page...

memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	AHB RX Data Buffer register (ARDB13)	32	R/W	0000_0000h	40.13.1.1/ 2468
38	AHB RX Data Buffer register (ARDB14)	32	R/W	0000_0000h	40.13.1.1/ 2468
3C	AHB RX Data Buffer register (ARDB15)	32	R/W	0000_0000h	40.13.1.1/ 2468
40	AHB RX Data Buffer register (ARDB16)	32	R/W	0000_0000h	40.13.1.1/ 2468
44	AHB RX Data Buffer register (ARDB17)	32	R/W	0000_0000h	40.13.1.1/ 2468
48	AHB RX Data Buffer register (ARDB18)	32	R/W	0000_0000h	40.13.1.1/ 2468
4C	AHB RX Data Buffer register (ARDB19)	32	R/W	0000_0000h	40.13.1.1/ 2468
50	AHB RX Data Buffer register (ARDB20)	32	R/W	0000_0000h	40.13.1.1/ 2468
54	AHB RX Data Buffer register (ARDB21)	32	R/W	0000_0000h	40.13.1.1/ 2468
58	AHB RX Data Buffer register (ARDB22)	32	R/W	0000_0000h	40.13.1.1/ 2468
5C	AHB RX Data Buffer register (ARDB23)	32	R/W	0000_0000h	40.13.1.1/ 2468
60	AHB RX Data Buffer register (ARDB24)	32	R/W	0000_0000h	40.13.1.1/ 2468
64	AHB RX Data Buffer register (ARDB25)	32	R/W	0000_0000h	40.13.1.1/ 2468
68	AHB RX Data Buffer register (ARDB26)	32	R/W	0000_0000h	40.13.1.1/ 2468
6C	AHB RX Data Buffer register (ARDB27)	32	R/W	0000_0000h	40.13.1.1/ 2468
70	AHB RX Data Buffer register (ARDB28)	32	R/W	0000_0000h	40.13.1.1/ 2468
74	AHB RX Data Buffer register (ARDB29)	32	R/W	0000_0000h	40.13.1.1/ 2468
78	AHB RX Data Buffer register (ARDB30)	32	R/W	0000_0000h	40.13.1.1/ 2468
7C	AHB RX Data Buffer register (ARDB31)	32	R/W	0000_0000h	40.13.1.1/ 2468

40.13.1.1 AHB RX Data Buffer register (ARDB*n*)

The AHB RX Data Buffer register 0 to 31 can be used to read the buffer content of the RX Buffer from successive addresses. QSPI_ARDB0 corresponds to the RX Buffer register entry corresponding to the current value of the read pointer with increasing order.

The increment of the read pointer depends from the access scheme (DMA or flag-driven). Refer to "Data Transfer from the QuadSPI Module Internal Buffers" section in Flash Read section, RX Buffer, data read via register interface and AHB read, for the description of successive accesses to the RX Buffer content. Refer also to Byte Ordering of Serial Flash Read Data for the byte ordering scheme.

Valid address range accessible in the QSPI_ARDBn range depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

- Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QSPI_ARDB0 to QSPI_ARDB31.
- Example 2, RX Buffer filled with 5 valid words, RX Buffer fill level QSPI_RBSR[RDBFL] is 5. In this case an access to QSPI_ARDB4 provides the last valid entry.

Address: 0h base + 0h offset + (4d × i), where i=0d to 31d

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

ARDB*n* field descriptions

Field	Description
ARXD	ARDB provided RX Buffer Data. Byte order (endianness) is identical to the RX Buffer Data Registers.

40.14 Peripheral Bus Register Descriptions

This section provides the peripheral bus register information of the QuadSPI module.

This section provides the memory map and register definitions of the QuadSPI module.

QuadSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_0000	Module Configuration Register (QuadSPI_MCR)	32	R/W	000F_4000h	40.14.1/ 2476
21E_0008	IP Configuration Register (QuadSPI_IPCR)	32	R/W	0000_0000h	40.14.2/ 2479
21E_000C	Flash Configuration Register (QuadSPI_FLSHCR)	32	R/W	0000_0303h	40.14.3/ 2480
21E_0010	Buffer0 Configuration Register (QuadSPI_BUF0CR)	32	R/W	0000_0000h	40.14.4/ 2480
21E_0014	Buffer1 Configuration Register (QuadSPI_BUF1CR)	32	R/W	0000_0000h	40.14.5/ 2481
21E_0018	Buffer2 Configuration Register (QuadSPI_BUF2CR)	32	R/W	0000_0000h	40.14.6/ 2482
21E_001C	Buffer3 Configuration Register (QuadSPI_BUF3CR)	32	R/W	See section	40.14.7/ 2483
21E_0020	Buffer Generic Configuration Register (QuadSPI_BFGENCR)	32	R/W	0000_0000h	40.14.8/ 2484
21E_0030	Buffer0 Top Index Register (QuadSPI_BUF0IND)	32	R/W	0000_0000h	40.14.9/ 2484
21E_0034	Buffer1 Top Index Register (QuadSPI_BUF1IND)	32	R/W	0000_0000h	40.14.10/ 2485
21E_0038	Buffer2 Top Index Register (QuadSPI_BUF2IND)	32	R/W	0000_0000h	40.14.11/ 2486
21E_0100	Serial Flash Address Register (QuadSPI_SFAR)	32	R/W	0000_0000h	40.14.12/ 2487
21E_0108	Sampling Register (QuadSPI_SMPR)	32	R/W	0000_0000h	40.14.13/ 2487
21E_010C	RX Buffer Status Register (QuadSPI_RBSR)	32	R	0000_0000h	40.14.14/ 2488
21E_0110	RX Buffer Control Register (QuadSPI_RBCT)	32	R/W	0000_0000h	40.14.15/ 2489
21E_0150	TX Buffer Status Register (QuadSPI_TBSR)	32	R	0000_0000h	40.14.16/ 2490
21E_0154	TX Buffer Data Register (QuadSPI_TBDR)	32	R/W	0000_0000h	40.14.17/ 2490
21E_015C	Status Register (QuadSPI_SR)	32	R	0000_3800h	40.14.18/ 2492
21E_0160	Flag Register (QuadSPI_FR)	32	w1c	0800_0000h	40.14.19/ 2495
21E_0164	Interrupt and DMA Request Select and Enable Register (QuadSPI_RSER)	32	R/W	0000_0000h	40.14.20/ 2498
21E_0168	Sequence Suspend Status Register (QuadSPI_SPNDST)	32	R	0000_0000h	40.14.21/ 2501
21E_016C	Sequence Pointer Clear Register (QuadSPI_SPTRCLR)	32	R/W	0000_0000h	40.14.22/ 2503

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_0180	Serial Flash A1 Top Address (QuadSPI_SFA1AD)	32	R/W	0000_0000h	40.14.23/ 2503
21E_0184	Serial Flash A2 Top Address (QuadSPI_SFA2AD)	32	R/W	0000_0000h	40.14.24/ 2504
21E_0188	Serial Flash B1Top Address (QuadSPI_SFB1AD)	32	R/W	0000_0000h	40.14.25/ 2504
21E_018C	Serial Flash B2Top Address (QuadSPI_SFB2AD)	32	R/W	0000_0000h	40.14.26/ 2505
21E_0200	RX Buffer Data Register (QuadSPI_RBDR0)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0204	RX Buffer Data Register (QuadSPI_RBDR1)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0208	RX Buffer Data Register (QuadSPI_RBDR2)	32	R/W	0000_0000h	40.14.27/ 2505
21E_020C	RX Buffer Data Register (QuadSPI_RBDR3)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0210	RX Buffer Data Register (QuadSPI_RBDR4)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0214	RX Buffer Data Register (QuadSPI_RBDR5)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0218	RX Buffer Data Register (QuadSPI_RBDR6)	32	R/W	0000_0000h	40.14.27/ 2505
21E_021C	RX Buffer Data Register (QuadSPI_RBDR7)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0220	RX Buffer Data Register (QuadSPI_RBDR8)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0224	RX Buffer Data Register (QuadSPI_RBDR9)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0228	RX Buffer Data Register (QuadSPI_RBDR10)	32	R/W	0000_0000h	40.14.27/ 2505
21E_022C	RX Buffer Data Register (QuadSPI_RBDR11)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0230	RX Buffer Data Register (QuadSPI_RBDR12)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0234	RX Buffer Data Register (QuadSPI_RBDR13)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0238	RX Buffer Data Register (QuadSPI_RBDR14)	32	R/W	0000_0000h	40.14.27/ 2505
21E_023C	RX Buffer Data Register (QuadSPI_RBDR15)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0240	RX Buffer Data Register (QuadSPI_RBDR16)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0244	RX Buffer Data Register (QuadSPI_RBDR17)	32	R/W	0000_0000h	40.14.27/ 2505

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_0248	RX Buffer Data Register (QuadSPI_RBDR18)	32	R/W	0000_0000h	40.14.27/ 2505
21E_024C	RX Buffer Data Register (QuadSPI_RBDR19)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0250	RX Buffer Data Register (QuadSPI_RBDR20)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0254	RX Buffer Data Register (QuadSPI_RBDR21)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0258	RX Buffer Data Register (QuadSPI_RBDR22)	32	R/W	0000_0000h	40.14.27/ 2505
21E_025C	RX Buffer Data Register (QuadSPI_RBDR23)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0260	RX Buffer Data Register (QuadSPI_RBDR24)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0264	RX Buffer Data Register (QuadSPI_RBDR25)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0268	RX Buffer Data Register (QuadSPI_RBDR26)	32	R/W	0000_0000h	40.14.27/ 2505
21E_026C	RX Buffer Data Register (QuadSPI_RBDR27)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0270	RX Buffer Data Register (QuadSPI_RBDR28)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0274	RX Buffer Data Register (QuadSPI_RBDR29)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0278	RX Buffer Data Register (QuadSPI_RBDR30)	32	R/W	0000_0000h	40.14.27/ 2505
21E_027C	RX Buffer Data Register (QuadSPI_RBDR31)	32	R/W	0000_0000h	40.14.27/ 2505
21E_0300	LUT Key Register (QuadSPI_LUTKEY)	32	R/W	5AF0_5AF0h	40.14.28/ 2506
21E_0304	LUT Lock Configuration Register (QuadSPI_LCKCR)	32	R/W	0000_0002h	40.14.29/ 2507
21E_0310	Look-up Table register (QuadSPI_LUT0)	32	R/W	0818_0403h	40.14.30/ 2508
21E_0314	Look-up Table register (QuadSPI_LUT1)	32	R/W	2400_1C08h	40.14.31/ 2509
21E_0318	Look-up Table register (QuadSPI_LUT2)	32	R/W	0000_0000h	40.14.32/ 2510
21E_031C	Look-up Table register (QuadSPI_LUT3)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0320	Look-up Table register (QuadSPI_LUT4)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0324	Look-up Table register (QuadSPI_LUT5)	32	R/W	0000_0000h	40.14.32/ 2510

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_0328	Look-up Table register (QuadSPI_LUT6)	32	R/W	0000_0000h	40.14.32/ 2510
21E_032C	Look-up Table register (QuadSPI_LUT7)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0330	Look-up Table register (QuadSPI_LUT8)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0334	Look-up Table register (QuadSPI_LUT9)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0338	Look-up Table register (QuadSPI_LUT10)	32	R/W	0000_0000h	40.14.32/ 2510
21E_033C	Look-up Table register (QuadSPI_LUT11)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0340	Look-up Table register (QuadSPI_LUT12)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0344	Look-up Table register (QuadSPI_LUT13)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0348	Look-up Table register (QuadSPI_LUT14)	32	R/W	0000_0000h	40.14.32/ 2510
21E_034C	Look-up Table register (QuadSPI_LUT15)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0350	Look-up Table register (QuadSPI_LUT16)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0354	Look-up Table register (QuadSPI_LUT17)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0358	Look-up Table register (QuadSPI_LUT18)	32	R/W	0000_0000h	40.14.32/ 2510
21E_035C	Look-up Table register (QuadSPI_LUT19)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0360	Look-up Table register (QuadSPI_LUT20)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0364	Look-up Table register (QuadSPI_LUT21)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0368	Look-up Table register (QuadSPI_LUT22)	32	R/W	0000_0000h	40.14.32/ 2510
21E_036C	Look-up Table register (QuadSPI_LUT23)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0370	Look-up Table register (QuadSPI_LUT24)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0374	Look-up Table register (QuadSPI_LUT25)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0378	Look-up Table register (QuadSPI_LUT26)	32	R/W	0000_0000h	40.14.32/ 2510
21E_037C	Look-up Table register (QuadSPI_LUT27)	32	R/W	0000_0000h	40.14.32/ 2510

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_0380	Look-up Table register (QuadSPI_LUT28)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0384	Look-up Table register (QuadSPI_LUT29)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0388	Look-up Table register (QuadSPI_LUT30)	32	R/W	0000_0000h	40.14.32/ 2510
21E_038C	Look-up Table register (QuadSPI_LUT31)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0390	Look-up Table register (QuadSPI_LUT32)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0394	Look-up Table register (QuadSPI_LUT33)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0398	Look-up Table register (QuadSPI_LUT34)	32	R/W	0000_0000h	40.14.32/ 2510
21E_039C	Look-up Table register (QuadSPI_LUT35)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03A0	Look-up Table register (QuadSPI_LUT36)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03A4	Look-up Table register (QuadSPI_LUT37)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03A8	Look-up Table register (QuadSPI_LUT38)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03AC	Look-up Table register (QuadSPI_LUT39)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03B0	Look-up Table register (QuadSPI_LUT40)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03B4	Look-up Table register (QuadSPI_LUT41)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03B8	Look-up Table register (QuadSPI_LUT42)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03BC	Look-up Table register (QuadSPI_LUT43)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03C0	Look-up Table register (QuadSPI_LUT44)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03C4	Look-up Table register (QuadSPI_LUT45)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03C8	Look-up Table register (QuadSPI_LUT46)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03CC	Look-up Table register (QuadSPI_LUT47)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03D0	Look-up Table register (QuadSPI_LUT48)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03D4	Look-up Table register (QuadSPI_LUT49)	32	R/W	0000_0000h	40.14.32/ 2510

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_03D8	Look-up Table register (QuadSPI_LUT50)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03DC	Look-up Table register (QuadSPI_LUT51)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03E0	Look-up Table register (QuadSPI_LUT52)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03E4	Look-up Table register (QuadSPI_LUT53)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03E8	Look-up Table register (QuadSPI_LUT54)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03EC	Look-up Table register (QuadSPI_LUT55)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03F0	Look-up Table register (QuadSPI_LUT56)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03F4	Look-up Table register (QuadSPI_LUT57)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03F8	Look-up Table register (QuadSPI_LUT58)	32	R/W	0000_0000h	40.14.32/ 2510
21E_03FC	Look-up Table register (QuadSPI_LUT59)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0400	Look-up Table register (QuadSPI_LUT60)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0404	Look-up Table register (QuadSPI_LUT61)	32	R/W	0000_0000h	40.14.32/ 2510
21E_0408	Look-up Table register (QuadSPI_LUT62)	32	R/W	0000_0000h	40.14.32/ 2510
21E_040C	Look-up Table register (QuadSPI_LUT63)	32	R/W	0000_0000h	40.14.32/ 2510

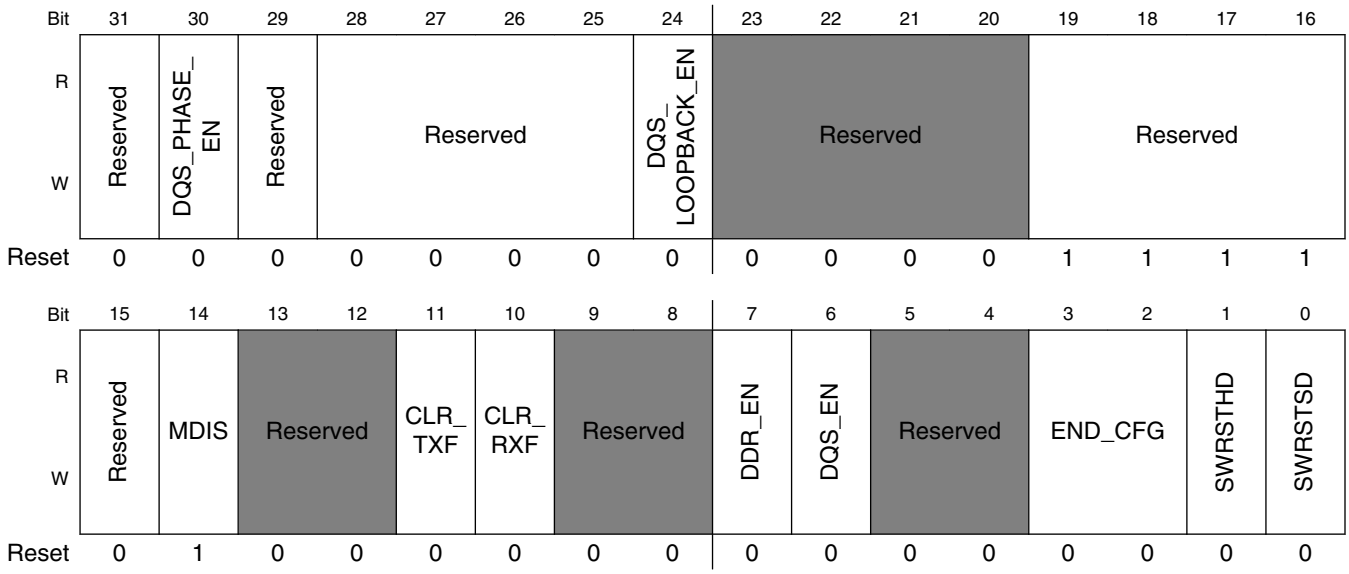
40.14.1 Module Configuration Register (QuadSPI_MCR)

The QuadSPI_MCR holds configuration data associated with QuadSPI operation.

Write:

- *All other fields: Anytime*

Address: 21E_0000h base + 0h offset = 21E_0000h



QuadSPI_MCR field descriptions

Field	Description
31 Reserved	This field is reserved. This field should always be set to 0.
30 DQS_PHASE_EN	This bit controls internal DQS output phase. If DQS_EN and DQS_LOOPBACK_EN are both set to 1, this bit should be set in SDR mode, and cleared in DDR mode. If either DQS_EN or DQS_LOOPBACK_EN is set to 0, this bit is ignored.
29 Reserved	This field is reserved. This field should always be set to 0.
28–25 Reserved	This field is reserved. This field should always be set to 0.
24 DQS_LOOPBACK_EN	Quadspi will output serial data strobe signal which will be loopback from pad to sample input flash serial data. Please note pad should be force input for loopback.Quadspi will output serial data strobe signal which will be loopback from pad to sample input flash serial data. Please note pad should be force input for loopback. This bit is a don't care when DQS_EN is set to 0. 1'b1 DQS loopback sampling enabled 1'b0 DQS loopback sampling disabled
23–20 Reserved	This field is reserved.
19–16 Reserved	This field is reserved. This field is reserved and should always be set to 0xF.
15 Reserved	This field is reserved. This field is reserved.
14 MDIS	Module Disable. The MDIS bit allows the clock to the non-memory mapped logic in the QuadSPI to be stopped, putting the QuadSPI in a software controlled power-saving state. Please refer to , for more information.

Table continues on the next page...

QuadSPI_MCR field descriptions (continued)

Field	Description
	0 Enable QuadSPI clocks. 1 Allow external logic to disable QuadSPI clocks.
13–12 Reserved	This field is reserved.
11 CLR_TXF	Clear TX FIFO/Buffer. Invalidate the TX Buffer content. 0 No action. 1 Read and write pointers of the TX Buffer are reset to 0. QSPI_TBSR[TRCTR] is reset to 0.
10 CLR_RXF	Clear RX FIFO. Invalidate the RX Buffer. 0 No action. 1 Read and write pointers of the RX Buffer are reset to 0. QSPI_RBSR[RDBFL] is reset to 0.
9–8 Reserved	This field is reserved.
7 DDR_EN	DDR mode enable: 0 2x and 4x clocks are disabled for SDR instructions only 1 2x and 4x clocks are enabled supports both SDR and DDR instruction.
6 DQS_EN	DQS enable: This field is valid for both SDR and DDR mode.For more details Refer Data Strobe Signal Functionality 0 DQS disabled. 1 DQS enabled- When enabled, the incoming data is sampled on both the edges of DQS input when QSPI_MCR[DDR_EN] is set, else, on only one edge when QSPI_MCR[DDR_EN] is 0. The QSPI_SMPR[DDR_SMP] values are ignored.
5–4 Reserved	This field is reserved.
3–2 END_CFG	Defines the endianness of the QSPI module.For more details refer to Byte Ordering Endianess
1 SWRSTHD	Software reset for AHB domain **NOTE:** Please keep other fields value when write to SWRSTHD and SWRSTSD **NOTE:** These software reset don't reset register setting but only reset internal flip-flops in quadspi controller **NOTE:** To remove the reset, need to write 0 to SWRSTHD and SWRSTSD 0 No action 1 AHB domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects.
0 SWRSTSD	Software reset for Serial Flash domain **NOTE:** Please keep other fields value when write to SWRSTHD and SWRSTSD **NOTE:** These software reset don't reset register setting but only reset internal flip-flops in quadspi controller **NOTE:** To remove the reset, need to write 0 to SWRSTHD and SWRSTSD

Table continues on the next page...

QuadSPI_MCR field descriptions (continued)

Field	Description
	0 No action 1 Serial Flash domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects.

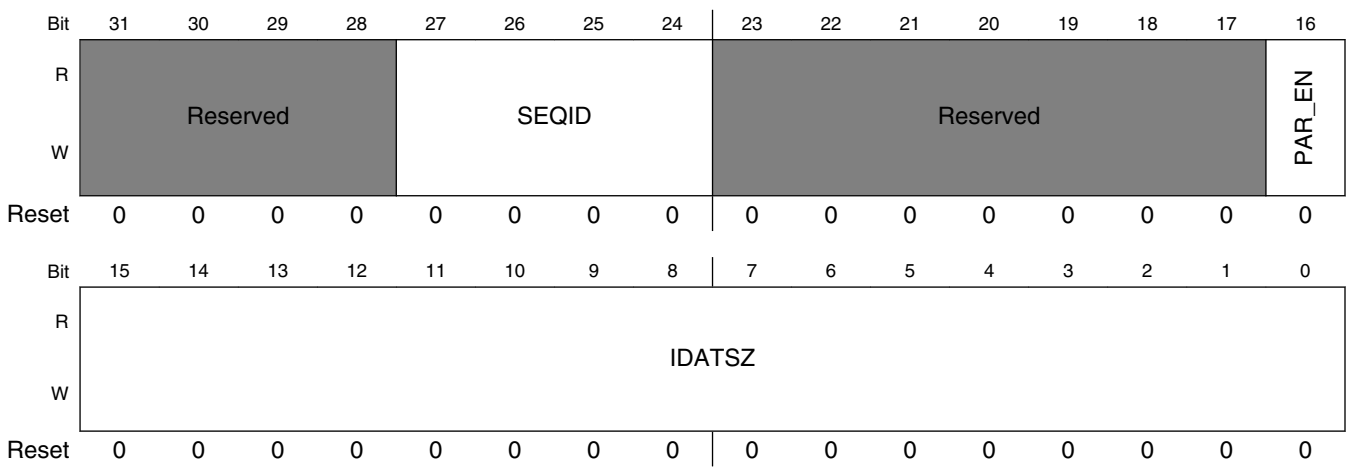
40.14.2 IP Configuration Register (QuadSPI_IPCR)

The IP configuration register provides all the configuration required for an IP initiated command. An IP command can be triggered by writing in the SEQID field of this register. If the SEQID field is written successfully, a new command to the external serial flash is started as per the sequence pointed to by the SEQID field. Refer to Normal Mode , for details about the command triggering and command execution.

Write:

- *QSPI_SR[IP_ACC]=0*

Address: 21E_0000h base + 8h offset = 21E_0008h



QuadSPI_IPCR field descriptions

Field	Description
31–28 Reserved	This field is reserved.
27–24 SEQID	Points to a sequence in the Look-up-table. The SEQID defines the bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table for more details. A write to this bit -field triggers a transaction on the serial flash interface.
23–17 Reserved	This field is reserved.
16 PAR_EN	When set, a transaction to two serial flash devices is triggered in parallel mode. Refer to Parallel Flash Mode for more details.
IDATSZ	IP data transfer size: Defines the data transfer size in bytes of the IP command.

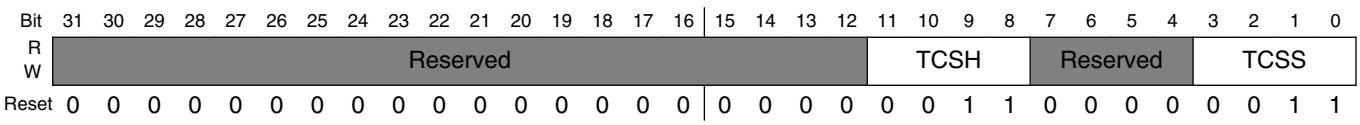
40.14.3 Flash Configuration Register (QuadSPI_FLSHCR)

The Flash configuration register contains the flash device specific timings that must be met by the QuadSPI controller for the device to function correctly.

Write:

- *QSPI_SR[AHB_ACC] = 0*
- *QSPI_SR[IP_ACC] = 0*

Address: 21E_0000h base + Ch offset = 21E_000Ch



QuadSPI_FLSHCR field descriptions

Field	Description
31–12 Reserved	This field is reserved. Reserved.
11–8 TCSH	Serial flash CS hold time in terms of serial flash clock cycles. **NOTE:** The actual delay between chip select assertion and clock fall edge is defined as: • 1 SCK cycle if TCSH is set 0 or 1 • N SCK cycle if TCSH is set N (N>1)
7–4 Reserved	This field is reserved. Reserved.
TCSS	Serial flash CS setup time in terms of serial flash clock cycles. **NOTE:** 1. The actual delay between chip select assertion and clock fall edge is defined as: • 0.5 SCK cycle if TCSS is set 0 or 1 • N+0.5 SCK cycle if TCSS is set N (N>1) 2. Any update to TCSS register bits is visible on the flash interface only from the second transaction following the update.

40.14.4 Buffer0 Configuration Register (QuadSPI_BUF0CR)

This register provides the configuration for any access to buffer0. An access is routed to buffer0 when the master port number of the incoming AHB request matches the MSTRID field of the BUF0CR. Any buffer "miss" leads to a serial flash transaction being triggered as per the sequence pointed to the SEQID field. Buffer0 may also be configured as a high priority buffer by setting the HP_EN field of this register.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 10h offset = 21E_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HP_EN	Reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

QuadSPI_BUF0CR field descriptions

Field	Description
31 HP_EN	High Priority Enable: When set, the master associated with this buffer is assigned a priority higher than the rest of the masters. An access by a high priority master will suspend any ongoing prefetch by another AHB master and will be serviced on high priority. Refer to Flexible AHB Buffers for details.
30–16 Reserved	This field is reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash. For example,a value of 0x2 will set transfer size to 16bytes.When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved. Reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER0. Any AHB access with this master port number is routed to this buffer.

40.14.5 Buffer1 Configuration Register (QuadSPI_BUF1CR)

This register provides the configuration for any access to buffer1. An access is routed to buffer1 when the master port number of the incoming AHB request matches the MSTRID field of the BUF1CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 14h offset = 21E_0014h

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

QuadSPI_BUF1CR field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash.For example, a value of 0x2 will set transfer size to 16bytes. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER1. Any AHB access with this master port number is routed to this buffer.

40.14.6 Buffer2 Configuration Register (QuadSPI_BUF2CR)

This register provides the configuration for any access to buffer2. An access is routed to buffer2 when the master port number of the incoming AHB request matches the MSTRID field of the BUF2CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 18h offset = 21E_0018h

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

QuadSPI_BUF2CR field descriptions

Field	Description
31–16 Reserved	This field is reserved. Reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 Bytes of an AHB triggered access to serial flash. For example, a value of 0x2 will set transfer size to 16bytes.When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved. Reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER2. Any AHB access with this master port number is routed to this buffer.

40.14.7 Buffer3 Configuration Register (QuadSPI_BUF3CR)

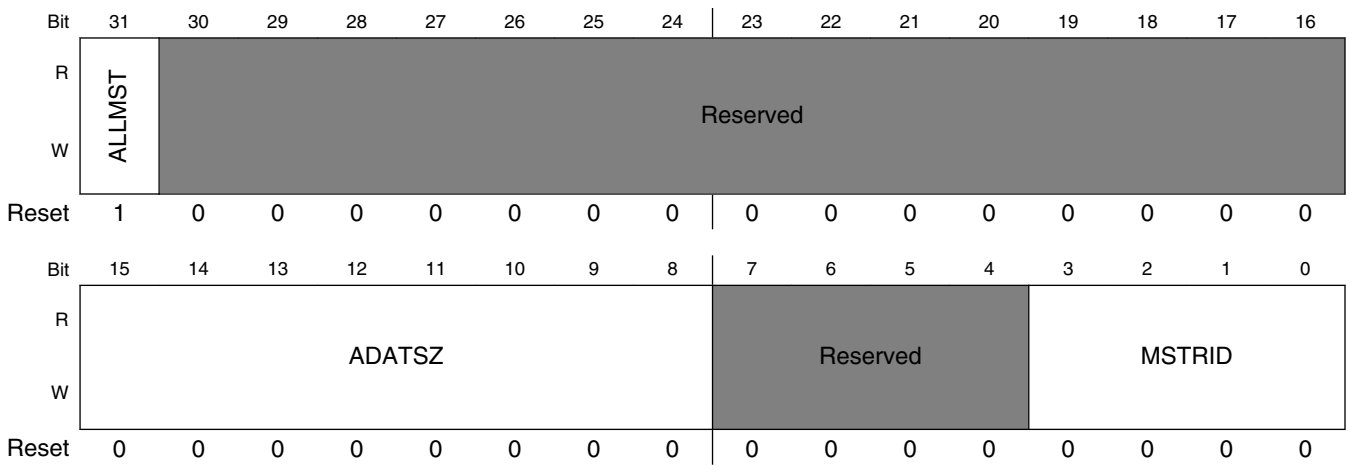
This register provides the configuration for any access to buffer3. An access is routed to buffer3 when the master port number of the incoming AHB request matches the MSTRID field of the BUF3CR. Any buffer "miss" leads to the buffer being flushed a serial flash transaction being triggered as per the sequence pointed to by the SEQID field. If the ALLMST field is set, any transaction where the master port number does not match any of the buffer MSTRID fields will be routed to this buffer. In the case that the ALLMST field is not set, any such transaction (where master port number does not match any of the MSTRID fields) will be returned an ERROR response.

Write:

- *QSPI_SR[AHB_ACC] = 0*

.

Address: 21E_0000h base + 1Ch offset = 21E_001Ch



QuadSPI_BUF3CR field descriptions

Field	Description
31 ALLMST	All master enable: When set, buffer3 acts as an all-master buffer. Any AHB access with a master port number not matching with the master ID of buffer0 or buffer1 or buffer2 is routed to buffer3. When set, the MSTRID field of this register is ignored.
30–16 Reserved	This field is reserved. Reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 Bytes of an AHB triggered access to serial flash. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER3. Any AHB access with this master port number is routed to this buffer.

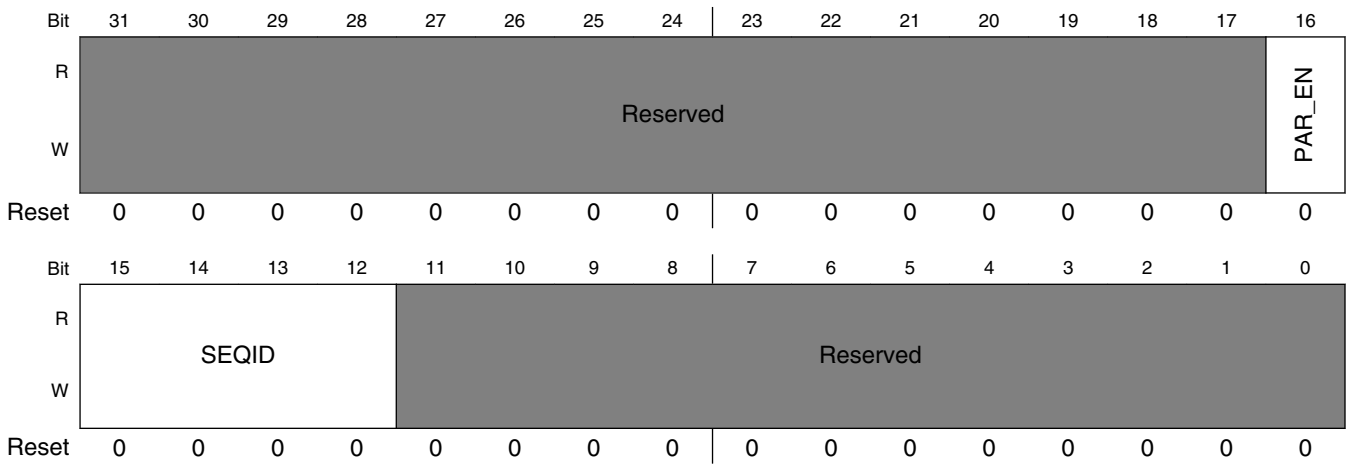
40.14.8 Buffer Generic Configuration Register (QuadSPI_BFGENCR)

This register provides the generic configuration to any of the buffer accesses. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field. If the PAR_EN field is set, all the buffer accesses result in parallel accesses to the flashes.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 20h offset = 21E_0020h



QuadSPI_BFGENCR field descriptions

Field	Description
31–17 Reserved	This field is reserved.
16 PAR_EN	When set, a transaction to two serial flash devices is triggered in parallel mode. Refer to Parallel Flash Mode for more details.
15–12 SEQID	Points to a sequence in the Look-up-table. The SEQID defines the bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table. **NOTE:** If the sequence pointer differs between the new and previous sequence then the user should reset this. See QSPI_SPTRCLR for more information.
Reserved	This field is reserved.

40.14.9 Buffer0 Top Index Register (QuadSPI_BUF0IND)

This register specifies the top index of buffer0, which defines its size. Note that that the 3 LSBs of this register are set to zero - this ensures that the buffer is 64bit aligned, as each buffer entry is 64bits long.

The register value should be set to the desired number of bytes less 8. For example, setting BUF0IND to 0 gives 8 bytes, 1 give 16bytes etc.

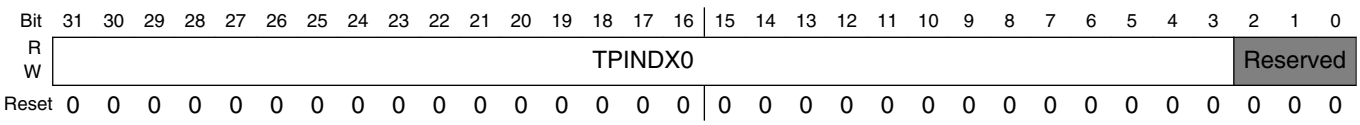
The size of buffer0 is the difference between the BUF0IND+8 and 0.

It is the responsibility of the software to ensure that BUF0IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 30h offset = 21E_0030h



QuadSPI_BUF0IND field descriptions

Field	Description
31–3 TPINDX0	Top index of buffer 0.
Reserved	This field is reserved. Reserved.

40.14.10 Buffer1 Top Index Register (QuadSPI_BUF1IND)

This register specifies the top index of buffer1, which defines its size. Note that the 3 LSBs of this register are set to zero - this ensures that the buffer is 64bit aligned as each buffer entry is 64bits long.

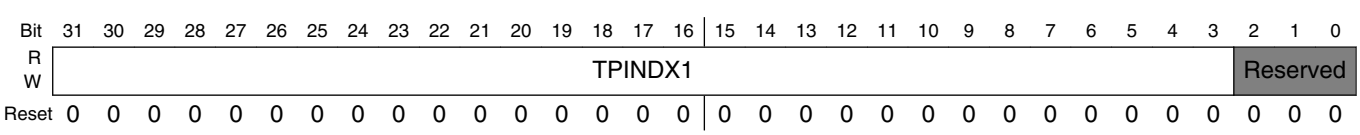
The register value should be set to the desired number of bytes less. The size of buffer1 is the difference between the BUF1IND and BUF0IND.

It is the responsibility of the software to ensure that BUF1IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 34h offset = 21E_0034h



QuadSPI_BUF1IND field descriptions

Field	Description
31–3 TPINDX1	Top index of buffer 1.
Reserved	This field is reserved.

40.14.11 Buffer2 Top Index Register (QuadSPI_BUF2IND)

This register specifies the top index of buffer2, which defines its size. Note that that the 3 LSBs of this register are set to zero - this ensures that the buffer is 64bit aligned as each buffer entry is 64bits long.

The register value should be set to the desired number of bytes less 8. The size of buffer2 is the difference between the BUF2IND and BUF1IND.

It is the responsibility of the software to ensure that BUF2IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 38h offset = 21E_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPINDX2																													Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPI_BUF2IND field descriptions

Field	Description
31–3 TPINDX2	Top index of buffer 2.
Reserved	This field is reserved.

40.14.12 Serial Flash Address Register (QuadSPI_SFAR)

The module automatically translates this address on the memory map to the address on the flash itself. When operating in 24bit mode, only bits 23-0 are sent to the flash, in 32bit mode, bits 27-0 are used with bits 31-28 driven to 0 Refer to Table 40-5 for the mapping between the access mode and the QSPI_SFAR content and to Normal Mode for details about the command triggering and command execution. The software should ensure that the serial flash address provided in the QSPI_SFAR register lies in the valid flash address range as defined in Table 40-5.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: 21E_0000h base + 100h offset = 21E_0100h

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

QuadSPI_SFAR field descriptions

Field	Description
SFADR	Serial Flash Address. The register content is used as byte address for all following IP Commands.

40.14.13 Sampling Register (QuadSPI_SMPR)

The Sampling Register allows configuration of how the incoming data from the external serial flash devices are sampled in the QuadSPI module.

Write:Disabled Mode

Address: 21E_0000h base + 108h offset = 21E_0108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													DDRSMP		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

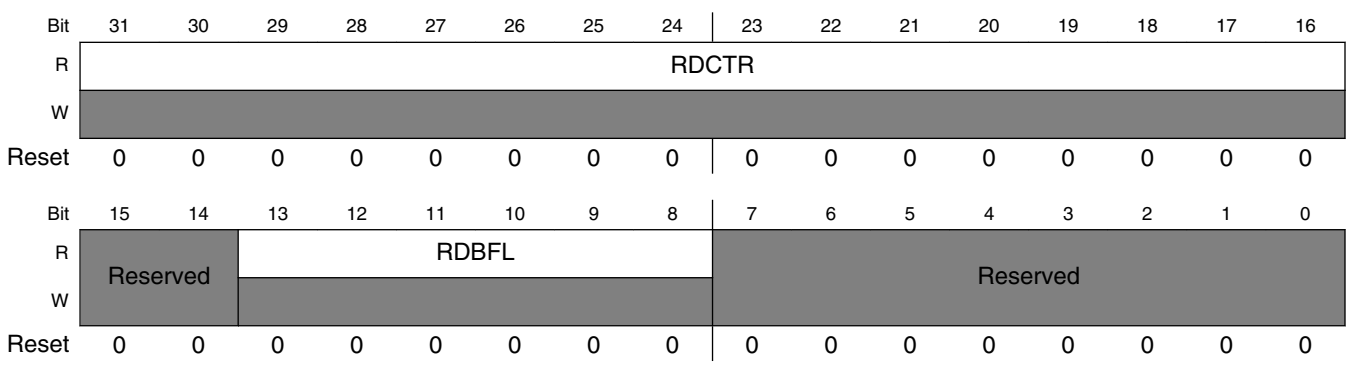
QuadSPI_SMPR field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–16 DDRSMP	DDR Sampling point. Select the sampling point for incoming data when serial flash is executing a DDR instruction. Refer to Input timing in DDR mode with internal sampling , for details on the sampling points.
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–5 SDRSMP	SDR sampling point.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.14.14 RX Buffer Status Register (QuadSPI_RBSR)

This register contains information related to the receive data buffer.

Address: 21E_0000h base + 10Ch offset = 21E_010Ch



QuadSPI_RBSR field descriptions

Field	Description
31–16 RDCTR	Read Counter, indicates how many entries of 4 bytes have been removed from the RX Buffer.For example a value of 0x2 would indicate 8bytes have been removed It is incremented by the number (QSPI_RBCT[WMRK] + 1) on RX Buffer POP event. The RX Buffer can be popped using DMA or pop flag QSPI_FR[RBDF]. The QSPI_RSER[RBDDE] defines which pop has to be done. For further details please refer to AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) and "Data Transfer from the QuadSPI Module Internal Buffers section in Flash Read section.
15–14 Reserved	This field is reserved.
13–8 RDBFL	RX Buffer Fill Level, indicates how many entries of 4 bytes are still available in the RX Buffer.For example a value of 0x2 would indicate 8bytes are available.
Reserved	This field is reserved.

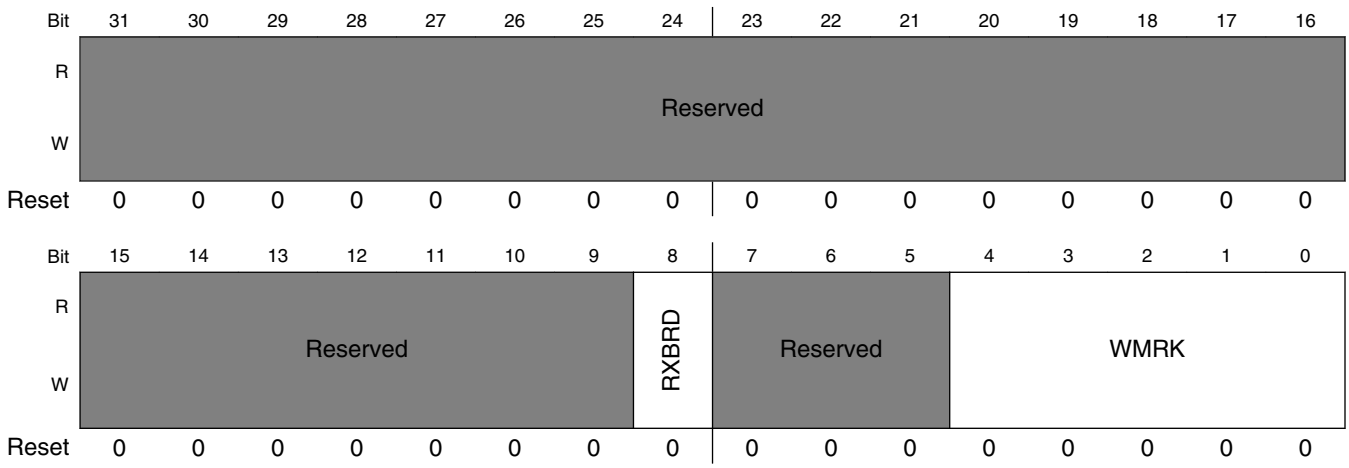
40.14.15 RX Buffer Control Register (QuadSPI_RBCT)

This register contains control data related to the receive data buffer.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: 21E_0000h base + 110h offset = 21E_0110h



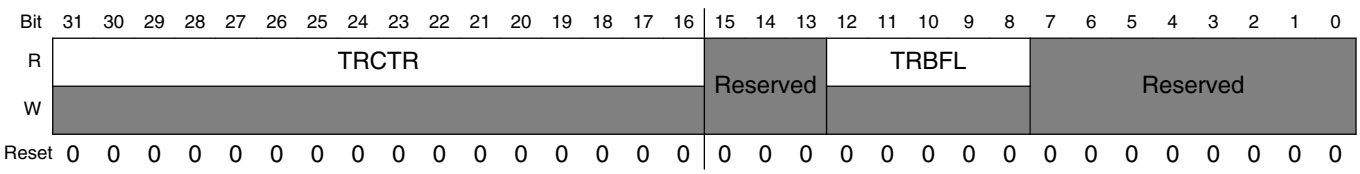
QuadSPI_RBCT field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 RXBRD	RX Buffer Readout: This bit specifies the access scheme for the RX Buffer readout. 0 RX Buffer content is read using the AHB Bus registers QSPI_ARDB0 to QSPI_ARDB31. For details, refer to Exclusive Access to Serial Flash for AHB Commands. 1 RX Buffer content is read using the IP Bus registers QSPI_RBDR0 to QSPI_RBDR31.
7–5 Reserved	This field is reserved.
WMRK	RX Buffer Watermark: This field determines when the readout action of the RX Buffer is triggered. When the number of valid entries in the RX Buffer is equal to or greater than the number given by (WMRK+1) the QSPI_SR[RXWE] flag is asserted.The value should be entered as the number of 4byte entries minus 1. For example a value of 0x0 would set the watermark to 4bytes, 1 to 8bytes, 2 to 12bytes etc. For details, refer to DMA Usage.

40.14.16 TX Buffer Status Register (QuadSPI_TBSR)

This register contains information related to the transmit data buffer.

Address: 21E_0000h base + 150h offset = 21E_0150h



QuadSPI_TBSR field descriptions

Field	Description
31–16 TRCTR	Transmit Counter. This field indicates how many entries of 4 bytes have been written into the TX Buffer by host accesses. It is reset to 0 when a 1 is written into the QSPI_MCR[CLR_TXF] bit. It is incremented on each write access to the QSPI_TBDR register when another word has been pushed onto the TX Buffer. When it is not cleared the TRCTR field wraps around to 0. Refer to TX Buffer Data Register (QuadSPI_TBDR) for details.
15–13 Reserved	This field is reserved.
12–8 TRBFL	TX Buffer Fill Level. The TRBFL field contains the number of entries of 4 bytes each available in the TX Buffer for the QuadSPI module to transmit to the serial flash device.
Reserved	This field is reserved.

40.14.17 TX Buffer Data Register (QuadSPI_TBDR)

The QSPI_TBDR register provides access to the circular TX Buffer of depth 128 bytes. This buffer provides the data written into it as write data for the page programming commands to the serial flash device. Refer to Table 40-14 for the byte ordering scheme. A write transaction on the flash with data size of less than 32 bits will lead to the removal of four data entry from the TX buffer. The valid bits will be used and the rest of the bits will be discarded.

Write:

- *QSPI_SR[TXFULL] = 0*

32-bit write access required

NOTE

There is no limitation on Flash programming size. But the data size written to TX Buffer should be 64 Byte aligned. If flash programming size is not 64 bytes aligned, please write redundant data to TX Buffers. The redundant data will be ignored by QuadSPI controller.

Address: 21E_0000h base + 154h offset = 21E_0154h

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

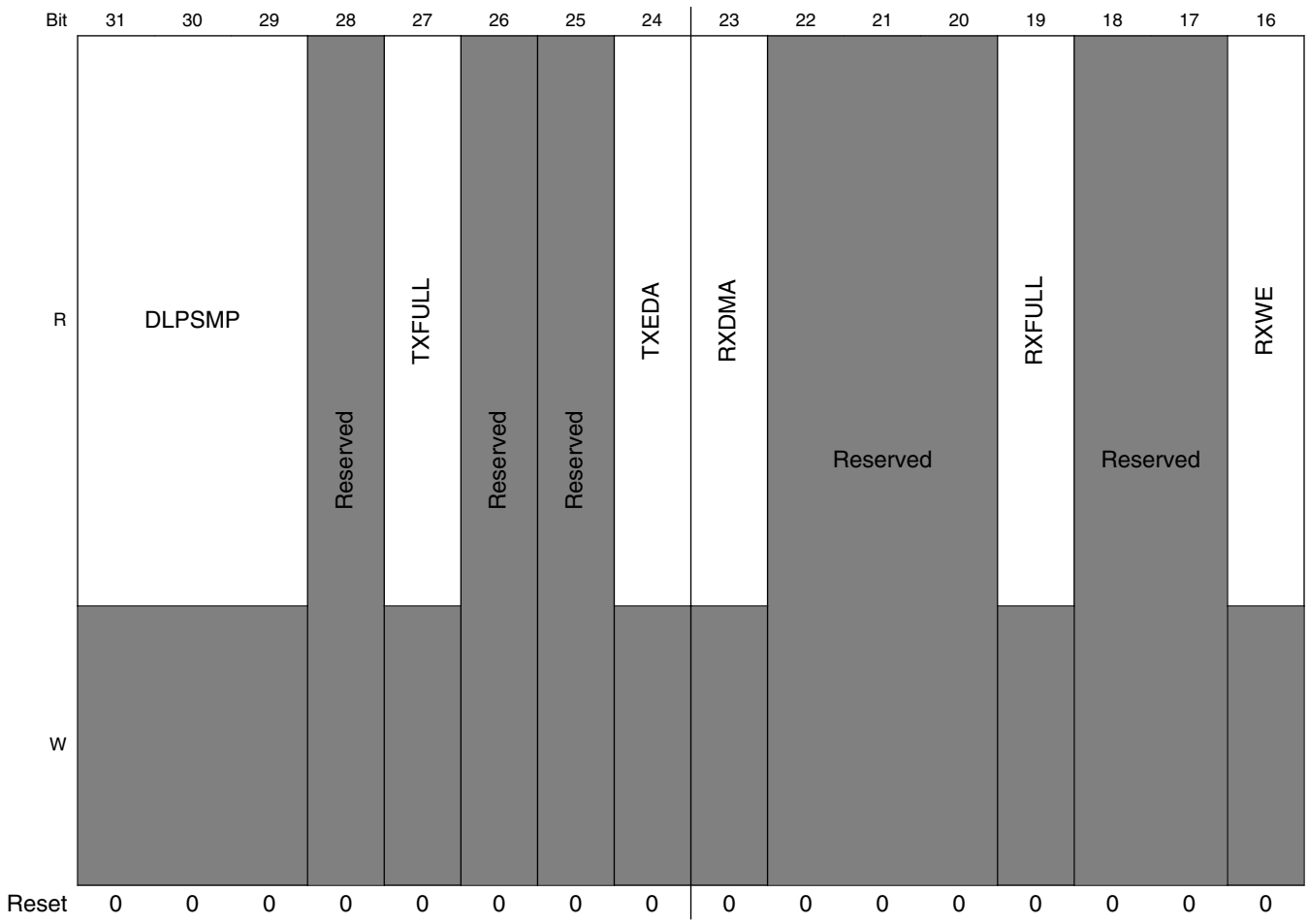
QuadSPI_TBDR field descriptions

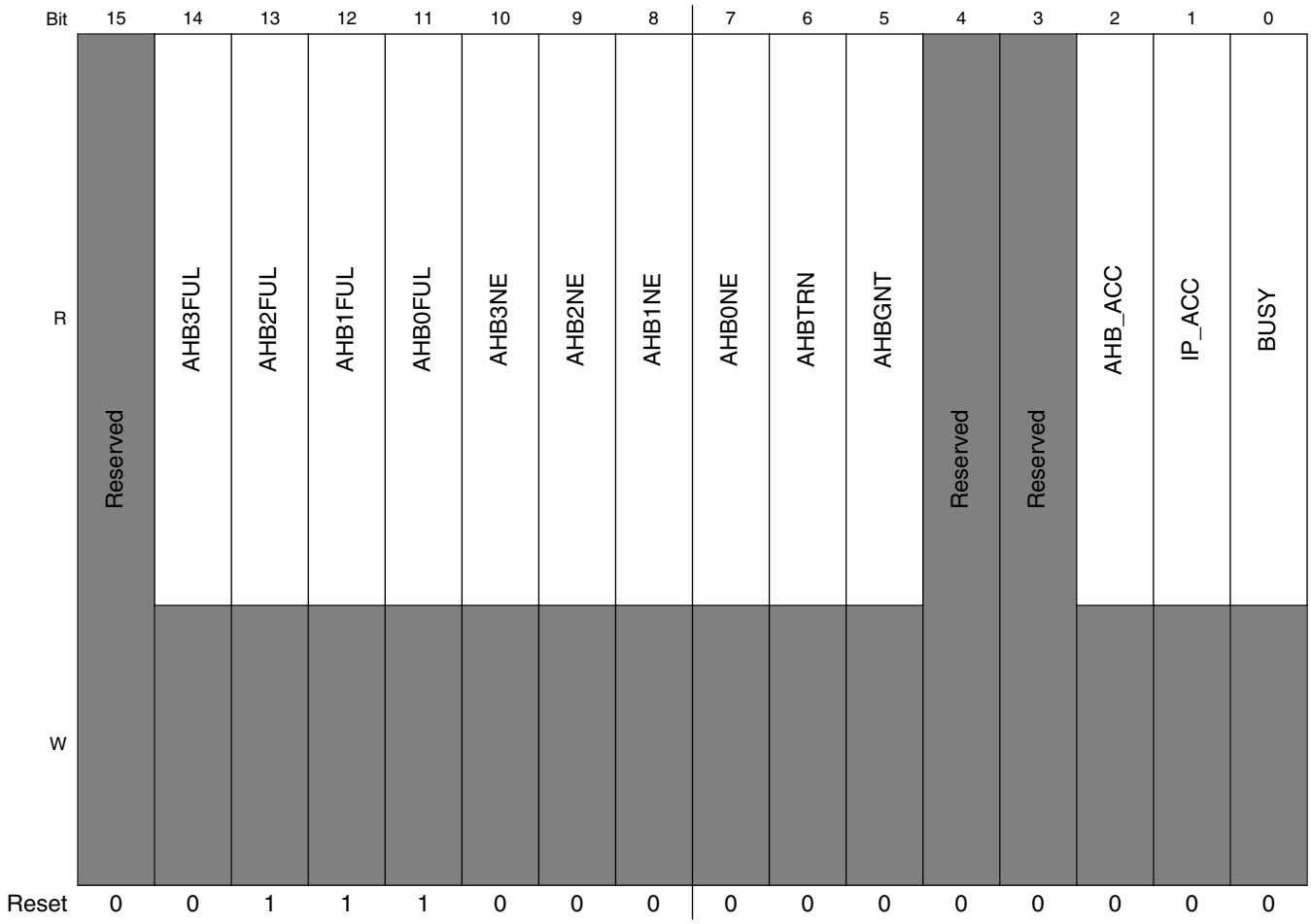
Field	Description
TXDATA	TX Data On write access the data is written into the next available entry of the TX Buffer and the QPSI_TBSR[TRBFL] field is updated accordingly. On a read access, the last data written to the register is returned.

40.14.18 Status Register (QuadSPI_SR)

The QSPI_SR register provides all available status information about SFM command execution and arbitration, the RX Buffer and TX Buffer and the AHB Buffer.

Address: 21E_0000h base + 15Ch offset = 21E_015Ch





QuadSPI_SR field descriptions

Field	Description
31–29 DLPSMP	**NOTE:** Data learning is not implemented on this chip. Data learning pattern sampling point: The sampling point found by the controller with the data learning pattern. • This is used for DDR only. • If the learning fails, this field will return garbage and DLPFF bit will be set. •
28 Reserved	This field is reserved.
27 TXFULL	TX Buffer Full: Asserted when no more data can be stored.
26 Reserved	This field is reserved.
25 Reserved	This field is reserved.
24 TXEDA	Tx Buffer Enough Data Available Asserted when TX Buffer contains enough data for any pop operation to take place. There must be atleast 128bit data available in TX FIFO for any pop operation otherwise QSPI_FR[TBUF] will be set.

Table continues on the next page...

QuadSPI_SR field descriptions (continued)

Field	Description
23 RXDMA	RX Buffer DMA: Asserted when RX Buffer read out via DMA is active i.e DMA is requested or running.
22–20 Reserved	This field is reserved.
19 RXFULL	RX Buffer Full: Asserted when the RX Buffer is full, i.e. that QSPI_RBSR[RDBFL] field is equal to 32.
18–17 Reserved	This field is reserved.
16 RXWE	RX Buffer Watermark Exceeded: Asserted when the number of valid entries in the RX Buffer exceeds the number given in the QSPI_RBCT[WMRK] field.
15 Reserved	This field is reserved.
14 AHB3FUL	AHB 3 Buffer Full: Asserted when AHB 3 buffer is full.
13 AHB2FUL	AHB 2 Buffer Full: Asserted when AHB 2 buffer is full.
12 AHB1FUL	AHB 1 Buffer Full: Asserted when AHB 1 buffer is full.
11 AHB0FUL	AHB 0 Buffer Full: Asserted when AHB 0 buffer is full.
10 AHB3NE	AHB 3 Buffer Not Empty: Asserted when AHB 3 buffer contains data.
9 AHB2NE	AHB 2 Buffer Not Empty: Asserted when AHB 2 buffer contains data.
8 AHB1NE	AHB 1 Buffer Not Empty: Asserted when AHB 1 buffer contains data.
7 AHB0NE	AHB 0 Buffer Not Empty: Asserted when AHB 0 buffer contains data.
6 AHBTRN	AHB Access Transaction pending: Asserted when there is a pending request on the AHB interface. Refer to the AMBA specification for details.
5 AHBGNT	AHB Command priority Granted: Asserted when another module has been granted priority of AHB Commands against IP Commands. For details refer to Command Arbitration.
4 Reserved	This field is reserved.
3 RESERVED	This field is reserved.
2 AHB_ACC	AHB Access: Asserted when the transaction currently executed was initiated by AHB bus.
1 IP_ACC	IP Access: Asserted when transaction currently executed was initiated by IP bus.
0 BUSY	Module Busy: Asserted when module is currently busy handling a transaction to an external flash device.

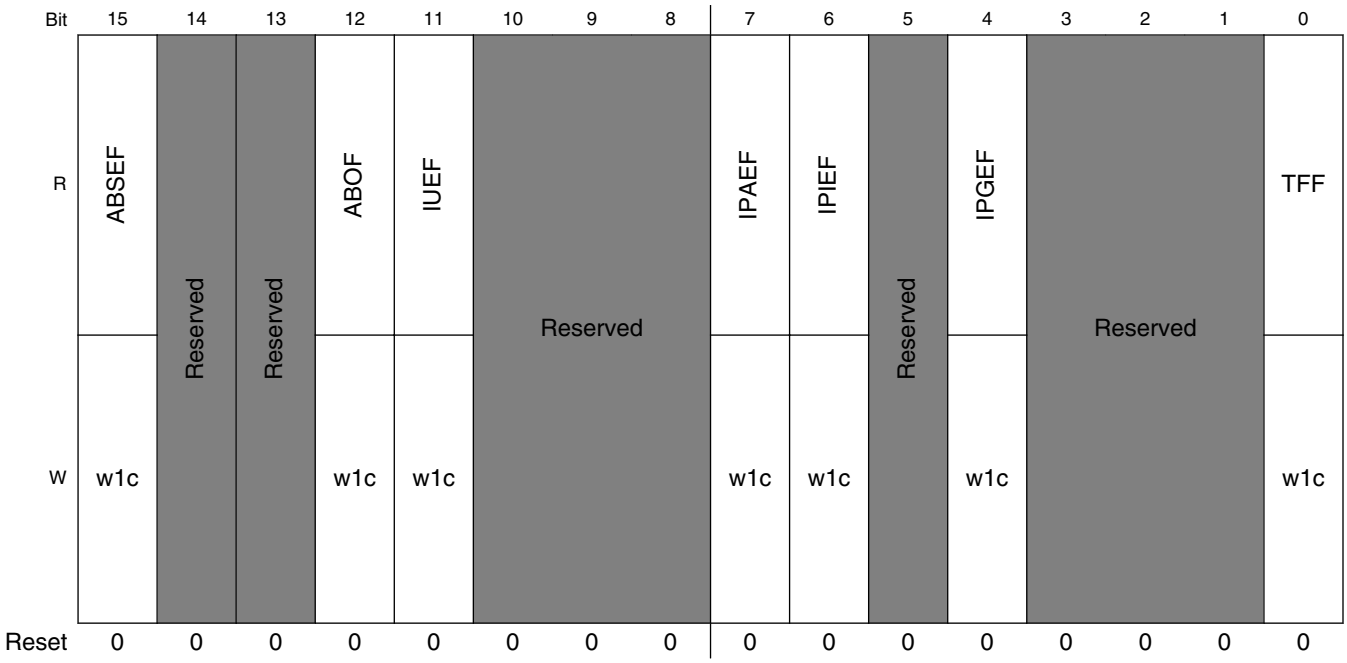
40.14.19 Flag Register (QuadSPI_FR)

The QSPI_FR register provides all available flags about SFM command execution and arbitration which may serve as source for the generation of interrupt service requests. Note that the error flags in this register do not relate directly to the execution of the transaction in the serial flash device itself but only to the behavior and conditions visible in the QuadSPI module.

Write:Enabled Mode

Address: 21E_0000h base + 160h offset = 21E_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLPFF	Reserved	Reserved		TBFF	TBUF	Reserved		ILLINE	Reserved					RBOF	RBDF
W	w1c				w1c	w1c			w1c						w1c	w1c
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0



QuadSPI_FR field descriptions

Field	Description
31 DLPFF	**NOTE:** Data learning is not implemented on this chip. Data Learning Pattern Failure Flag: Set when DATA_LEARN instruction was encountered in a sequence but no sampling point was found for the data learning pattern The controller automatically starts sampling using the value in QSPI_SMPR[DDRSMP].
30 Reserved	This field is reserved.
29–28 Reserved	This field is reserved.
27 TBFF	TX Buffer Fill Flag: Before writing to the TX buffer, this bit should be cleared. Then this bit has to be read back. If the bit is set, the TX Buffer can take more data. If the bit remains cleared, the TX buffer is full. Refer to Tx Buffer Operation for details.
26 TBUF	TX Buffer Underrun Flag: Set when the module tried to pull data although TX Buffer was emptyor the buffer contains less than 128bits of data. The application must ensure that the buffer never goes empty during a transaction expect for the last data fetch. The IP Command leading to the TX Buffer underrun is continued (data sent to the serial flash device is all F in case of valid tx underrun. The application must clear the TX Buffer in response to this event by writing a 1 into the QSPI_MCR[CLR_TXF] bit.
25–24 Reserved	This field is reserved.
23 ILLINE	Illegal Instruction Error Flag: Set when an illegal instruction is encountered by the controller in any of the sequences. Refer to Table 40-12 for a list of legal instructions.
22–18 Reserved	This field is reserved.
17 RBOF	RX Buffer Overflow Flag: Set when not all the data read from the serial flash device could be pushed into the RX Buffer. The IP Command leading to this condition is continued until the number of bytes according to the QSPI_IPCR[IDATSZ] field has been read from the serial flash device.

Table continues on the next page...

QuadSPI_FR field descriptions (continued)

Field	Description
	The content of the RX Buffer is not changed.
16 RBDF	RX Buffer Drain Flag: Will be set if the QuadSPI_SR[RXWE] status bit is asserted. Writing 1 into this bit triggers one of the following actions: • If the RX Buffer has up to QuadSPI_RBCT[WMRK] valid entries then the flag is cleared. • If the RX Buffer has more than QuadSPI_RBCT[WMRK] valid entries and the QuadSPI_RSER[RBDDE] bit is not set (flag driven mode) a RX Buffer POP event is triggered. The flag remains set if the RX Buffer contains more than QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. The flag is cleared if the RX Buffer contains less than or equal to QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. Refer to "Receive Buffer Drain Interrupt or DMA Request" section in Normal Mode Interrupt and DMA Requests, for details.
15 ABSEF	AHB Sequence Error Flag: Set when the execution of an AHB Command is started with an WRITE or WRITE_DDR Command in the sequence pointed to by the QSPI_BUFxCR register Communication with the serial flash device is terminated before the execution of WRITE/WRITE_DDR command by the QuadSPI module. The AHB bus request which triggered this command is answered with an ERROR response.
14 Reserved	Reserved This field is reserved.
13 Reserved	Reserved This field is reserved.
12 ABOF	AHB Buffer Overflow Flag: Set when the size of the AHB access exceeds the size of the AHB buffer. This condition can occur only if the QSPI_BUFxCR[ADATSZ] field is programmed incorrectly. The AHB Command leading to this condition is continued until the number of entries according to the QSPI_BUFxCR[ADATSZ] field has been read from the serial flash device. The content of the AHB Buffer is not changed.
11 IUEF	IP Command Usage Error Flag: Set when in parallel flash mode the execution of an IP Command is started and the sequence pointed to by the sequence ID contains a WRITE or a WRITE_DDR command. Refer to Table 40-12 table for the related commands. Communication with the serial flash device is terminated before the execution of WRITE/WRITE_DDR command by the QuadSPI module.
10–8 Reserved	This field is reserved.
7 IPAEF	IP Command Trigger during AHB Access Error Flag. Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHB_ACC] bit is set. Any command leading to the assertion of the IPAEF flag is ignored.
6 IPIEF	IP Command Trigger could not be executed Error Flag. Set when the QSPI_SR[IP_ACC] bit is set (i.e. an IP triggered command is currently executing) and any of the following conditions occurs: • Write access to the QSPI_IPCR register. Any command leading to the assertion of the IPIEF flag is ignored • Write access to the QSPI_SFAR register. • Write access to the QSPI_RBCT register.
5 Reserved	This field is reserved.

Table continues on the next page...

QuadSPI_FR field descriptions (continued)

Field	Description
4 IPGEF	IP Command Trigger during AHB Grant Error Flag: Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHBGNT] bit is set. Any command leading to the assertion of the IPGEF flag is ignored.
3–1 Reserved	This field is reserved.
0 TFF	IP Command Transaction Finished Flag: Set when the QuadSPI module has finished a running IP Command. If an error occurred the related error flags are valid, at the latest, in the same clock cycle when the TFF flag is asserted.

1. QSPI_BUFxCR implies anyone of QSPI_BUF0CR/QSPI_BUF1CR/QSPI_BUF2CR/QSPI_BUF3CR

40.14.20 Interrupt and DMA Request Select and Enable Register (QuadSPI_RSER)

The QuadSPI_RSER register provides enables and selectors for the interrupts in the QuadSPI module.

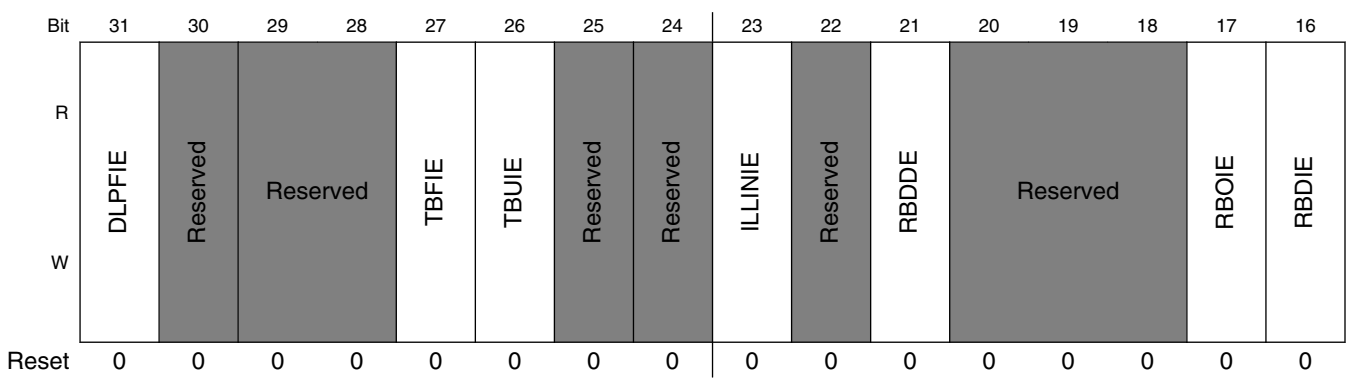
NOTE

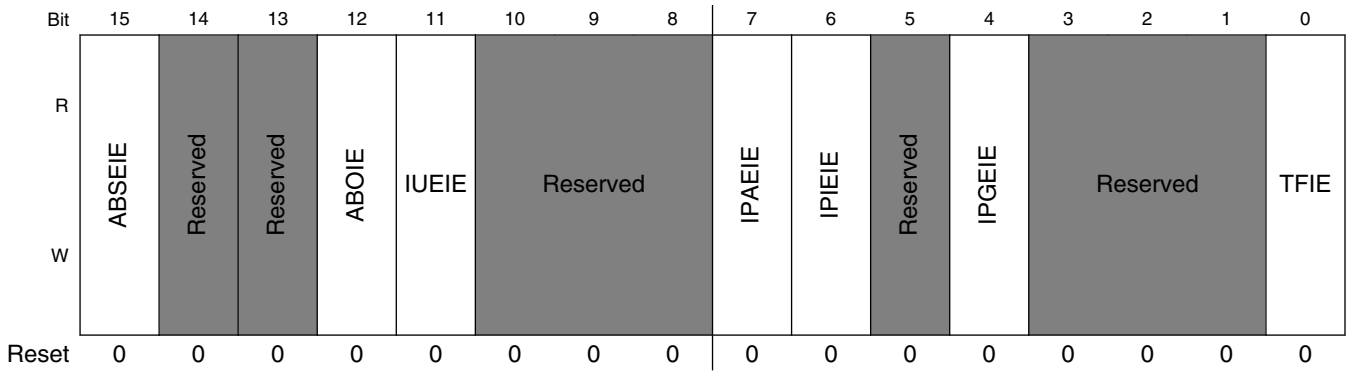
Each flag of the QuadSPI_FR register enabled as source for an interrupt prevents the QuadSPI module from entering Stop Mode or Module Disable Mode when this flag is set.

>

Write:Anytime

Address: 21E_0000h base + 164h offset = 21E_0164h





QuadSPI_RSER field descriptions

Field	Description
31 DLPFIE	**NOTE:** Data learning is not implemented on this chip. Data Learning Pattern Failure Interrupt enable . Triggered by DLPFF flag in QSPI_FR register 0 No DLPFF interrupt will be generated 1 DLPFF interrupt will be generated
30 Reserved	This field is reserved.
29–28 RESERVED	This field is reserved.
27 TBFIE	TX Buffer Fill Interrupt Enable 0 No TBFF interrupt will be generated 1 TBFF interrupt will be generated
26 TBUIE	TX Buffer Underrun Interrupt Enable 0 No TBUF interrupt will be generated 1 TBUF interrupt will be generated
25 Reserved	This field is reserved.
24 Reserved	This field is reserved.
23 ILLINIE	Illegal Instruction Error Interrupt Enable. Triggered by ILLINE flag in QSPI_FR 0 No ILLINE interrupt will be generated 1 ILLINE interrupt will be generated
22 Reserved	This field is reserved.
21 RBDDE	RX Buffer Drain DMA Enable: Enables generation of DMA requests for RX Buffer Drain. When this bit is set DMA requests are generated as long as the QSPI_SR[RXWE] status bit is set. 0 No DMA request will be generated 1 DMA request will be generated
20–18 Reserved	This field is reserved.

Table continues on the next page...

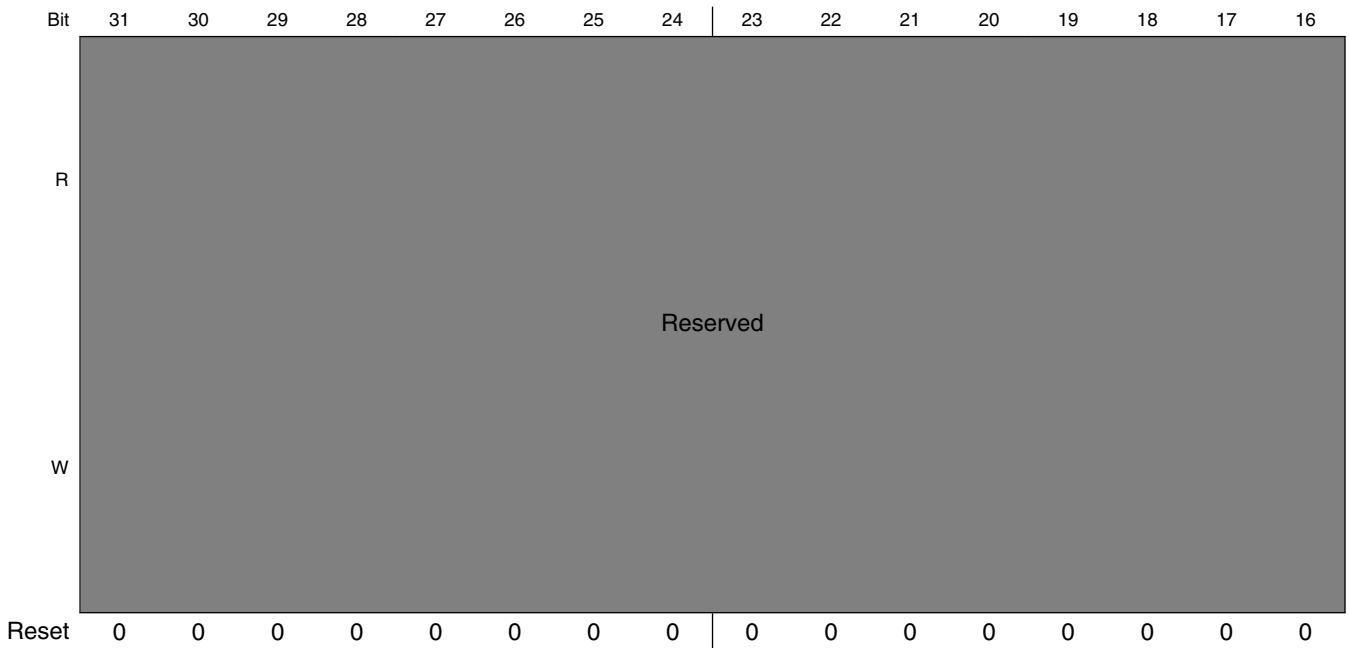
QuadSPI_RSER field descriptions (continued)

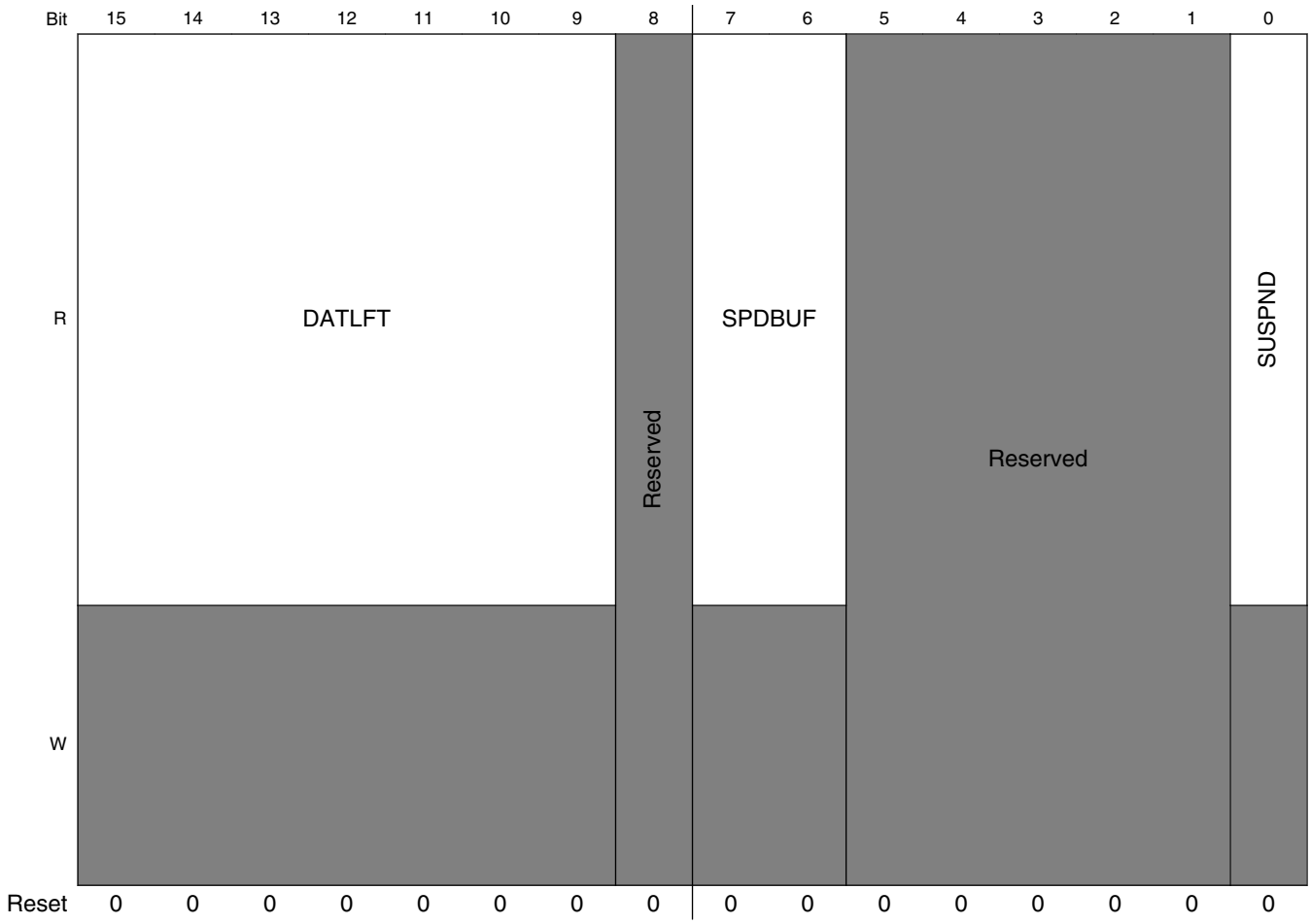
Field	Description
17 RBOIE	RX Buffer Overflow Interrupt Enable 0 No RBOF interrupt will be generated 1 RBOF interrupt will be generated
16 RBDIE	RX Buffer Drain Interrupt Enable: Enables generation of IRQ requests for RX Buffer Drain. When this bit is set the interrupt is asserted as long as the QuadSPI_SR[RBDF] flag is set. 0 No RBDF interrupt will be generated 1 RBDF Interrupt will be generated
15 ABSEIE	AHB Sequence Error Interrupt Enable: Triggered by ABSEF flags of QSPI_FR 0 No ABSEF interrupt will be generated 1 ABSEF interrupt will be generated
14 Reserved	Reserved This field is reserved.
13 Reserved	Reserved This field is reserved.
12 ABOIE	AHB Buffer Overflow Interrupt Enable 0 No ABOF interrupt will be generated 1 ABOF interrupt will be generated
11 IUEIE	IP Command Usage Error Interrupt Enable 0 No IUEF interrupt will be generated 1 IUEF interrupt will be generated
10–8 Reserved	This field is reserved.
7 IPAEIE	IP Command Trigger during AHB Access Error Interrupt Enable 0 No IPAEF interrupt will be generated 1 IPAEF interrupt will be generated
6 IPIEIE	IP Command Trigger during IP Access Error Interrupt Enable 0 No IPIEF interrupt will be generated 0 IPIEF interrupt will be generated
5 Reserved	This field is reserved.
4 IPGEIE	IP Command Trigger during AHB Grant Error Interrupt Enable 0 No IPGEF interrupt will be generated 1 IPGEF interrupt will be generated
3–1 Reserved	This field is reserved. Reserved.
0 TFIE	Transaction Finished Interrupt Enable 0 No TFF interrupt will be generated 1 TFF interrupt will be generated

40.14.21 Sequence Suspend Status Register (QuadSPI_SPNDST)

The sequence suspend status register provides information specific to any suspended sequence. An AHB sequence may be suspended when a high priority AHB master makes an access before the AHB sequence completes the data transfer requested.

Address: 21E_0000h base + 168h offset = 21E_0168h





QuadSPI_SPNDST field descriptions

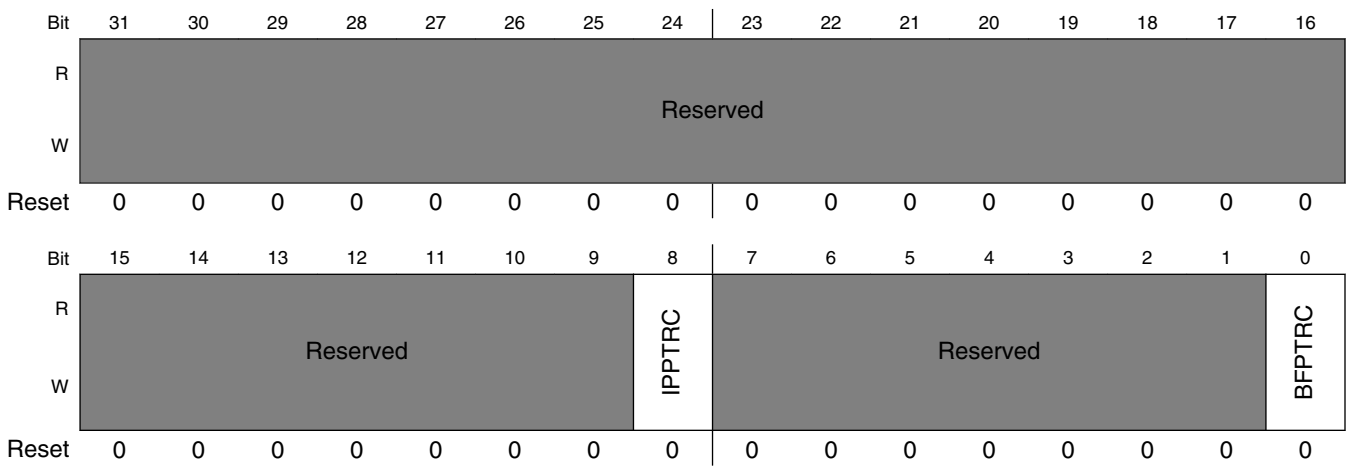
Field	Description
31–16 Reserved	This field is reserved.
15–9 DATLFT	Data left: Provides information about the amount of data left to be read in the suspended sequence. Valid only when SUSPND is set to 1'b1. Value in terms of 64 bits or 8 bytes
8 Reserved	This field is reserved.
7–6 SPDBUF	Suspended Buffer: Provides the suspended buffer number. Valid only when SUSPND is set to 1'b1
5–1 Reserved	This field is reserved.
0 SUSPND	When set, it signifies that a sequence is in suspended state

40.14.22 Sequence Pointer Clear Register (QuadSPI_SPTRCLR)

The sequence pointer clear register provides bits to reset the IP and Buffer sequence pointers. The sequence pointer contains the index of which instruction within the LUT entry is to be executed next. For example, if the LUT entry ends on a JMP_ON_CS value of 2, the index will be stored as 2.

The software should reset the sequence pointers whenever the sequence ID is changed by updating the SEQID field in QSPI_IPCR or QSPI_BFGENCR.

Address: 21E_0000h base + 16Ch offset = 21E_016Ch



QuadSPI_SPTRCLR field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 IPPTRC	IP Pointer Clear: 1: Clears the sequence pointer for IP accesses as defined in QuadSPI_IPCR
7–1 Reserved	This field is reserved. Reserved.
0 BFPTRC	Buffer Pointer Clear: 1: Clears the sequence pointer for AHB accesses as defined in QuadSPI_BFGENCR.

40.14.23 Serial Flash A1 Top Address (QuadSPI_SFA1AD)

The QSPI_SFA1AD register provides the address mapping for the serial flash A1.The difference between QSPI_SFA1AD[TPADA1] and QSPI_AMBA_BASE defines the size of the memory map for serial flash A1.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 180h offset = 21E_0180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADA1																Reserved																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPI_SFA1AD field descriptions

Field	Description
31–10 TPADA1	Top address for Serial Flash A1. In effect, TPADxx is the first location of the next memory.
Reserved	This field is reserved.

40.14.24 Serial Flash A2 Top Address (QuadSPI_SFA2AD)

The QSPI_SFA2AD register provides the address mapping for the serial flash A2.The difference between QSPI_SFA2AD[TPADA2] and QSPI_SFA1AD[TPADA1] defines the size of the memory map for serial flash A2.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 184h offset = 21E_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADA2																Reserved																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPI_SFA2AD field descriptions

Field	Description
31–10 TPADA2	Top address for Serial Flash A2. In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

40.14.25 Serial Flash B1Top Address (QuadSPI_SFB1AD)

The QSPI_SFB1AD register provides the address mapping for the serial flash B1.The difference between QSPI_SFB1AD[TPADB1] and QSPI_SFA2AD[TPADA2] defines the size of the memory map for serial flash B1.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 188h offset = 21E_0188h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADB1																Reserved																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPI_SFB1AD field descriptions

Field	Description
31–10 TPADB1	Top address for Serial Flash B1.In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

40.14.26 Serial Flash B2Top Address (QuadSPI_SFB2AD)

The QSPI_SFB2AD register provides the address mapping for the serial flash B2.The difference between QSPI_SFB2AD[TPADB2] and QSPI_SFB1AD[TPADB1] defines the size of the memory map for serial flash B2.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 21E_0000h base + 18Ch offset = 21E_018Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADB2																Reserved																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPI_SFB2AD field descriptions

Field	Description
31–10 TPADB2	Top address for Serial Flash B2. In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

40.14.27 RX Buffer Data Register (QuadSPI_RBDR*n*)

The QuadSPI_RBDR registers provide access to the individual entries in the RX Buffer. Refer to Table 40-14 for the byte ordering scheme.

QuadSPI_RBDR0 corresponds to the actual position of the read pointer within the RX Buffer. The number of valid entries available depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QuadSPI_RBDR0 to QuadSPI_RBDR31.

Example 2, RX Buffer filled with 5 valid words: RX Buffer fill level QuadSPI_RBSR[RDBFL] is 5. In this case an access to QuadSPI_RBDR4 provides the last valid entry.

Any access beyond the range of valid RX Buffer entries provides undefined results.

Address: 21E_0000h base + 200h offset + (4d × i), where i=0d to 31d

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

QuadSPI_RBDR*n* field descriptions

Field	Description
RXDATA	RX Data. The RXDATA field contains the data associated with the related RX Buffer entry. Data format and byte ordering is given in Byte Ordering of Serial Flash Read Data .

40.14.28 LUT Key Register (QuadSPI_LUTKEY)

The LUT Key register contains the key to lock and unlock the Look-up-table. Refer to Look-up Table for details.

Write:Anytime

Address: 21E_0000h base + 300h offset = 21E_0300h

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

QuadSPI_LUTKEY field descriptions

Field	Description
KEY	The key to lock or unlock the LUT. The KEY is 0x5AF05AF0. The read value is always 0x5AF05AF0

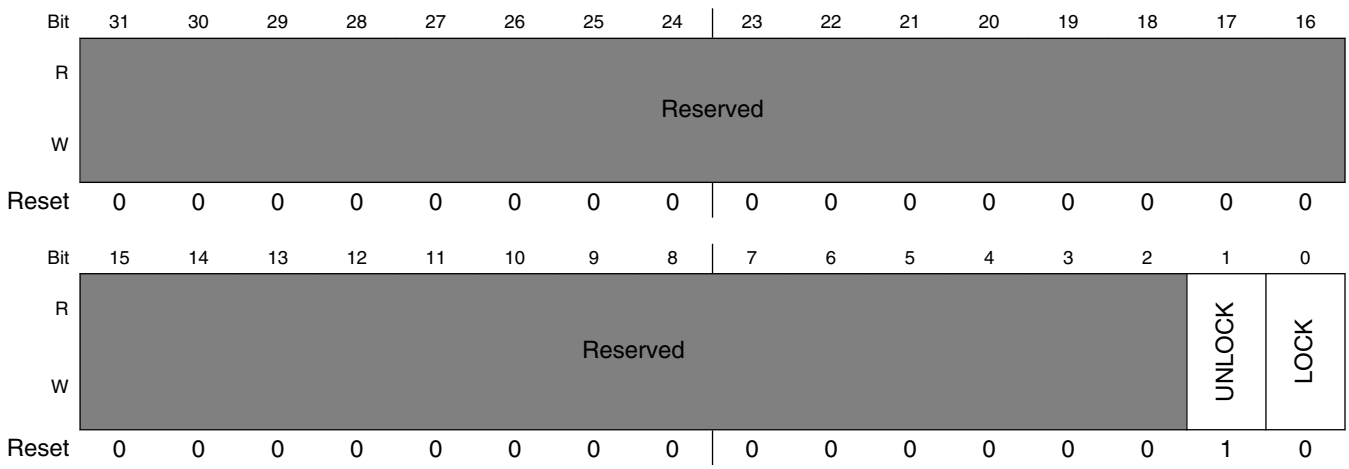
40.14.29 LUT Lock Configuration Register (QuadSPI_LCKCR)

The LUT lock configuration register is used along with QSPI_LUTKEY register to lock or unlock the LUT. This register has to be written immediately after QSPI_LUTKEY register for the lock or unlock operation to be successful. Refer to Look-up Table for details. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Write:Just after writing the LUT Key Register

(QSPI_LUTKEY)

Address: 21E_0000h base + 304h offset = 21E_0304h



QuadSPI_LCKCR field descriptions

Field	Description
31–2 Reserved	This field is reserved.
1 UNLOCK	Unlocks the LUT when the following two conditions are met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key
0 LOCK	Locks the LUT when the following condition is met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key

40.14.30 Look-up Table register (QuadSPI_LUT0)

The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

Write:Once the LUT is unlocked

Address: 21E_0000h base + 310h offset = 21E_0310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
Reset	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR0						PAD0		OPRND0							
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1

QuadSPI_LUT0 field descriptions

Field	Description
31–26 INSTR1	Instruction 1
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
OPRND0	Operand for INSTR0.

40.14.31 Look-up Table register (QuadSPI_LUT1)

The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

Write:Once the LUT is unlocked

Address: 21E_0000h base + 314h offset = 21E_0314h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
Reset	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR0						PAD0		OPRND0							
Reset	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0

QuadSPI_LUT1 field descriptions

Field	Description
31–26 INSTR1	Instruction 1
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
OPRND0	Operand for INSTR0.

40.14.32 Look-up Table register (QuadSPI_LUT*n*)

The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

Write:Once the LUT is unlocked

Address: 21E_0000h base + 318h offset + (4d × i), where i=0d to 61d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR0						PAD0		OPRND0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPI_LUT*n* field descriptions

Field	Description
31–26 INSTR1	Instruction 1
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
OPRND0	Operand for INSTR0.

Chapter 41
ROM Controller with Patch (ROMC)

41.1 Overview

The Read Only Memory Controller with ROM Patch (ROMC) acts as an interface between the ARM advanced high-performance bus (AHB - Lite) and the Read Only Memory. The ROMC consists of a ROM Controller and a ROM Patch. The ROM Patch is used to either patch code routines or fix data tables in the ROM area. There is an IP Bus interface to access the ROM Patch Registers . The figure below depicts the main functional sub-blocks of the ROMC.

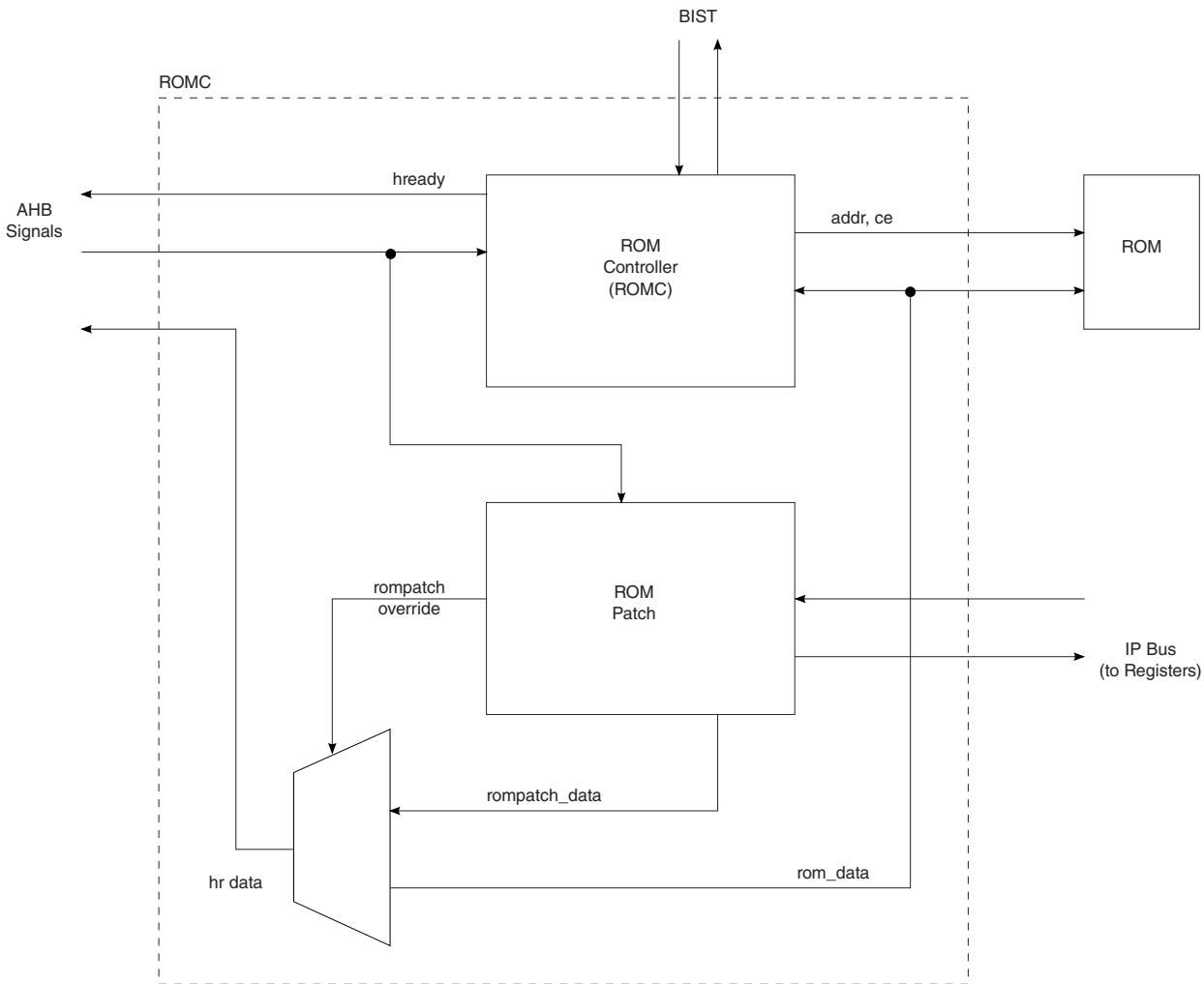


Figure 41-1. ROMC Block Diagram

41.1.1 Features

- Supports ROM size ranges from 16 Kbyte up to 4 Mbyte with increments of 1 Kbyte
- Supports opcode patching for a maximum of 16 different addresses in 4 Mbytes of ROM space
- Supports one-word data fixes for a max of 8 memory locations in 4 Mbytes of ROM space
- Supports patching of the Reset Vector (at 0x0000_0000) to allow external booting

41.1.2 Modes of Operation

There are two modes of operation: normal mode and BIST mode. In normal mode, the ROMC ensures correct reads from the ROM, assuming the memory complies with the characteristics and requirements for which the ROMC was designed.

41.1.2.1 Low Power Mode

There are two clock enables that are used to switch off parts of the ROMC logic when inactive. The first clock enable is used to disable the ROM Controller when the master connected to the AHB interface is not initiating a read to the ROM. The second clock enable is used to disable the registers used to program the ROM patch feature when the registers are not being accessed.

41.2 Clocks

The table found here describes the clock sources for ROMCP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 41-1. ROMCP Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	System / bus clock
hclk_reg	ipg_clk_root	System access clock
96krom_CLK	ahb_clk_root	ROM clock

41.3 Memory Map

41.3.1 ROM Memory Map in detail

The ROMC supports ROM sizes with a range of 16 Kbyte to 4 Mbyte with an increment of 1 Kbyte. The 16 Kbyte lower limit was chosen because the minimum size of security code on an ARM platform is approximately 16 Kbyte of code, which is only accessible in supervisor mode. Note that it is the MMU that controls whether any region of memory is secure.

The exception vectors must be secured as well, and must be put in the same area as the security code. Since they must reside at address 0x0000_0000, the entire 16 Kbyte of ROM which can only be accessible in supervisor mode is located at the very beginning of the platform memory map.

If the user chooses not to use the security code, a memory size smaller than 16 Kbyte can be connected to the platform (minimum of 1 Kbyte). The MMU can be programmed to allow any kind of access into this memory. However, if the ROM size is less than 16 Kbyte, memory aliasing will occur for all invalid addresses greater than the memory size but within the 16 Kbyte of space.

For ROM sizes bigger than 16 Kbyte, the rest of its physical size resides at the address starting at 0x0040_4000 (4 M+16 Kbyte) going up to [0x0040_4000 + (mem. size - 16Kbyte)]. .

41.4 Functional Description

This section is divided up into the ROM Controller Functional Description and the ROMC functional description.

41.4.1 ROM Controller (ROMC) Functional Description

41.4.1.1 Functionality overview

The ROMC serves two main functions. First, as an interface between the AHB-Lite bus on an ARM platform and the ROM. Second, it drives and receives several signals for the BIST engine. In normal mode of operation, the ROMC monitors the AHB-Lite for memory access requests and performs the memory operation to the ROM.

The ROMC includes the option to wait state all accesses from either the ARM or non-ARM masters to ROM in the event that timing requirements will not allow single HCLK clock cycle reads. If a wait state is required, the static inputs rom_wait_arm or rom_wait_alt_mstr can be set to 1 and accesses will take two HCLK clock cycles. If wait states are not required, rom_wait_arm or rom_wait_alt_mstr can be set to 0 and accesses will take one HCLK clock cycle to complete.

41.4.2 ROMC Functional Description

41.4.2.1 ROMC Disabling

All the bits in the ROMC_ROMPATCHENL register are cleared on Reset, disabling all the address comparators. Once the comparators have been enabled, the ROMC functions of data fixing and opcode patching can be quickly disabled by setting the DIS bit in the ROMC_ROMPATCHCNTL register. This bit is used to enable secure operations in which patching functions need to be disabled. This bit is cleared on Reset.

41.4.2.2 ROMC Event Priority

The ROMC has a total of 16 address comparators. The first 8 (0 through 7) comparators can be programmed for the data fixing function (through the 8 data fix enable bits in the ROMC_ROMPATCHCNTL register) while the rest are for opcode patching by default. This allows for potential multiple matching events involving both data fixing and opcode patch types. In these cases the ROMC assigns the highest priority to a data fixing event.

For example, if the ROMC is set up to data fix a certain address with comparator 4 and also opcode patch the same address with comparator 7, it will let comparator 4 have higher priority in indicating a match, and data from ROMC_ROMPATCHD4 will be put on the rompatch_romc_hrdata bus as the override value.

If multiple address matches of the same type level occur concurrently, then the ROMC will choose the source number based on the one with the highest source number. For example, the ROMC is setup to data fix the same location with address comparators 4 and 7, then address comparator 7 will have higher priority in indicating a match, and the value from ROMC_ROMPATCHD7 will be put on the rompatch_romc_hrdata bus as the orverride value. The same priority applies for an opcode patch event, except the override data is in the form of an SWI instruction with the comment field set to the source number with the highest priority.

41.4.2.3 Data Fixing

The data fixing feature allows ROM data to be updated by direct replacement when it is being read. This data usually originates from data tables, but can include ARM instructions. To enable data fixing on a certain address, this address value is written in to one of the first eight (0 through 7) of ROMC_ROMPATCHAxx registers and the same numbered bit set in the ROMC_ROMPATCHENL and ROMC_ROMPATCHCNTL registers. The data to be used for replacement is placed in the corresponding ROMC_ROMPATCHDxx.

The ROMC looks for a read access to ROM (either code fetch or data load) by snooping the AHB interface for read transactions. The address is compared with the values stored in the ROMC_ROMPATCHAxx[22:2] registers. If a match occurs from one of the comparators, the ROMC places the value in the corresponding ROMC_ROMPATCHDxx register on the read data bus by overriding the read data coming from the actual ROM (see the mux in Figure 41-1). The value on the read data bus is maintained until hready is asserted to terminate the access. In data fixing, the entire word is replaced so if a byte or half-word access occurs on a "data fix" location, the entire data word is replaced. The word being replaced is word aligned. (The two LSBs of the matching ROMC_ROMPATCHAxx are ignored in the data fix operation.)

41.4.2.4 Opcode Patching

The opcode patch feature provides the ARM core a mechanism to fetch updated versions of code routines that were originally programmed in ROM. This patching mechanism makes use of the SWI (software interrupt instruction) and a table of function pointers residing in writable memory. The opcode being patched is replaced with a SWI instruction by the ROMC. Subsequent processing of the SWI reads from a function pointers table to obtain the address of the replacement code. Execution resumes with this code patch.

To enable opcode patching of a certain address, this address value is written into one of the ROMPATCHAxx registers and the corresponding bit set in the ROMPATCHENL to enable the associated comparator. The register's LSB (ROMC_ROMPATCHxx[0]) should be set if THUMB mode patching is in effect for this address. The ROMC identifies a ROM read access by snooping the AHB interface. The address is compared with the values stored in the ROMC_ROMPATCHAxx[22:2] registers. If a match occurs from one of the comparators, the ROMC generates the opcode of a software interrupt (SWI) instruction with the comment field containing the number of the matching address comparator. This opcode and comment is placed on the read data bus until hready is asserted by the ROM controller to terminate the read access.

The type of SWI generated, (that is, either ARM or THUMB), is determined by the LSB of the ROMC_ROMPATCHAxx register associated with the opcode patch. This bit is cleared for ARM mode (32 bits). The ROMC generates a 32-bit SWI (opcode field is 0xEF, occupying bits [31:24] of the word), with the least significant 5 bits of the 24-bit comment field (bits [23:0]) containing the number of the matching address comparator. The rest of the comment field is filled with zeros. This means that the ROMC will use 16 of the 16777216 possible software interrupts. The ROMC overrides the read data from the ROM.

If the LSB of the matching ROMC_ROMPATCHAxx register is set, the opcode patch is in THUMB mode (16 bits or half word). The ROMC generates a 16-bit SWI instruction (opcode field is 0xDF, occupying bits [15:8] of the half word) with the least significant 5 bits of the 8-bit comment field containing with the source number of the address comparator. The rest of the comments field is filled with zeros. This means that the ROMC will use 16 of the 256 possible software interrupts. The ROMC puts this 16 bit SWI instruction value on the proper half of the rompatch_romc_hrdata bus. The other half is zeroed out. Which half of the bus contains the SWI opcode and comment depends on the mode (Big Endian or Little Endian) and the bit 1 of the matching ROMC_ROMPATCHAxx register. In Little Endian mode, the lower half is bits {15:0] and the upper half is bits {31:16]. The order is reversed in Big Endian mode.

In Little Endian mode (bigend signal negated), if bit 1 of the matching ROMC_ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the lower 16 bits of the read data bus and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten by the ROMC data. If ROMC_ROMPATCHAxx[1] is set (upper half word selected), the SWI instruction is put on the upper 16 bits of the read data bus and the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten.

In Big Endian mode (bigend asserted), if bit 1 of the matching ROMC_ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the upper 16 bits of the read data bus while the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten. If ROMC_ROMPATCHAxx[1] is set (upper word selected), the SWI instruction is put on the lower 16 bits and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten.

The eventual execution of the SWI causes the ARM to save the CPSR in SPSR_SVC, the address of the next instruction after the SWI in R14_SVC, enter Supervisor mode, and fetch the SWI vector at 0x8, which then takes it to a handler for further processing as described in the next section.

41.4.2.4.1 Typical Software Response to Opcode Patch

When the SWI handler executes it needs to determine whether the SWI was generated by the ROMC. This is done by loading the SWI instruction and extracting its comment field. The state of the ARM core (ARM or THUMB) when the SWI was executed dictates whether to load the instruction word (ARM) or half word (THUMB). This state information can be determined by testing the T bit (bit 5) of the SPSR. If it's set, the execution was in THUMB mode.

By convention, if the comment field of the SWI is greater than 16, the software interrupt was initiated by software (i.e. an operating system call), and a branch is taken to the appropriate handler routine for further processing. If the comment field is less than 16, the SWI was generated by the ROMC performing a code patch operation. In this case, the software then reads from a table of function pointers, using the value in the SWI comment field as the index into the table. The value that is read is the address of the code patch. This value is loaded into the PC to begin the execution of the code patch. The following code segment illustrates a typical handling of the SWI.

```
stmfd        sp!, {r0-r1,lr}       @ push register onto SWI stack
mrs         r0, spsr        @ get saved status register
tst          r0, #0x20     @ check if call was in THUMB mode
ldrneh      r0, [lr,#-2]       @ yes: load opcode half-word and
bicne       r0, r0, #0xff00   @ yes: extract THUMB comment
ldreq       r0, [lr,#-4]       @ no: load opcode word and
biceq       r0, r0, #0xff000000 @ no: extract ARM comment
                    @ now r0 has comment field
cmp         r0, #16      @ compare to 16 (maximum for ROMC)
ldrlt          lr, =rompatch_tbl_ptr @ < 16: get top of current ROMC
                    @ table; global variable which is
                    @ changeable per context
ldrlt         r1, [lr, r0, lsl #2]   @ < 16: read function pointer from
                    @ table assumed an array of pointers
                    @ patch functions
strlt          r1, [sp, #8]     @ < 16: store function pointer onto
                    @ stack in position of link register
ldmltfd       sp!, {r0-r1,pc}^      @ < 16: "fake" return from SWI, will
                    @ vector core to appropriate patch
                    @ function and set core back to previous
                    @ mode of operating
ldr          r1, =swi_hdler   @ >= 16: pointer to standard SWI
                    @ handler
mov         lr, pc         @ >= 16: set link register
bx          r1          @ >= 16: jump to standard SWI
                    @ handler
ldmfd        sp!, {r0-r1,pc}^      @ >= 16: pop registers from stack
```

41.4.2.5 External Boot Feature

Following a Reset event, the ARM issues an instruction fetch of the Reset Vector from address 0x0. This instruction, normally residing in ROM is usually a branch to a Reset handler or boot code which also normally resides in ROM. The ROMC external boot feature allows the bypassing of this code, using a different boot code residing perhaps in external memory.

This feature uses the data fix mechanism and works as follows: if the boot_int signal is negated when a Reset event occurred, the ROMC will perform a data fix of the Reset Vector at 0x0 with the following instruction (opcode 0xE59FF00C):

```
ldr         pc,[pc,#12]      @ read 0x0000_0014 for reset_vector
```

The value of PC when this instruction is executed is 8 so that a PC relative offset of 12 makes the source address 20 or 0x14. When this instruction executes, the ARM core reads from address 0x0000_0014, triggering a ROMC data fix operation which places the

value taken from the external boot address on the read data bus, with the two LSBs zeroed out. This value is returned to the ARM to be placed in the PC causing code fetch and execution to start from that address.

41.4.2.6 Alternate Masters and ROMC

The ROMC sits on the AHB bus of the internal ROM (ROMC). This means that the ROMC can modify values on the read data bus going to the master. Therefore, any master which reads an opcode patched or data patched location will read patched data.

41.5 ROMCP Memory Map/Register Definition

All registers are accessible through an IP Bus and can only be accessed in privileged mode. These registers can only be written with 32-bits stores and are clocked by hclk_reg.

The ROMC register placement was originated from the AWPT design used in the ARM7 platform of Neptune

ROMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21A_C0D4	ROMC Data Registers (ROMC_ROMPATCH0D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0D8	ROMC Data Registers (ROMC_ROMPATCH1D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0DC	ROMC Data Registers (ROMC_ROMPATCH2D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0E0	ROMC Data Registers (ROMC_ROMPATCH3D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0E4	ROMC Data Registers (ROMC_ROMPATCH4D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0E8	ROMC Data Registers (ROMC_ROMPATCH5D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0EC	ROMC Data Registers (ROMC_ROMPATCH6D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0F0	ROMC Data Registers (ROMC_ROMPATCH7D)	32	R/W	0000_0000h	41.5.1/2520
21A_C0F4	ROMC Control Register (ROMC_ROMPATCHCNTL)	32	R/W	0840_0000h	41.5.2/2521
21A_C0F8	ROMC Enable Register High (ROMC_ROMPATCHENH)	32	R	0000_0000h	41.5.3/2522
21A_C0FC	ROMC Enable Register Low (ROMC_ROMPATCHENL)	32	R/W	0000_0000h	41.5.4/2522
21A_C100	ROMC Address Registers (ROMC_ROMPATCH0A)	32	R/W	0000_0000h	41.5.5/2523
21A_C104	ROMC Address Registers (ROMC_ROMPATCH1A)	32	R/W	0000_0000h	41.5.5/2523
21A_C108	ROMC Address Registers (ROMC_ROMPATCH2A)	32	R/W	0000_0000h	41.5.5/2523
21A_C10C	ROMC Address Registers (ROMC_ROMPATCH3A)	32	R/W	0000_0000h	41.5.5/2523

Table continues on the next page...

ROMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21A_C110	ROMC Address Registers (ROMC_ROMPATCH4A)	32	R/W	0000_0000h	41.5.5/2523
21A_C114	ROMC Address Registers (ROMC_ROMPATCH5A)	32	R/W	0000_0000h	41.5.5/2523
21A_C118	ROMC Address Registers (ROMC_ROMPATCH6A)	32	R/W	0000_0000h	41.5.5/2523
21A_C11C	ROMC Address Registers (ROMC_ROMPATCH7A)	32	R/W	0000_0000h	41.5.5/2523
21A_C120	ROMC Address Registers (ROMC_ROMPATCH8A)	32	R/W	0000_0000h	41.5.5/2523
21A_C124	ROMC Address Registers (ROMC_ROMPATCH9A)	32	R/W	0000_0000h	41.5.5/2523
21A_C128	ROMC Address Registers (ROMC_ROMPATCH10A)	32	R/W	0000_0000h	41.5.5/2523
21A_C12C	ROMC Address Registers (ROMC_ROMPATCH11A)	32	R/W	0000_0000h	41.5.5/2523
21A_C130	ROMC Address Registers (ROMC_ROMPATCH12A)	32	R/W	0000_0000h	41.5.5/2523
21A_C134	ROMC Address Registers (ROMC_ROMPATCH13A)	32	R/W	0000_0000h	41.5.5/2523
21A_C138	ROMC Address Registers (ROMC_ROMPATCH14A)	32	R/W	0000_0000h	41.5.5/2523
21A_C13C	ROMC Address Registers (ROMC_ROMPATCH15A)	32	R/W	0000_0000h	41.5.5/2523
21A_C208	ROMC Status Register (ROMC_ROMPATCHSR)	32	w1c	0000_0000h	41.5.6/2524

41.5.1 ROMC Data Registers (ROMC_ROMPATCH*n*D)

The ROMC data registers (ROMC_ROMPATCH7D through ROMC_ROMPATCH0D) store the data to use for the 8 1-word data fix events. Each register is associated with an address comparator (7 through 0). When a data fixing event occurs, the value in the data register corresponding to the comparator that has the address match is put on the romc_hrdata[31:0] bus until romc_hready is asserted by the ROM controller to terminate the access. A MUX external to the ROMC will select this data over that of romc_hrdata[31:0] in returning read data to the ARM core. The selection is done with the control bus rompatch_romc_hrdata_ovr[1:0] with both bits asserted by the ROMC.

If more than one address comparators match, the highest-numbered one takes precedence, and the value in corresponding data register is used for the patching event.

Address: 21A_C000h base + D4h offset + (4d × i), where i=0d to 7d

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

ROMC_ROMPATCH*n*D field descriptions

Field	Description
DATAX	Data Fix Registers - Stores the data used for 1-word data fix operations. The values stored within these registers do not affect the writes to the memory system. They are selected over the read data from ROM when a data fix event occurs.

ROMC_ROMPATCH*n*D field descriptions (continued)

Field	Description
	If any part of the 1-word data fix is read, then the entire word is replaced. Therefore, a byte or half-word read will cause the ROMC to replace the entire word. The word is word address aligned.

41.5.2 ROMC Control Register (ROMC_ROMPATCHCNTL)

The ROMC control register (ROMC_ROMPATCHCNTL) contains the block disable bit and the data fix enable bits. The block disable bit provides a means to disable the ROMC data fix and opcode patching functions, even when the address comparators are enabled. The External Boot feature is not affected by this bit. The eight data fix enable bits (0 through 7), when set, assign the associated address comparators to data fix operations

NOTE

Bits 27 and 22 always read as 1s.

Address: 21A_C000h base + F4h offset = 21A_C0F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		DIS	Reserved												
Reset	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0

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

ROMC_ROMPATCHCNTL field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29 DIS	ROMC Disable -- This bit, when set, disables all ROMC operations. This bit is used to enable secure operations. 0 Does not affect any ROMC functions (default) 1 Disable all ROMC functions: data fixing, and opcode patching
28–8 -	This field is reserved. Reserved
DATAFIX	**Data Fix Enable - Controls the use of the first 8 address comparators for 1-word data fix or for code patch routine.** 0 Address comparator triggers a opcode patch 1 Address comparator triggers a data fix

41.5.3 ROMC Enable Register High (ROMC_ROMPATCHENH)

The ROMC enable register high (ROMC_ROMPATCHENH) and ROMC enable register low (ROMC_ROMPATCHENL) control whether or not the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMC only has 16 comparators, therefore ROMC_ROMPATCHENH and the upper half of ROMC_OMPATCHENL are read-only. ROMC_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Address: 21A_C000h base + F8h offset = 21A_C0F8h

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

ROMC_ROMPATCHENH field descriptions

Field	Description
-	This field is reserved. Reserved

41.5.4 ROMC Enable Register Low (ROMC_ROMPATCHENL)

The ROMC enable register high (ROMC_ROMPATCHENH) and ROMC enable register low (ROMC_ROMPATCHENL) control whether or not the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMC only has 16 comparators, therefore ROMC_ROMPATCHENH and the upper half of ROMC_ROMPATCHENL are read-only. ROMC_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Address: 21A_C000h base + FCh offset = 21A_C0FCh

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

ROMC_ROMPATCHENL field descriptions

Field	Description
31–16 Reserved	This field is reserved.
ENABLE	**Enable** Address **Comparator -** This bit enables the corresponding address comparator to trigger an event. 0 Address comparator disabled 1 Address comparator enabled, ROMC will trigger a opcode patch or data fix event upon matching of the associated address

41.5.5 ROMC Address Registers (ROMC_ROMPATCH*n*A)

The ROMC address registers (ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15) store the memory addresses where opcode patching begins and data fixing occurs. The address registers ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15 are each 21 bits wide and dedicated to one 4 Mbyte memory space. Bits 21 through 2 are address bits, to be compared with romc_haddr[21:2] for a match; bit 1 is also an address bit used for half word selection. Bit 0 is the mode bit (set to 1 for THUMB mode). 1-word data fixing can only be used on the first 8 of the address comparators. ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15 are associated each with address comparators 0 through 15.

Address: 21A_C000h base + 100h offset + (4d × i), where i=0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ADDRX							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ROMC_ROMPATCH*n*A field descriptions

Field	Description
31–23 -	This field is reserved. Reserved

Table continues on the next page...

ROMC_ROMPATCH*n*A field descriptions (continued)

Field	Description
22–1 ADDRX	Address Comparator Registers - Indicates the memory address to be watched. All 16 registers can be used for code patch address comparison. Only the first 8 registers can be used for a 1-word data fix address comparison. Bit 1 is ignored if data fix. Only used in code patch
0 THUMBX	THUMB Comparator Select - Indicates that this address will trigger a THUMB opcode patch or an ARM opcode patch. If this watchpoint is selected to be a data fix, then this bit is ignored as all data fixes are 1-word data fixes. 0 ARM patch 1 THUMB patch (ignore if data fix)

41.5.6 ROMC Status Register (ROMC_ROMPATCHSR)

The ROMC status register (ROMC_ROMPATCHSR) indicates the current state of the ROMC and the source number of the most recent address comparator event.

Address: 21A_C000h base + 208h offset = 21A_C208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved														SW	Reserved
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

ROMC_ROMPATCHSR field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 SW	ROMC AHB Multiple Address Comparator matches Indicator - Indicates that multiple address comparator matches occurred. Writing a 1 to this bit will clear this it. 0 no event or comparator collisions 1 a collision has occurred
16–6 -	This field is reserved. Reserved
SOURCE	ROMC Source Number - Binary encoding of the number of the address comparator which has an address match in the most recent patch event on ROMC AHB. If multiple matches occurred, the highest priority source number is used.

Table continues on the next page...

ROMC_ROMPATCHSR field descriptions (continued)

Field	Description
	0 Address Comparator 0 matched 1 Address Comparator 1 matched 15 Address Comparator 15 matched

Chapter 42
Synchronous Audio Interface (SAI)

42.1 Overview

The synchronous audio interface (SAI) supports full-duplex serial interfaces with frame synchronization such as I^2S, AC97, TDM, and codec/DSP interfaces.

42.1.1 Features

Note that some of the features are not supported across all SAI instances; see the chip-specific information in the first section of this chapter.

- Transmitter with independent bit clock and frame sync supporting 1 data line
- Receiver with independent bit clock and frame sync supporting 1 data line
- Maximum Frame Size of 32 words
- Word size of between 8-bits and 32-bits
- Word size configured separately for first word and remaining words in frame
- Asynchronous 32 × 32-bit FIFO for each transmit and receive channel
- Supports graceful restart after FIFO error

42.1.2 Block diagram

The following block diagram also shows the module clocks.

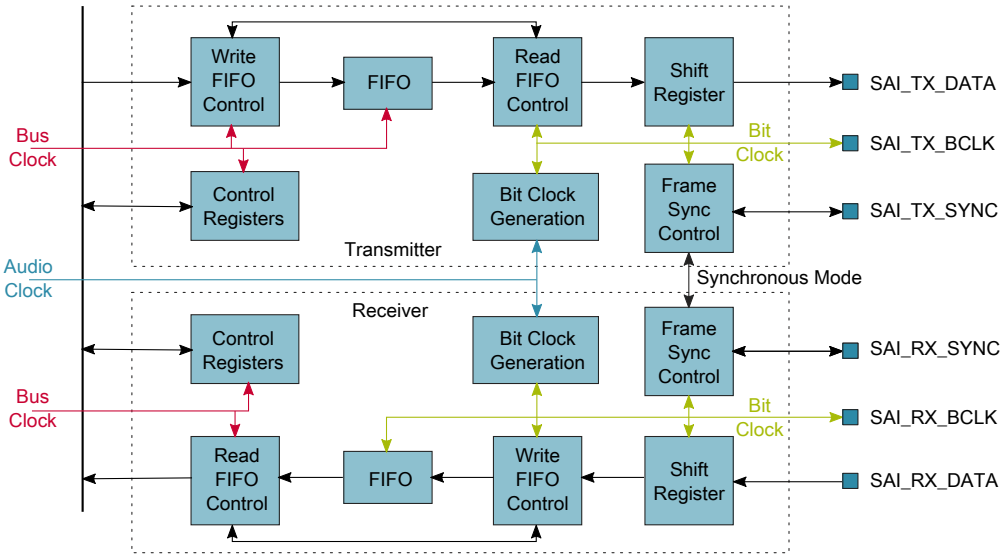


Figure 42-1. I²S/SAI block diagram

42.1.3 Modes of operation

The module operates in these power modes: Run mode, stop modes, and Debug mode.

42.1.3.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

42.1.3.2 Stop modes

In Stop mode, the transmitter is disabled after completing the current transmit frame, and, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented–not acknowledged–while waiting for the transmitter and receiver to be disabled at the end of the current frame.

42.1.3.3 Debug mode

In Debug mode, the SAI transmitter and/or receiver can continue operating provided the Debug Enable bit is set. When TCSR[DBGE] or RCSR[DBGE] bit is clear and Debug mode is entered, the SAI is disabled after completing the current transmit or receive frame. The transmitter and receiver bit clocks are not affected by Debug mode.

42.2 External Signals

The following table describes the external signals of SAI:

Table 42-1. SAI External Signals

Signal	Description	Pad	Mode	Direction
SAI1_MCLK	Audio Master Clock. The master clock is an input when externally generated and an output when internally generated.	CSI_DATA01	ALT6	IO
		LCD_DATA00	ALT8	
SAI1_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	CSI_DATA03	ALT6	IO
SAI1_RX_DATA	Receive Data. The receive data is sampled synchronously by the bit clock.	CSI_DATA06	ALT6	I
		LCD_DATA03	ALT8	
SAI1_RX_SYNC	Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	CSI_DATA02	ALT6	IO
SAI1_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	CSI_DATA05	ALT6	IO
		LCD_DATA02	ALT8	
SAI1_TX_DATA	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	CSI_DATA07	ALT6	O
		LCD_DATA04	ALT8	
SAI1_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	CSI_DATA04	ALT6	IO
		LCD_DATA01	ALT8	
SAI2_MCLK	Audio Master Clock. The master clock is an input when externally generated and an output when internally generated.	JTAG_TMS	ALT2	IO
		SD1_CLK	ALT2	
SAI2_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	NAND_DATA06	ALT2	IO
SAI2_RX_DATA	Receive Data. The receive data is sampled synchronously by the bit clock.	JTAG_TCK	ALT2	I
		SD1_DATA2	ALT2	
SAI2_RX_SYNC	Receive Frame Sync. The frame sync is an input sampled	SD1_CMD	ALT2	IO

Table continues on the next page...

Table 42-1. SAI External Signals (continued)

Signal	Description	Pad	Mode	Direction
	synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.			
SAI2_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	JTAG_TDI	ALT2	IO
		SD1_DATA1	ALT2	
SAI2_TX_DATA	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	JTAG_TRST_B	ALT2	O
		SD1_DATA3	ALT2	
SAI2_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	JTAG_TDO	ALT2	IO
		SD1_DATA0	ALT2	
SAI3_MCLK	Audio Master Clock. The master clock is an input when externally generated and an output when internally generated.	LCD_CLK	ALT3	IO
		LCD_DATA09	ALT1	
SAI3_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	LCD_DATA11	ALT1	IO
SAI3_RX_DATA	Receive Data. The receive data is sampled synchronously by the bit clock.	LCD_DATA14	ALT1	I
SAI3_RX_SYNC		LCD_VSYNC	ALT3	IO
	Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	LCD_DATA10	ALT1	
SAI3_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	LCD_DATA13	ALT1	IO
		LCD_HSYNC	ALT3	
SAI3_TX_DATA	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	LCD_DATA15	ALT1	O
		LCD_RESET	ALT3	
SAI3_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	LCD_DATA12	ALT1	IO
		LCD_ENABLE	ALT3	

42.3 Functional description

This section provides a complete functional description of the block.

42.3.1 SAI clocking

The SAI clocks include:

- The audio master clock
- The bit clock
- The bus clock

42.3.1.1 Audio master clock

The audio master clock is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio master clocks to generate the bit clock.

The audio master clock generation and selection is chip-specific. Refer to chip-specific clocking information about how the audio master clocks are generated.

42.3.1.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames

If the SAI transmitter or receiver is using an externally generated bit clock in asynchronous mode and that bit clock is generated by an SAI that is disabled in stop mode, then the transmitter or receiver should be disabled by software before entering stop mode. This issue does not apply when the transmitter or receiver is in a synchronous mode because all synchronous SAIs are enabled and disabled simultaneously.

42.3.1.3 Bus clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

NOTE

Although there is no specific minimum bus clock frequency specified, the bus clock frequency must be fast enough (relative to the bit clock frequency) to ensure that the FIFOs can be serviced, without generating either a transmitter FIFO underrun or receiver FIFO overflow condition.

42.3.2 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

42.3.2.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

42.3.2.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This empties the FIFO contents and is to be used after TCSR[FEF] is set, and before the FIFO is re-initialized and TCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the RCSR[FEF] is set and any remaining data has been read from the FIFO, and before the RCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

42.3.3 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other.

42.3.3.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If the transmitter bit clock and frame sync are to be used by both the transmitter and receiver:

- The transmitter must be configured for asynchronous operation and the receiver for synchronous operation.
- In synchronous mode, the receiver is enabled only when both the transmitter and receiver are enabled.
- It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both the transmitter and receiver:

- The receiver must be configured for asynchronous operation and the transmitter for synchronous operation.
- In synchronous mode, the transmitter is enabled only when both the receiver and transmitter are both enabled.
- It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

42.3.4 Frame sync configuration

When enabled, the SAI continuously transmits and/or receives frames of data. Each frame consists of a fixed number of words and each word consists of a fixed number of bits. Within each frame, any given word can be masked causing the receiver to ignore that word and the transmitter to tri-state for the duration of that word.

The frame sync signal is used to indicate the start of each frame. A valid frame sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the transmitter or receiver cannot be busy with a previous frame. A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the transmitter or receiver.

The transmitter and receiver frame sync can be configured independently with any of the following options:

- Externally generated or internally generated
- Active high or active low
- Assert with the first bit in frame or asserts one bit early
- Assert for a duration between 1 bit clock and the first word length
- Frame length from 1 to 32 words per frame
- Word length to support 8 to 32 bits per word
    - First word length and remaining word lengths can be configured separately
- Words can be configured to transmit/receive MSB first or LSB first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

42.3.5 Data FIFO

Each transmit and receive channel includes a FIFO of size 32 × 32-bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers.

42.3.5.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration are illustrated in Figure 42-2 for LSB First configurations and Figure 42-3 for MSB First configurations.

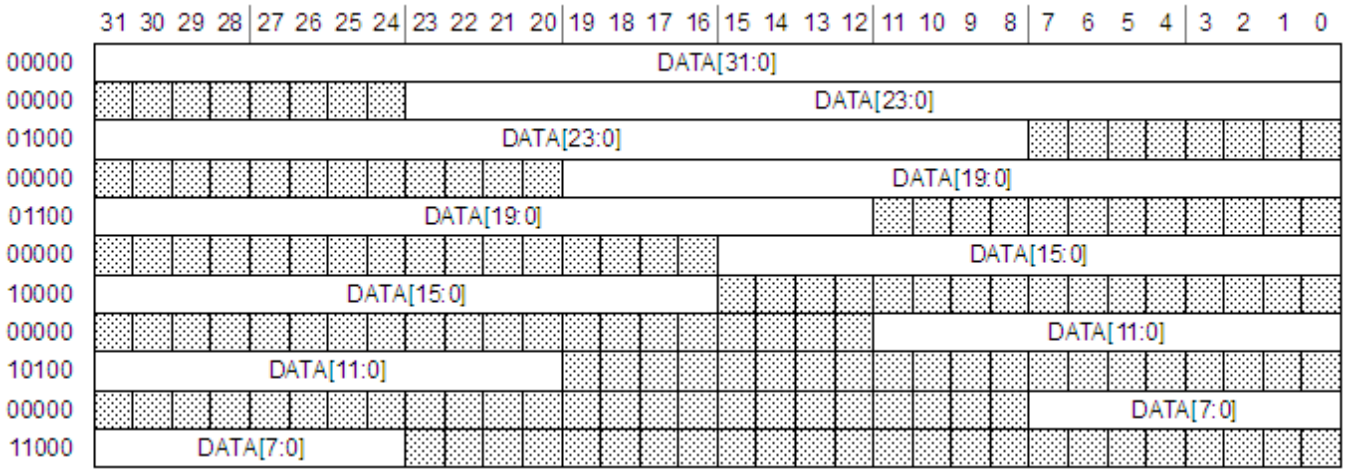


Figure 42-2. SAI first bit shifted, LSB first

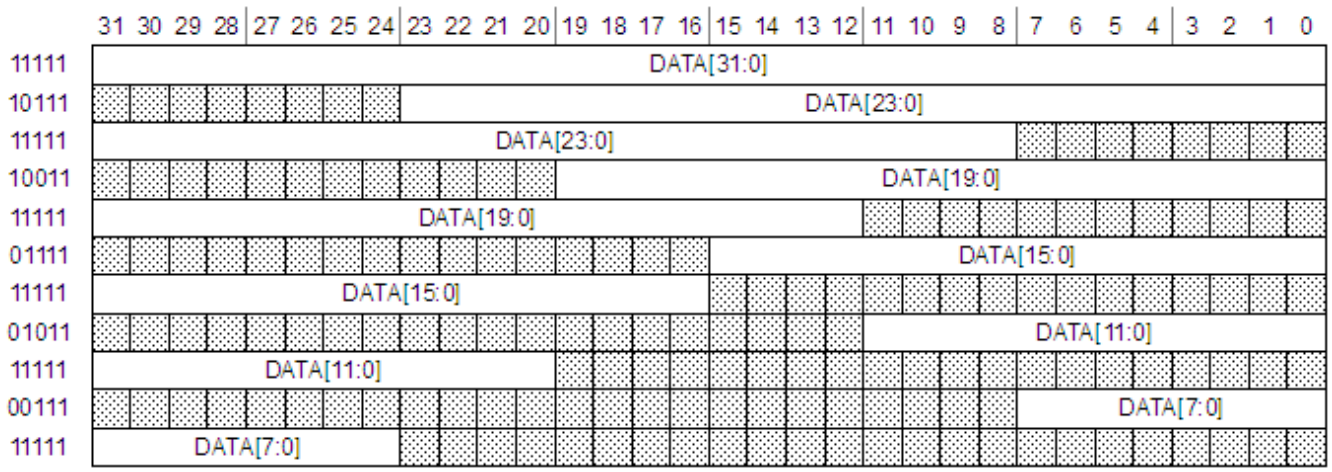


Figure 42-3. SAI first bit shifted, MSB first

42.3.5.2 FIFO pointers

When writing to a TDR, the WFP of the corresponding TFR increments after each valid write. The SAI supports 8-bit, 16-bit and 32-bit writes to the TDR and the FIFO pointer will increment after each individual write. Note that 8-bit writes should only be used when transmitting up to 8-bit data and 16-bit writes should only be used when transmitting up to 16-bit data.

Writes to a TDR are ignored if the corresponding bit of TCR3[TCE] is clear or if the FIFO is full. If the Transmit FIFO is empty, the TDR must be written at least three bit clocks before the start of the next unmasked word to avoid a FIFO underrun.

When reading an RDR, the RFP of the corresponding RFR increments after each valid read. The SAI supports 8-bit, 16-bit and 32-bit reads from the RDR and the FIFO pointer will increment after each individual read. Note that 8-bit reads should only be used when receiving up to 8-bit data and 16-bit reads should only be used when receiving up to 16-bit data.

Reads from an RDR are ignored if the corresponding bit of RCR3[RCE] is clear or if the FIFO is empty. If the Receive FIFO is full, the RDR must be read at least three bit clocks before the end of an unmasked word to avoid a FIFO overrun.

42.3.6 Word mask register

The SAI transmitter and receiver each contain a word mask register, namely TMR and RMR, that can be used to mask any word in the frame. Because the word mask register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The TMR causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

42.3.7 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags.

42.3.7.1 FIFO request flag

The FIFO request flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO request flag can generate an interrupt or a DMA request.

42.3.7.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

42.3.7.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels repeat the last valid word read from the transmit FIFO until TCSR[FEF] is cleared and the next transmit frame starts. All enabled transmit FIFOs must be reset and initialized with new data before TCSR[FEF] is cleared.

RCSR[FEF] is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until RCSR[FEF] is cleared and the next next receive frame starts. All enabled receive FIFOs should be emptied before RCSR[FEF] is cleared.

The FIFO error flag can generate only an interrupt.

42.3.7.4 Sync error flag

The sync error flag, TCSR[SEF] or RCSR[SEF], is set when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag is set. When the sync error flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The sync error flag can generate an interrupt only.

42.3.7.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

42.4 Memory map and register definition

A read or write access to an address from offset and above will result in a bus error.

I2S memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_8000	SAI Transmit Control Register (I2S1_TCSR)	32	R/W	0000_0000h	42.4.1/2541
202_8004	SAI Transmit Configuration 1 Register (I2S1_TCR1)	32	R/W	0000_0000h	42.4.2/2544
202_8008	SAI Transmit Configuration 2 Register (I2S1_TCR2)	32	R/W	0000_0000h	42.4.3/2544
202_800C	SAI Transmit Configuration 3 Register (I2S1_TCR3)	32	R/W	0000_0000h	42.4.4/2546
202_8010	SAI Transmit Configuration 4 Register (I2S1_TCR4)	32	R/W	0000_0000h	42.4.5/2547
202_8014	SAI Transmit Configuration 5 Register (I2S1_TCR5)	32	R/W	0000_0000h	42.4.6/2548
202_8020	SAI Transmit Data Register (I2S1_TDR0)	32	W (always reads 0)	0000_0000h	42.4.7/2549
202_8040	SAI Transmit FIFO Register (I2S1_TFR0)	32	R	0000_0000h	42.4.8/2550
202_8060	SAI Transmit Mask Register (I2S1_TMR)	32	R/W	0000_0000h	42.4.9/2550
202_8080	SAI Receive Control Register (I2S1_RCSR)	32	R/W	0000_0000h	42.4.10/ 2551
202_8084	SAI Receive Configuration 1 Register (I2S1_RCR1)	32	R/W	0000_0000h	42.4.11/ 2554
202_8088	SAI Receive Configuration 2 Register (I2S1_RCR2)	32	R/W	0000_0000h	42.4.12/ 2555

Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_808C	SAI Receive Configuration 3 Register (I2S1_RCR3)	32	R/W	0000_0000h	42.4.13/ 2556
202_8090	SAI Receive Configuration 4 Register (I2S1_RCR4)	32	R/W	0000_0000h	42.4.14/ 2557
202_8094	SAI Receive Configuration 5 Register (I2S1_RCR5)	32	R/W	0000_0000h	42.4.15/ 2559
202_80A0	SAI Receive Data Register (I2S1_RDR0)	32	R	0000_0000h	42.4.16/ 2559
202_80C0	SAI Receive FIFO Register (I2S1_RFR0)	32	R	0000_0000h	42.4.17/ 2560
202_80E0	SAI Receive Mask Register (I2S1_RMR)	32	R/W	0000_0000h	42.4.18/ 2560
202_C000	SAI Transmit Control Register (I2S2_TCSR)	32	R/W	0000_0000h	42.4.1/2541
202_C004	SAI Transmit Configuration 1 Register (I2S2_TCR1)	32	R/W	0000_0000h	42.4.2/2544
202_C008	SAI Transmit Configuration 2 Register (I2S2_TCR2)	32	R/W	0000_0000h	42.4.3/2544
202_C00C	SAI Transmit Configuration 3 Register (I2S2_TCR3)	32	R/W	0000_0000h	42.4.4/2546
202_C010	SAI Transmit Configuration 4 Register (I2S2_TCR4)	32	R/W	0000_0000h	42.4.5/2547
202_C014	SAI Transmit Configuration 5 Register (I2S2_TCR5)	32	R/W	0000_0000h	42.4.6/2548
202_C020	SAI Transmit Data Register (I2S2_TDR0)	32	W (always reads 0)	0000_0000h	42.4.7/2549
202_C040	SAI Transmit FIFO Register (I2S2_TFR0)	32	R	0000_0000h	42.4.8/2550
202_C060	SAI Transmit Mask Register (I2S2_TMR)	32	R/W	0000_0000h	42.4.9/2550
202_C080	SAI Receive Control Register (I2S2_RCSR)	32	R/W	0000_0000h	42.4.10/ 2551
202_C084	SAI Receive Configuration 1 Register (I2S2_RCR1)	32	R/W	0000_0000h	42.4.11/ 2554
202_C088	SAI Receive Configuration 2 Register (I2S2_RCR2)	32	R/W	0000_0000h	42.4.12/ 2555
202_C08C	SAI Receive Configuration 3 Register (I2S2_RCR3)	32	R/W	0000_0000h	42.4.13/ 2556
202_C090	SAI Receive Configuration 4 Register (I2S2_RCR4)	32	R/W	0000_0000h	42.4.14/ 2557
202_C094	SAI Receive Configuration 5 Register (I2S2_RCR5)	32	R/W	0000_0000h	42.4.15/ 2559
202_C0A0	SAI Receive Data Register (I2S2_RDR0)	32	R	0000_0000h	42.4.16/ 2559
202_C0C0	SAI Receive FIFO Register (I2S2_RFR0)	32	R	0000_0000h	42.4.17/ 2560
202_C0E0	SAI Receive Mask Register (I2S2_RMR)	32	R/W	0000_0000h	42.4.18/ 2560
203_0000	SAI Transmit Control Register (I2S3_TCSR)	32	R/W	0000_0000h	42.4.1/2541
203_0004	SAI Transmit Configuration 1 Register (I2S3_TCR1)	32	R/W	0000_0000h	42.4.2/2544

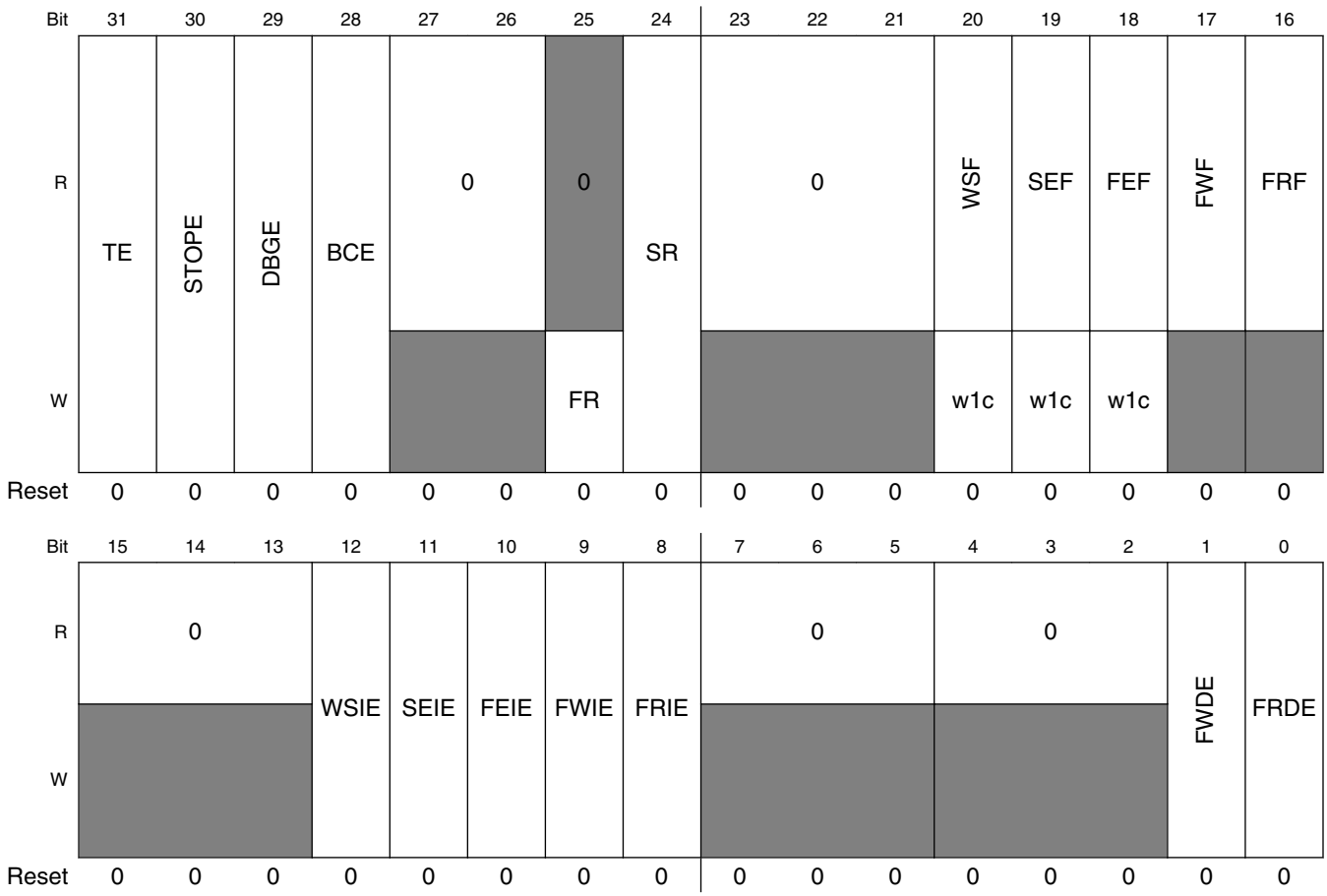
Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
203_0008	SAI Transmit Configuration 2 Register (I2S3_TCR2)	32	R/W	0000_0000h	42.4.3/2544
203_000C	SAI Transmit Configuration 3 Register (I2S3_TCR3)	32	R/W	0000_0000h	42.4.4/2546
203_0010	SAI Transmit Configuration 4 Register (I2S3_TCR4)	32	R/W	0000_0000h	42.4.5/2547
203_0014	SAI Transmit Configuration 5 Register (I2S3_TCR5)	32	R/W	0000_0000h	42.4.6/2548
203_0020	SAI Transmit Data Register (I2S3_TDR0)	32	W (always reads 0)	0000_0000h	42.4.7/2549
203_0040	SAI Transmit FIFO Register (I2S3_TFR0)	32	R	0000_0000h	42.4.8/2550
203_0060	SAI Transmit Mask Register (I2S3_TMR)	32	R/W	0000_0000h	42.4.9/2550
203_0080	SAI Receive Control Register (I2S3_RCSR)	32	R/W	0000_0000h	42.4.10/ 2551
203_0084	SAI Receive Configuration 1 Register (I2S3_RCR1)	32	R/W	0000_0000h	42.4.11/ 2554
203_0088	SAI Receive Configuration 2 Register (I2S3_RCR2)	32	R/W	0000_0000h	42.4.12/ 2555
203_008C	SAI Receive Configuration 3 Register (I2S3_RCR3)	32	R/W	0000_0000h	42.4.13/ 2556
203_0090	SAI Receive Configuration 4 Register (I2S3_RCR4)	32	R/W	0000_0000h	42.4.14/ 2557
203_0094	SAI Receive Configuration 5 Register (I2S3_RCR5)	32	R/W	0000_0000h	42.4.15/ 2559
203_00A0	SAI Receive Data Register (I2S3_RDR0)	32	R	0000_0000h	42.4.16/ 2559
203_00C0	SAI Receive FIFO Register (I2S3_RFR0)	32	R	0000_0000h	42.4.17/ 2560
203_00E0	SAI Receive Mask Register (I2S3_RMR)	32	R/W	0000_0000h	42.4.18/ 2560

42.4.1 SAI Transmit Control Register (I2Sx_TCSR)

Address: Base address + 0h offset



I2Sx_TCSR field descriptions

Field	Description
31 TE	Transmitter Enable Enables/disables the transmitter. When software clears this field, the transmitter remains enabled, and this bit remains set, until the end of the current frame. 0 Transmitter is disabled. 1 Transmitter is enabled, or transmitter has been disabled and has not yet reached end of frame.
30 STOPE	Stop Enable Configures transmitter operation in Stop mode. This field is ignored and the transmitter is disabled in all stop modes. 0 Transmitter disabled in Stop mode. 1 Transmitter enabled in Stop mode.
29 DBGE	Debug Enable

Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
	Enables/disables transmitter operation in Debug mode. The transmit bit clock is not affected by debug mode. 0 Transmitter is disabled in Debug mode, after completing the current frame. 1 Transmitter is enabled in Debug mode.
28 BCE	Bit Clock Enable Enables the transmit bit clock, separately from the TE. This field is automatically set whenever TE is set. When software clears this field, the transmit bit clock remains enabled, and this bit remains set, until the end of the current frame. 0 Transmit bit clock is disabled. 1 Transmit bit clock is enabled.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the transmitter is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
24 SR	Software Reset When set, resets the internal transmitter logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0 Start of word not detected. 1 Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0 Sync error not detected. 1 Frame sync error detected.
18 FEF	FIFO Error Flag Indicates that an enabled transmit FIFO has underrun. Write a logic 1 to this field to clear this flag. 0 Transmit underrun not detected. 1 Transmit underrun detected.

Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
17 FWF	FIFO Warning Flag Indicates that an enabled transmit FIFO is empty. 0 No enabled transmit FIFO is empty. 1 Enabled transmit FIFO is empty.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark. 0 Transmit FIFO watermark has not been reached. 1 Transmit FIFO watermark has been reached.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FWDE	FIFO Warning DMA Enable

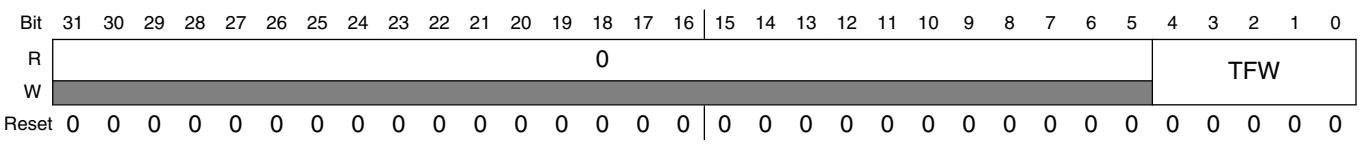
Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
	Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

42.4.2 SAI Transmit Configuration 1 Register (I2Sx_TCR1)

Address: Base address + 4h offset



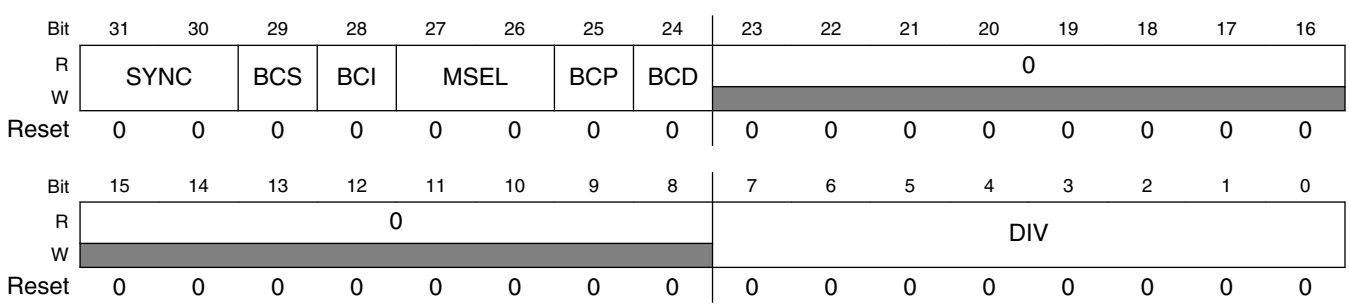
I2Sx_TCR1 field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TFW	Transmit FIFO Watermark Configures the watermark level for all enabled transmit channels.

42.4.3 SAI Transmit Configuration 2 Register (I2Sx_TCR2)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 8h offset



I2Sx_TCR2 field descriptions

Field	Description
31–30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the receiver must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with receiver.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (SAI_RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (SAI_TX_SYNC). When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (SAI_TX_BCLK) but use the receiver frame sync (SAI_RX_SYNC). 0 Use the normal bit clock source. 1 Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the transmitter is delayed by the pad output delay (the transmitter is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the transmitter when this bit is set. In synchronous mode, this bit allows the transmitter to use the slave mode timing from the datasheet, while the receiver uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
27–26 MSEL	MCLK Select Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the meaning of each option. 00 Master Clock (MCLK) 1 option selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction

Table continues on the next page...

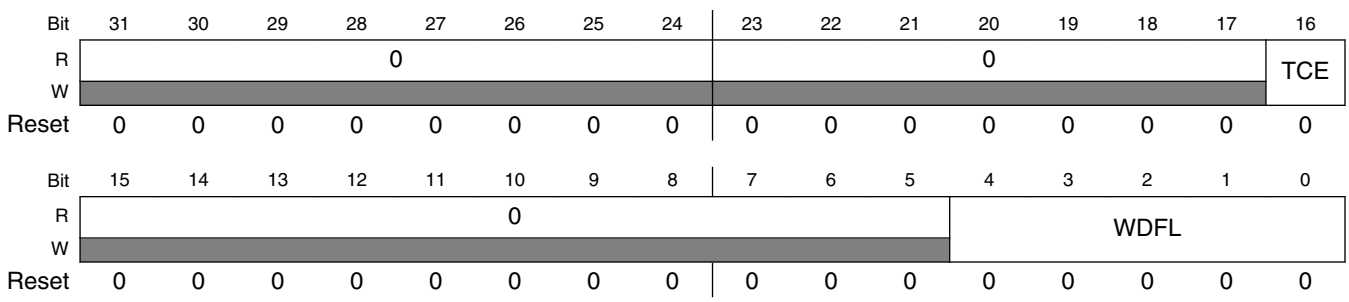
I2Sx_TCR2 field descriptions (continued)

Field	Description
	Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

42.4.4 SAI Transmit Configuration 3 Register (I2Sx_TCR3)

This register must not be altered when TCSR[TE] is set.

Address: Base address + Ch offset



I2Sx_TCR3 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TCE	Transmit Channel Enable Enables the corresponding data channel for transmit operation. A channel must be enabled before its FIFO is accessed. 0 Transmit data channel N is disabled. 1 Transmit data channel N is enabled.
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WDFL	Word Flag Configuration

Table continues on the next page...

I2Sx_TCR3 field descriptions (continued)

Field	Description
	Configures which word sets the start of word flag. The value written must be one less than the word number. For example, writing 0 configures the first word in the frame. When configured to a value greater than TCR4[FRSZ], then the start of word flag is never set.

42.4.5 SAI Transmit Configuration 4 Register (I2Sx_TCR4)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			0	0		0		0			FRSZ				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

I2Sx_TCR4 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 FRSZ	Frame size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

I2Sx_TCR4 field descriptions (continued)

Field	Description
4 MF	MSB First Configures whether the LSB or the MSB is transmitted first. 0 LSB is transmitted first. 1 MSB is transmitted first.
3 FSE	Frame Sync Early 0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame sync is generated externally in Slave mode. 1 Frame sync is generated internally in Master mode.

42.4.6 SAI Transmit Configuration 5 Register (I2Sx_TCR5)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WNW					0			W0W					0			FBT					0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_TCR5 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–24 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

I2Sx_TCR5 field descriptions (continued)

Field	Description
20–16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 FBT	First Bit Shifted Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

42.4.7 SAI Transmit Data Register (I2S*x*_TDR*n*)

Address: Base address + 20h offset + (4d × i), where i=0d to 0d

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

I2S*x*_TDR*n* field descriptions

Field	Description
TDR	Transmit Data Register The corresponding TCR3[TCE] bit must be set before accessing the channel's transmit data register. Writes to this register when the transmit FIFO is not full will push the data written into the transmit data FIFO. Writes to this register when the transmit FIFO is full are ignored.

42.4.8 SAI Transmit FIFO Register (I2Sx_TFR*n*)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: Base address + 40h offset + (4d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0									WFP					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

I2Sx_TFR*n* field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 WFP	Write FIFO Pointer FIFO write pointer for transmit data channel.
15–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFP	Read FIFO Pointer FIFO read pointer for transmit data channel.

42.4.9 SAI Transmit Mask Register (I2Sx_TMR)

This register is double-buffered and updates:

1. When TCSR[TE] is first set
2. At the end of each frame.

This allows the masked words in each frame to change from frame to frame.

Address: Base address + 60h offset

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

I2Sx_TMR field descriptions

Field	Description
TWM	Transmit Word Mask Configures whether the transmit word is masked (transmit data pin tristated and transmit data not read from FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked. The transmit data pins are tri-stated when masked.

42.4.10 SAI Receive Control Register (I2Sx_RCSR)

Address: Base address + 80h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RE	STOPE	DBGE	BCE	0		0	SR	0			WSF	SEF	FEF	FWF	FRF
W							FR					w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WSIE	SEIE	FEIE	FWIE	FRIE	0			0			FWDE	FRDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCSR field descriptions

Field	Description
31 RE	Receiver Enable Enables/disables the receiver. When software clears this field, the receiver remains enabled, and this bit remains set, until the end of the current frame. 0 Receiver is disabled. 1 Receiver is enabled, or receiver has been disabled and has not yet reached end of frame.
30 STOPE	Stop Enable Configures receiver operation in Stop mode. This bit is ignored and the receiver is disabled in all stop modes. 0 Receiver disabled in Stop mode. 1 Receiver enabled in Stop mode.
29 DBGE	Debug Enable Enables/disables receiver operation in Debug mode. The receive bit clock is not affected by Debug mode. 0 Receiver is disabled in Debug mode, after completing the current frame. 1 Receiver is enabled in Debug mode.
28 BCE	Bit Clock Enable Enables the receive bit clock, separately from RE. This field is automatically set whenever RE is set. When software clears this field, the receive bit clock remains enabled, and this field remains set, until the end of the current frame. 0 Receive bit clock is disabled. 1 Receive bit clock is enabled.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the receiver is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
24 SR	Software Reset Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag.

Table continues on the next page...

I2Sx_RCSR field descriptions (continued)

Field	Description
	0 Start of word not detected. 1 Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0 Sync error not detected. 1 Frame sync error detected.
18 FEF	FIFO Error Flag Indicates that an enabled receive FIFO has overflowed. Write a logic 1 to this field to clear this flag. 0 Receive overflow not detected. 1 Receive overflow detected.
17 FWF	FIFO Warning Flag Indicates that an enabled receive FIFO is full. 0 No enabled receive FIFO is full. 1 Enabled receive FIFO is full.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark. 0 Receive FIFO watermark not reached. 1 Receive FIFO watermark has been reached.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts.

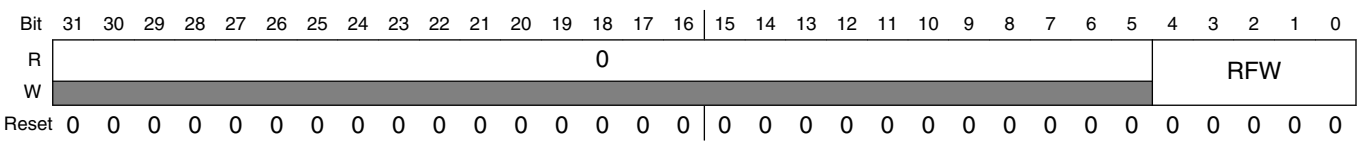
Table continues on the next page...

I2Sx_RCSR field descriptions (continued)

Field	Description
	0 Disables the interrupt. 1 Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

42.4.11 SAI Receive Configuration 1 Register (I2Sx_RCR1)

Address: Base address + 84h offset



I2Sx_RCR1 field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFW	Receive FIFO Watermark Configures the watermark level for all enabled receiver channels.

42.4.12 SAI Receive Configuration 2 Register (I2Sx_RCR2)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 88h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SYNC		BCS	BCI	MSEL		BCP	BCD	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

I2Sx_RCR2 field descriptions

Field	Description
31–30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the transmitter must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with transmitter.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the receiver. When the receiver is configured in asynchronous mode and this bit is set, the receiver is clocked by the transmitter bit clock (SAI_TX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the receiver continues to use the receiver frame sync (SAI_RX_SYNC). When the receiver is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the receiver bit clock (SAI_RX_BCLK) but use the transmitter frame sync (SAI_TX_SYNC). 0 Use the normal bit clock source. 1 Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the receiver is delayed by the pad output delay (the receiver is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the receiver when this bit is set. In synchronous mode, this bit allows the receiver to use the slave mode timing from the datasheet, while the transmitter uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
27–26 MSEL	MCLK Select

Table continues on the next page...

I2Sx_RCR2 field descriptions (continued)

Field	Description
	Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the availability and chip-specific meaning of each option. 00 Bus Clock selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit Clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit Clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

42.4.13 SAI Receive Configuration 3 Register (I2Sx_RCR3)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 8Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0							RCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

I2Sx_RCR3 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 RCE	Receive Channel Enable Enables the corresponding data channel for receive operation. A channel must be enabled before its FIFO is accessed. 0 Receive data channel N is disabled. 1 Receive data channel N is enabled.
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WDFL	Word Flag Configuration Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set.

42.4.14 SAI Receive Configuration 4 Register (I2Sx_RCR4)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			0	0		0		0			FRSZ				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

I2Sx_RCR4 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

I2Sx_RCR4 field descriptions (continued)

Field	Description
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 FRSZ	Frame Size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MF	MSB First Configures whether the LSB or the MSB is received first. 0 LSB is received first. 1 MSB is received first.
3 FSE	Frame Sync Early 0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame Sync is generated externally in Slave mode. 1 Frame Sync is generated internally in Master mode.

42.4.15 SAI Receive Configuration 5 Register (I2Sx_RCR5)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 94h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WNW					0			W0W					0			FBT					0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCR5 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–24 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 FBT	First Bit Shifted Configures the bit index for the first bit received for each word in the frame. If configured for MSB First, the index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

42.4.16 SAI Receive Data Register (I2Sx_RDR*n*)

Reading this register introduces one additional peripheral clock wait state on each read.

Address: Base address + A0h offset + (4d × i), where i=0d to 0d

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

I2Sx_RDRn field descriptions

Field	Description
RDR	Receive Data Register The corresponding RCR3[RCE] bit must be set before accessing the channel's receive data register. Reads from this register when the receive FIFO is not empty will return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored.

42.4.17 SAI Receive FIFO Register (I2Sx_RFRn)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: Base address + C0h offset + (4d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										WFP					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

I2Sx_RFRn field descriptions

Field	Description
31–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 WFP	Write FIFO Pointer FIFO write pointer for receive data channel.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFP	Read FIFO Pointer FIFO read pointer for receive data channel.

42.4.18 SAI Receive Mask Register (I2Sx_RMR)

This register is double-buffered and updates:

1. When RCSR[RE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

Address: Base address + E0h offset

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

I2Sx_RMR field descriptions

Field	Description
RWM	Receive Word Mask Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked.

Chapter 43
Smart Direct Memory Access Controller (SDMA)

43.1 Overview

The Smart Direct Memory Access (SDMA) controller offers highly-competitive DMA features combined with software-based virtual-DMA flexibility. It enables data transfers between peripheral I/O devices and internal/external memories.

The SDMA controller helps maximize system performance by off-loading the ARM core in dynamic data routing.

43.1.1 Block Diagram

The figure below shows a block diagram of the SDMA controller. It includes the custom RISC core along with its RAM, ROM, DMA units, and the scheduler.

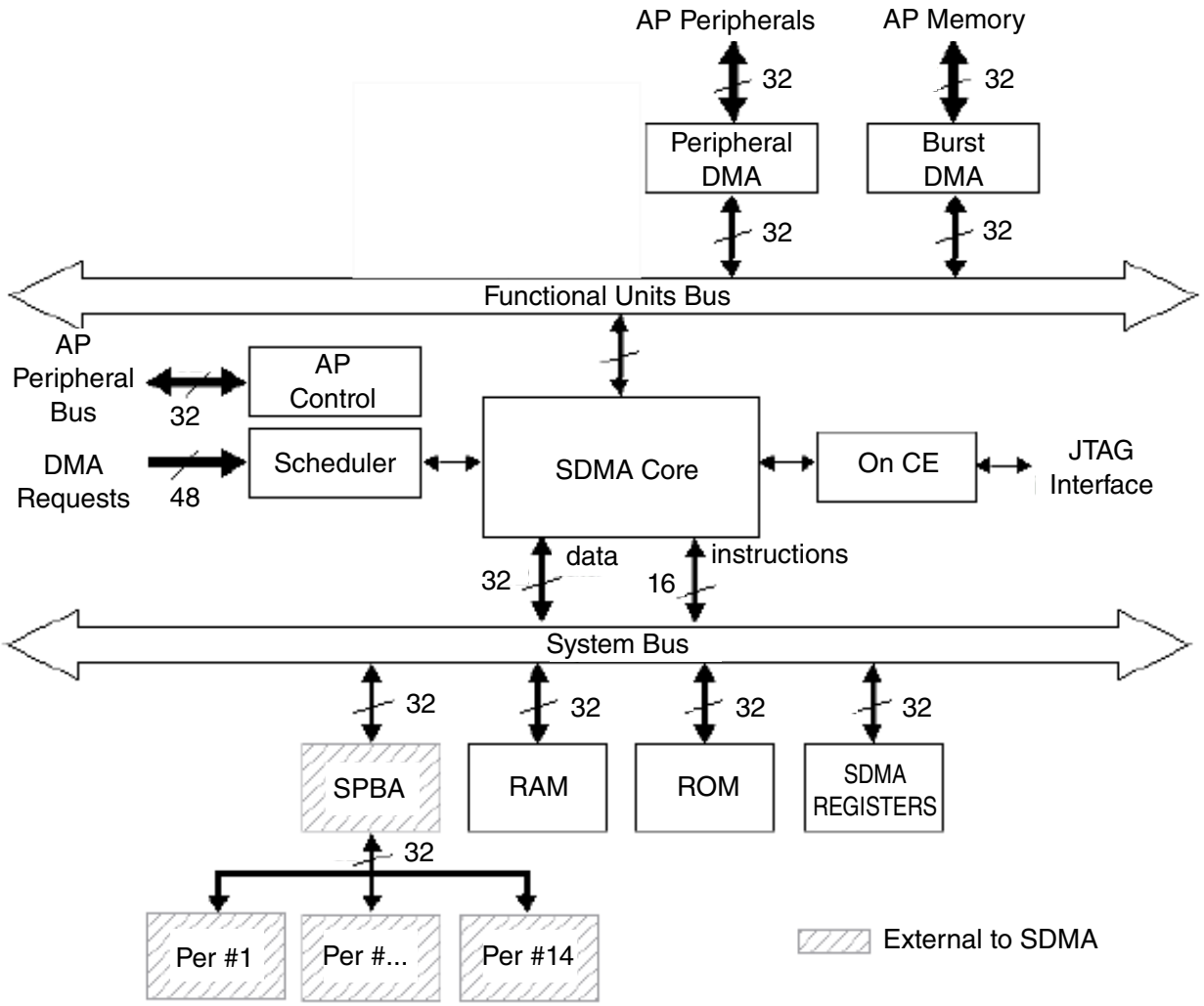


Figure 43-1. SDMA Block Diagram

The SDMA core executes short routines that perform DMA transfers; these routines are called *scripts*. The SDMA core interfaces to its own memory via the SDMA system bus. The SDMA system bus supports a 32-bit data path and a 16-bit address bus. The system bus datapath is used for both 16-bit instruction (program) memory access and 32-bit data access. DMA units interface to the core via the Functional Unit Bus and use dedicated registers to perform DMA transfers.

The SDMA memory contains a ROM and a RAM. The ROM contains startup scripts (for example, boot code) and other common utilities, which are referenced by the scripts that reside in the RAM. The internal RAM is divided into a context area and a script area (more details about this mapping are available in Instruction Memory Map and Data Memory Map).

Every transfer channel requires one context area to keep the contents of all the core and unit registers while inactive. Channel scripts are downloaded into the internal RAM by the SDMA using a dedicated channel that is started during the boot sequence. Downloads are invoked using commands and pointers provided by the ARM platform. Every channel contains a corresponding channel script located in RAM and/or ROM that can be reconfigured independently as-needed. Channel scripts can be stored in an external memory and downloaded when needed. The SDMA can be configured with any mixture of scripts to enable an endless combination of supported services.

The scheduler monitors and detects DMA requests, mapping them to channels, and mapping individual channels to a pre-configured priority. At any given point, the scheduler presents the highest priority channel that requires service to the SDMA core. A special SDMA core instruction is used to "conditionally yield" the current channel being executed to an eligible channel that requires service. If (and only if) there is an eligible channel pending, will the current channel execution be preempted.

There are two yield instructions that differently determine the eligible channels: In the first version, eligible channels are pending channels with a strictly higher priority than the current channel priority. In the second version (yieldge), eligible channels are pending channels with a priority that is greater or equal to the current channel priority. The scheduler detects devices that need service through its 48 DMA request inputs. After a request is detected, the scheduler determines the channel(s) that is (are) triggered by this request and marks it (them) as pending in the "Channel Pending (EP)" register. The priorities of all the pending channels are continuously evaluated in order to update the highest pending priority. The channel pending flag is cleared by the channel script when the transfer has completed.

The ARM platform control block contains the control registers used to configure the 32 individual channels. There are 48 Channel Enable registers, and every register maps one DMA request to any desired combination of channels. The 32 Priority registers are used to assign a programmable 1-of-7 level priority to every possible channel. This block also contains all other control registers that the ARM platform can access.

The 48 DMA requests that are connected to the scheduler come from a variety of sources. The "receive register full" and "transmit register empty" signals found in the UART and USB ports are typical examples of DMA requests that can be connected to the SDMA. These requests can be used to trigger a specific SDMA channel, or several channels.

There is an OnCE compatible debug port for product development. The OnCE includes support for setting breakpoints, single-step and trace, and register dump capability. In addition, all memory locations are accessible from the debug port.

43.1.2 Features

The following are the SDMA features:

- Multi-channel DMA supporting up to 32 time-division multiplexed DMA channels
- Hardware or software driven triggers for each channel
- 48 hardware driven triggers that can be mapped to any channel.
- Memory accesses including linear addressing, FIFO addressing and 2D addressing
- Fast context-switching with two-level, priority-based preemptive multi-tasking
- 16-bit instruction-set micro-RISC engine (the SDMA core)
- Two DMA units with some or all the following features:
  - Auto-flush and prefetch capability
  - Flexible address management (increment, decrement, and no address changes on source and destination address)
  - Misaligned data-transfer support
  - Uni-directional and bi-directional flows (copy mode)
  - Up to eight-word buffers for configurable burst transfers
- Support of byte-swapping
- An available API and library of scripts
- Little-Endian and Big-Endian modes
- Hardware handshakes for low-power entry sequence
- Security support to lock contents of the SDMA script RAM.
- 4-Kbyte ROM containing startup scripts (for example, boot code) and other common utilities that can be referenced by RAM-located scripts
- 8-Kbyte RAM area is divided into a processor context area and a code space area used to store channel scripts that are downloaded from the system memory
- Debug support, including a OnCE port, real-time monitors, and embedded cross-trigger events
- Supported clock frequencies in process:
  - Configurable clock options for the SDMA core and the ARM platform DMA units
    - 1:2 ratio with maximum of SDMA core running at ARM platform Peripheral Bus speed and DMA running at max DMA frequency.
    - 1:1 ratio when both SDMA core and ARM platform DMA clocks are set to the ARM platform Peripheral Bus speed.
- Peripheral bus interface for configuration register programming by the ARM platform
- The SDMA RISC engine (arithmetic and logic operations), which is referred to as the "SDMA core."
- An internal peripheral bus connected to the Shared Peripherals Bus Interface (SPBA) that enables access to up to 14 shared peripherals. SDMA supports 32-bit accesses to word peripherals and 16-bit accesses to half-word peripherals.

- The peripheral DMA unit that is hooked-up to the ARM platform Crossbar Switch to service ARM peripherals
- The burst DMA unit is able to perform burst accesses to the external memory
- All the DMA units are 32-bit AHB masters. They are connected to different buses, thus allowing concurrent accesses.

43.2 External Signals

The table found here describes the external signals of SDMA.

Table 43-1. SDMA External Signals

Signal	Description	Pad	Mode	Direction
SDMA_EVENT0	Event0 signal	GPIO1_IO02	ALT6	I
		JTAG_MOD	ALT6	
		SD1_CMD	ALT6	
SDMA_EVENT1	Event1 signal	JTAG_TMS	ALT6	I
		NAND_DQS	ALT6	

43.3 Clocks

The following table describes the clock sources for SDMA. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information. For functional information regarding module clocks, see SDMA Clocks and Low Power Modes.

Table 43-2. SDMA Clocks

Clock name	Clock Root	Description
events_sync_clk (clk)	ahb_clk_root	ARM peripheral / events clock
ips_hostctrl_clk	ipg_clk_root	Host control clock
ap_ahb_clk	ahb_clk_root	ARM platform bus clock
core_clk	ipg_clk_root	Module / Core clock
tck	-	JTAG access clock

43.4 Functional Description

The figure below shows the SDMA topology, and is composed of the following components:

- SDMA Core (SDMA Core)
- SDMA Scheduler (Scheduler)
- Functional Units:
  - Burst DMA (Burst DMA Unit)
  - Peripheral DMA (Peripheral DMA Unit)
- ARM platform Control for ARM control register access.
- Internal RAM and ROM Memory (SDMA Programming Model)
- OnCE debug Port (The OnCE Controller)

The functional unit bus provides access by the SDMA core to the DMA units. The system bus provides access to SDMA internal memory and also supports up to 14 peripherals.

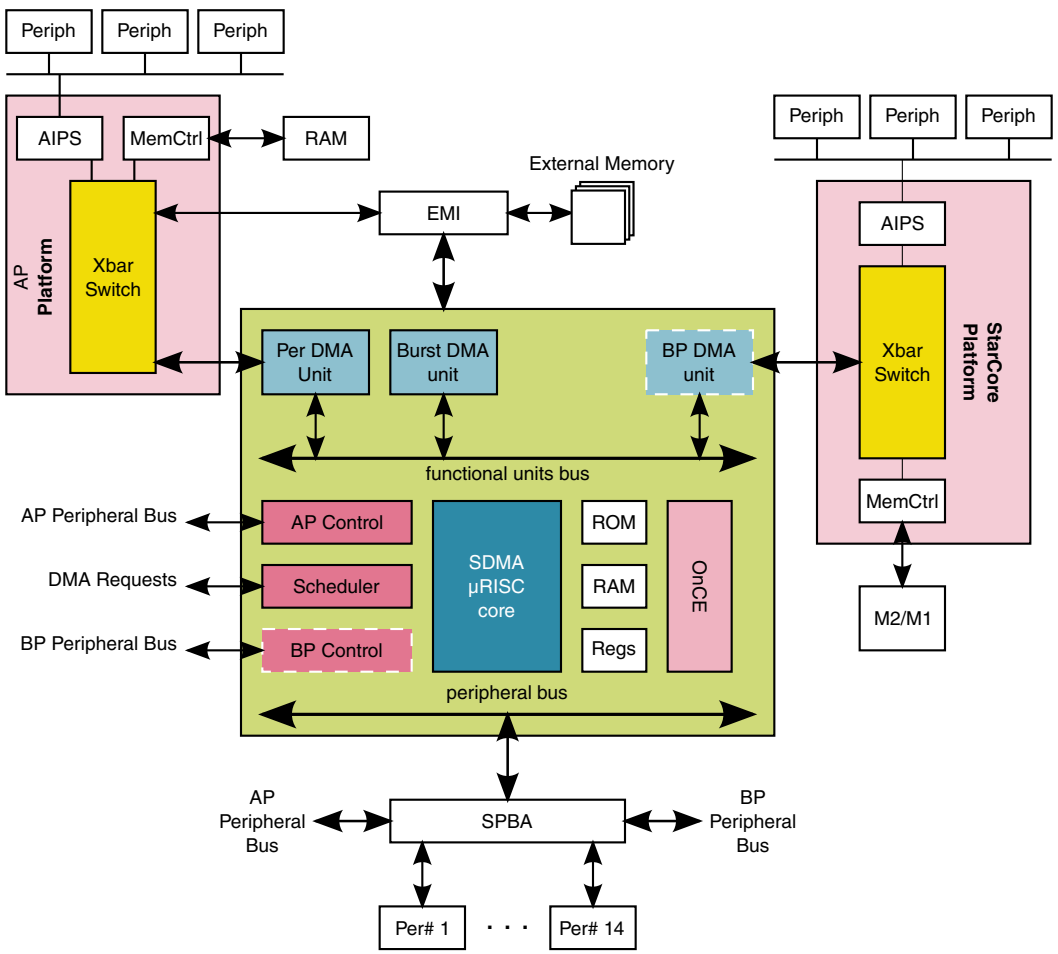


Figure 43-2. SDMA Connections

43.4.1 SDMA Core

The SDMA core is a customized RISC-like processor that is specifically developed to control DMA units and perform L1 tasks like byte-stuffing or framing.

The SDMA core incorporates on-chip debug capability using the OnCE.

The SDMA core is based on a 32-bit register architecture with 16-bit instructions. There are eight general purpose 32-bit registers, four flags (T, LM, SF, and DF), and four PCU registers (PC, RPC, SPC, and EPC) that can address 16,384 16-bit instructions.

43.4.1.1 SDMA Core Structure

The figure found here shows the structure of the SDMA core. It also shows the different registers, calculation resources, and possible data movements.

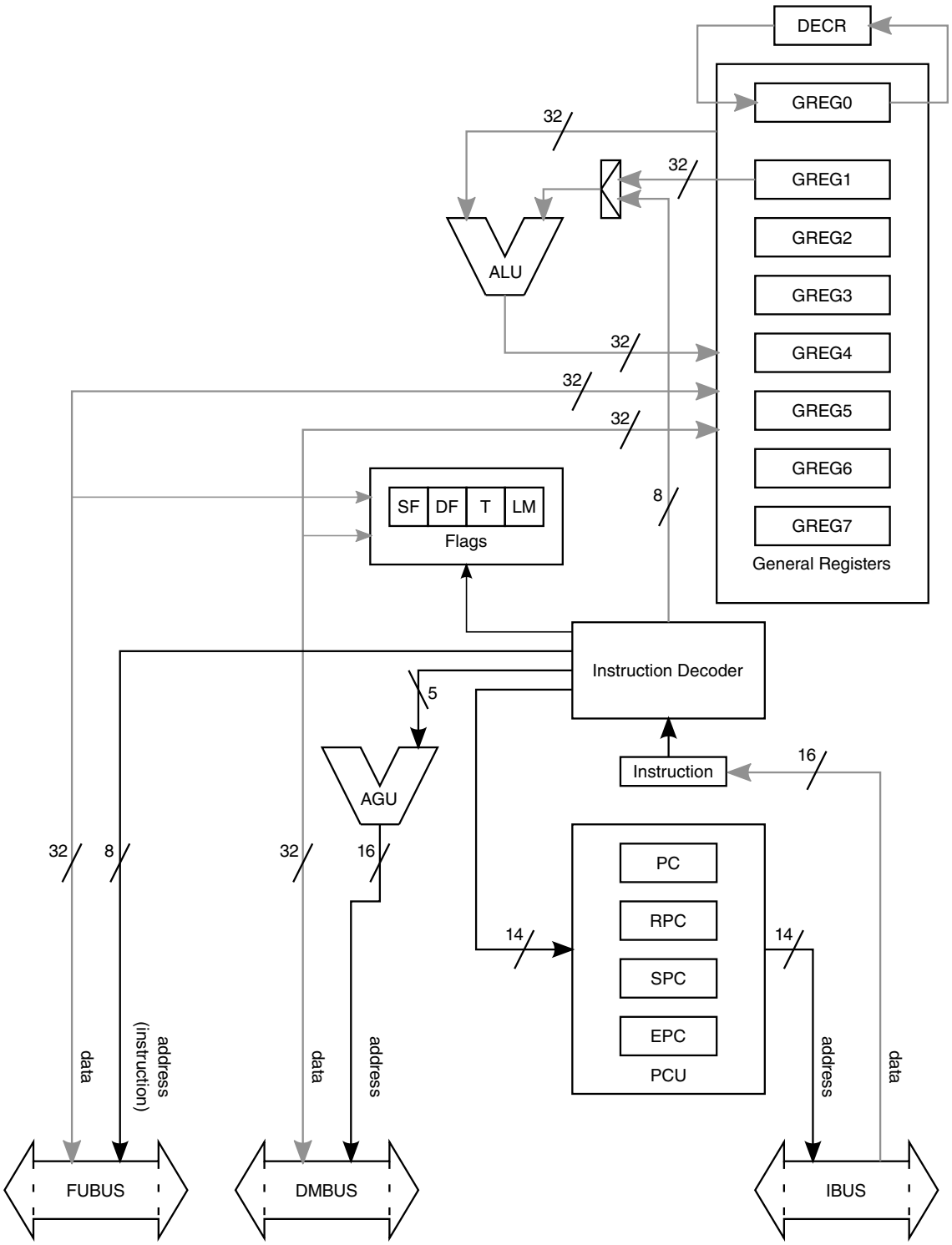


Figure 43-3. SDMA Core

- The Program Control Unit (PCU) is described in Program Control Unit (PCU). It handles the state of the core and generates the instruction fetch addresses. Instructions are retrieved from the Instruction Bus (IBUS) and stored in the SDMA

core instruction register prior to their decoding. The PCU contains the following registers:
    - The Program Counter (PC) contains the address of the current instruction.
    - The Return Program Counter (RPC) contains the address of the instruction that follows a jump to the subroutine.
    - The Start Program Counter (SPC) contains the address of the first instruction of the current hardware loop.
    - End Program Counter (EPC) contains the address of the last instruction of the current hardware loop.
- The other core registers are the general purpose registers (GREGn) and the flags.
    - The general purpose registers can be used to hold data and addresses. They can be loaded with immediate values (for example, 8-bit data that are encoded in the instruction), results of calculations that were performed with the ALU, 32-bit data that comes from the memory or peripherals via the Data Memory Bus (DMBUS), 32-bit data that comes from the DMAs via the Functional Units Bus (FUBUS) or another general purpose register. Their content can be the operands of the ALU, the data to send on either bus (DMBUS or FUBUS), or a pointer to memory (DMBUS address).
    - The general register 0 (GREG0) is also the hardware loop counter. In hardware loops, it cannot be used for any other purpose. This register uses a dedicated decrement unit (DECR) shown in Figure 43-3.
    - The flags reflect the status of operations:
        - SF and DF are set when the last load or store on either bus (FUBUS or DMBUS) received an error response.
        - LM is set when the core is executing instructions inside a hardware loop.
        - T is set when the ALU operation result was 0 or the loop counter reaches 0 (the latter is preponderant when an ALU operation is the last instruction of a hardware loop).
- The ALU has two operands: any general register and either a second general register or an immediate value. The result is always stored into the first general register. A NOP function can be utilized by moving a register's contents into itself (For example, the instruction: mov R0,R0).
- The 16-bit instructions are fetched via the instruction bus (IBUS) whose address is driven by the PC. The SDMA RAM and ROM are visible to the core as 16-bit devices through this interface.
- The memory (RAM and ROM), memory mapped registers, and external peripherals are accessed via the DMBUS. The address is always taken from a general register whose content is added to a 5-bit immediate value. This is the only available addressing mode. The DMBUS is a 32-bit data bus. Except for the peripherals that

are external to the SDMA, the address accuracy is the 32-bit word (for example, adding 1 to an address points to the next word, not the next byte).
- The functional units are accessed via the FUBUS connection. The data is exchanged with any general register, but the address (which in fact is the instruction and the selector of the functional unit) comes from an 8-bit field of the corresponding load or store.

43.4.1.2 Program Control Unit (PCU)

This part of the SDMA core is dedicated to the control of the RISC engine, as implied by the instructions that are executed. Its behavior is determined by the instruction type and the inputs of the SDMA.

It contains the PC, RPC, SPC, and EPC registers that are described in SDMA Core Structure.

43.4.1.2.1 Instruction Types

The state sequence and the delay of execution vary according to the type of the instruction. There are six possible categories of instructions, as follows:

1. Standard: Most of the instructions belong to this category, and always last 1 cycle.
2. ldf/stf: These are respectively the load and store instructions that access the functional units. They last 1+n cycles where n is the number of wait-states of the targeted functional unit.
3. ld/st: These are the load and store instructions that access the memory and peripherals. They last 1+n cycles where n is the number of wait-states of the targeted device (1 for the ROM, RAM, and memory mapped registers, 1 + the external peripheral wait-states). These instructions always last at least two cycles, but the core is able to handle them in one cycle. The first wait-state is inserted outside the core.
4. Branch: These are all the instructions that cause the Program Counter to point to another instruction other than the following one (for example, one that breaks the sequential flow). There are the absolute jumps, the conditional branches, the jump to the sub-routines, and the return from the sub-routine.
5. Loop,Modified Load or Store: The hardware loop instruction modifies the potential behavior of any load or store inside the loop (for example, when the LM flag is set). A jump may be implied after any such load or store if it received an error. The error causes an early exit of the loop, which means a jump to the instruction that follows the one that is pointed to by EPC. An additional cycle is required by the PCU to perform the jump (+1 to the ld/st/ldf/stf original execution delay). Although there is

usually an implicit jump after the last instruction of the loop when the PC goes back to SPC, this is performed at no cycle cost.

6. Done: The done, yield, or yieldge instructions are used to control channel switching. When no channel switching is performed, these instructions last a single cycle. When there is a change of channel or context switch, the delay is variable and depends on many factors (as detailed in Context Switching).

43.4.1.2.2 PCU States

The PCU state is visible through outputs of the SDMA (see Real-Time Debug Outputs) or the OnCE status register(see OnCE Status Register (OSTAT)).

The PCU state is a four-bit field that can take the values shown in the following table. Figure 43-4 shows the possible state transitions and the corresponding conditions.

Table 43-3. PCU States

Value	State	Description
0	Program	This is the usual instruction cycle.
1	Data	This state is inserted when there are wait-states during a load or a store on the data bus (ld/st type).
2	Change of Flow	This is the second cycle of any instruction that breaks the sequence of instructions (branch and done types). This state lasts only a single cycle; it is always followed by the Program state.
3	Error in Loop	This state is used when an error causes a hardware loop exit (loop-modified load or store type). This state only lasts a single cycle; it is always followed by the Program state.
4	Debug	The SDMA is stopped in debug mode.
5	Functional Unit	This state is inserted when there are wait-states during a load or a store on the functional units bus (ldf/stf type).
6	Sleep	No script is running: The core is idle after saving the last channel context.
7	Save	The context switch FSM is saving the current channel.
8	Program in Sleep	Same as Program except there is no associated channel, this state is used when instructions are executed after entering debug mode, whereas the core was in either Sleep mode.
9	Data in Sleep	This is the same as Data except there is no associated channel.
10	Change of Flow in Sleep	This is the same as Change of Flow except there is no associated channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
11	Error in Loop in Sleep	This is the same as Error in Loop except there is no associated. channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
12	Debug in Sleep	This is the same as Debug except the core was put in debug mode when no channel was active.
13	Functional Unit in Sleep	This is the same as Functional Unit except there is no associated channel.

Table continues on the next page...

Table 43-3. PCU States (continued)

Value	State	Description
14	Sleep after Reset	This shows that no script is running, and the core is idle after a reset. When a channel becomes active, no context is restored but the core starts its boot program located at address 0 (or the address available in register in Channel 0 Boot Address (SDMAARM_CHN0ADDR)).
15	Restore	The context switch FSM is restoring the next channel context.

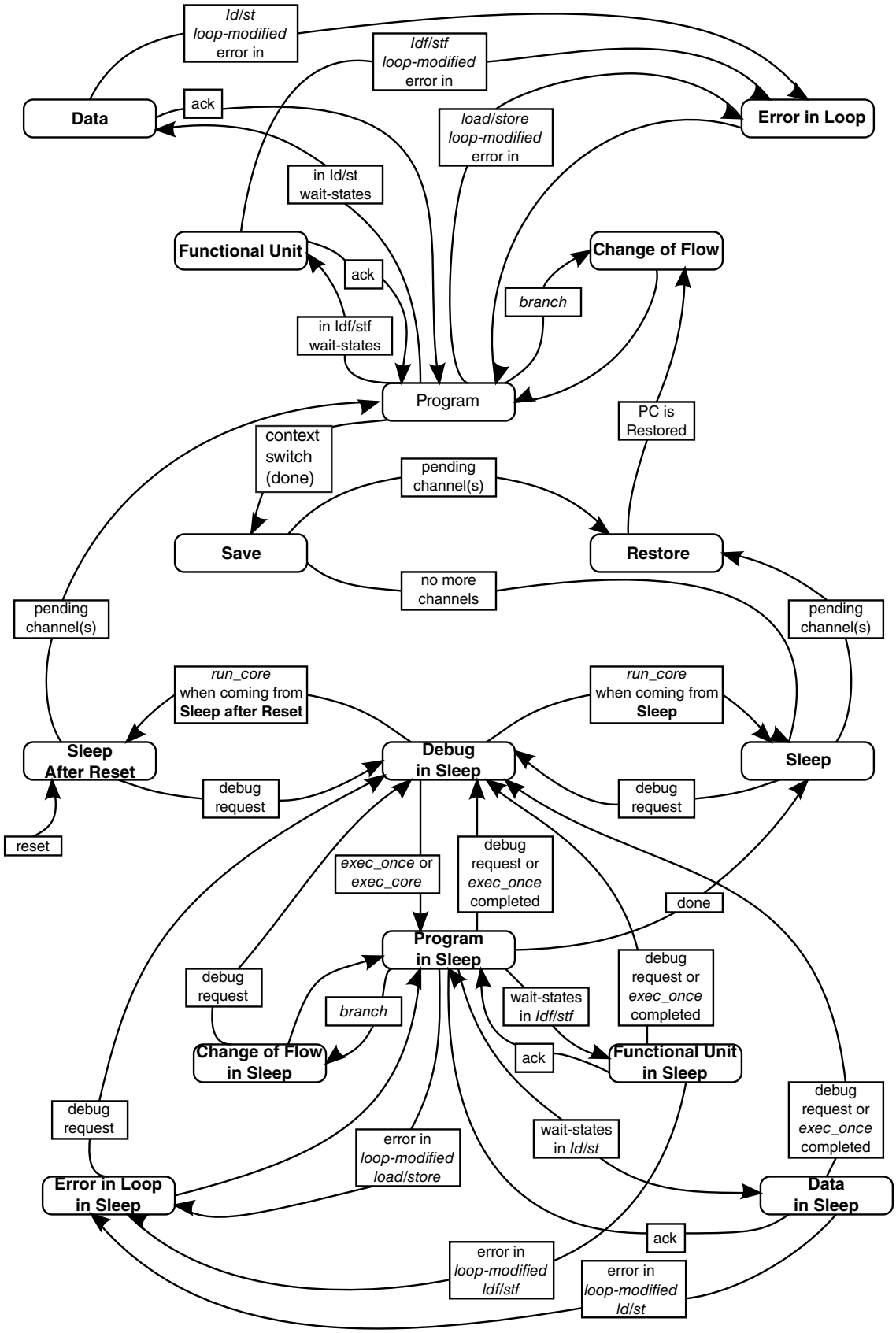


Figure 43-4. PCU State Diagram

43.4.1.3 SDMA Core Memory

The SDMA has two memory spaces: one for the instructions and one for the data. As both spaces share the same resources (ROM and RAM devices), the system bus manages possible conflicts when the core accesses the same resource for both an instruction read and a data read or write.

Program and data memory is further described in Address Space.

Instructions of 16-bit width are stored in 32-bit wide devices and can be accessed as data. The mapping is Big Endian: an even instruction address (terminated by 0) accesses the most significant part of the 32-bit data (bits [31:16]), and an odd instruction address (terminated by 1) accesses the least significant part of the 32-bit data (bits [15:0]). Instructions can be fetched out of internal ROM or RAM.

Data can be read from ROM, RAM, memory mapped registers, and external peripherals, and written to the same devices (except the ROM).

The ROM contains bootload scripts, channel scripts, and common subroutines which may be referenced by channel scripts elsewhere in the ROM or RAM.

The RAM is divided into a context area and a code space area which may be used to store channel scripts. The RAM contains undefined values after a hardware reset. Channel scripts and initial context values are downloaded into RAM using channel 0 which is reserved for bootload functions.

43.4.2 Scheduler

All channel scheduling hardware is included in the Scheduler.

43.4.2.1 Primary Functions

The scheduler is a hardware-based design used to coordinate the timely execution of 32 virtual DMA channels by the SDMA core on the basis of channel status and priority.

The scheduler performs the following functions:

- Monitors, detects, and registers the occurrence of any one of the 48 DMA requests
- Links a specific request to a channel or group of channels (channel mapping)
- Ignores requests that are not mapped to a previously configured channel
- Maintains a list of all the channels that are requesting service

- Assigns a pre-programmed priority level (1 of 7) to every channel requesting service
- Detects and flags overrun/underrun conditions

43.4.2.2 Channels and DMA Requests

43.4.2.2.1 Channels

A Virtual Channel (hereafter simply called a channel) manages a flow of data through the SDMA. Flows are typically unidirectional.

The SDMA can have up to 32 simultaneously operating channels, numbered from 0 to 31. Channel 0 is usually dedicated to control the SDMA script downloading. All the channels can be assigned by the ARM platform software.

43.4.2.2.2 DMA Requests

A DMA request is caused by externally (for example, external to the SDMA) controlled conditions (for example, UART receive FIFO reaches a threshold). The SDMA currently supports up to 48 DMA requests.

43.4.2.2.3 Mapping from DMA Requests to Channels and Priorities

A channel can stall waiting on a single DMA request. A single DMA request can awake more than one channel (in fact, any request can awake any combination of channels).

The mapping between DMA requests and channels is program-controlled. There is a storage element assigned for each of the 48 requests that contains a bitmap table of the channels that are awakened by the event.

Every channel also has a three-bit register that indicates its priority.

43.4.2.3 Scheduler Functional Description

Scheduler Overview describes the behavior of the SDMA scheduler-from the channel enabling conditions to the highest priority pending channel selection.

43.4.2.3.1 Scheduler Overview

The scheduler algorithm is built in hardware. It is provided with possibilities for the ARM platform to control its behavior.

The scheduler processes incoming DMA requests, maps detected requests to 0, one, or several channels, maintains a list of channels that are requesting service (pending channels), identifies the top priority and its associated channel, and selects the next active channel when the current channel yields.

The following figure shows a functional overview.

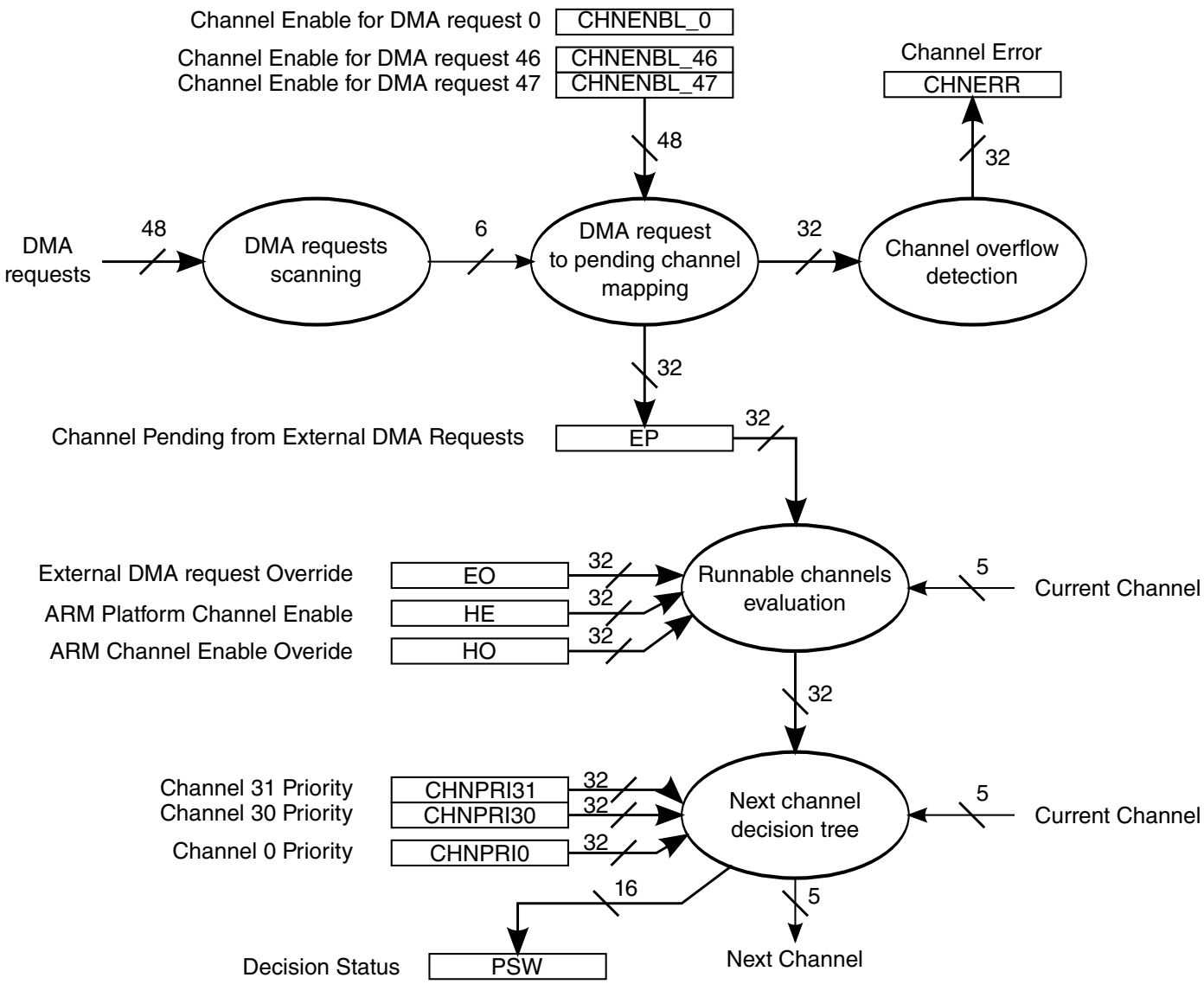


Figure 43-5. SDMA Hardware Scheduler

43.4.2.3.2 DMA Requests Scanning

The scheduler contains a 48-bit edge detection device that detects the rising edge of every DMA request and transmits the request number to the next stage.

The DMA requests are assumed to be generated on the same reference clock as the SDMA core clock; they are detected as soon as the signal goes from a 1-to-n-cycles low state to a 1-to-m-cycles high state.

This system is able to detect single-cycle pulses as well as level-based DMA requests such as a FIFO threshold crossing. In this case, the SDMA provides a memory mapped register that can be used by the channel script to monitor the DMA requests lines, and thus determines whether the data transfer is done or not done, and then continues with the transfer or closes the channel.

When several DMA requests are detected at the same time, they are forwarded to the next scheduler stage at the rate of one request per cycle. No request is lost.

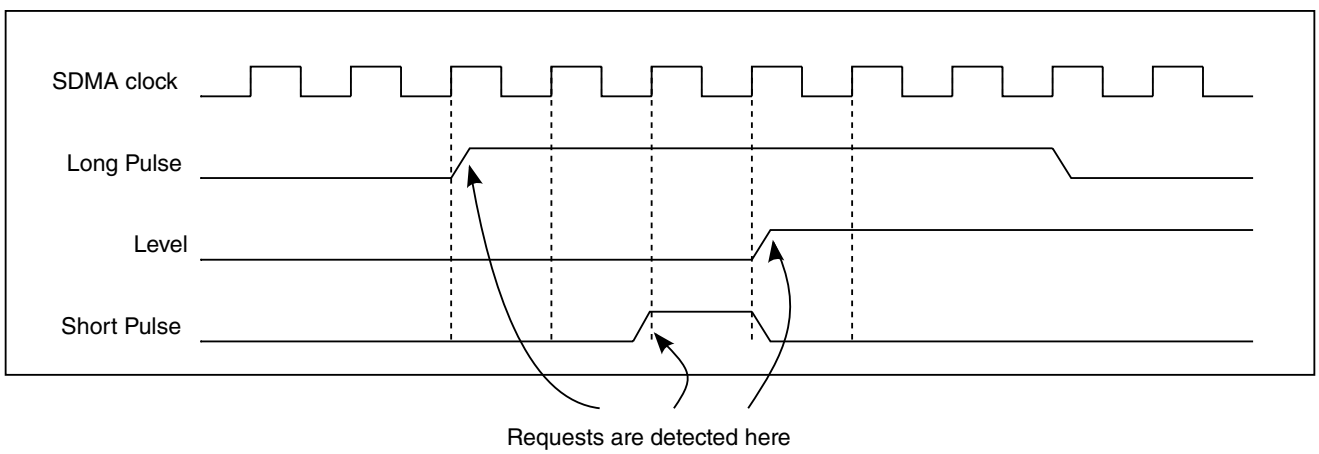


Figure 43-6. Examples of Valid DMA Requests

The DMA request inputs are connected to various sources that depend on the SoC. The exact list of DMA request inputs and their associated number is available in each respective project-specific chapter.

43.4.2.3.3 Mapping DMA Requests to Pending Channels

Whenever a DMA request is detected by the first stage, its number is used in the second stage to determine the channels that have to be activated.

This is performed with an array of 48 registers that are 32 bits wide: There are 48 Channel Enable Registers (CHNENBLn), one register per DMA request. The DMA request number selects the Channel Enable Registers, and every bit of this 32-bit register indicates that the corresponding channel must be activated when it is a 1.

This information is passed on the EP register. For every bit of the Channel Enable Register that is set, the corresponding bit of the EP register is also set, and the remaining bits of EP are left unchanged. The transformation of EP is summarized by the following equation:

EP = EP or CHNENBLn

The EP register is used to know which channels require service because they received a DMA request.

Typical contents of the CHNENBLn registers are all 0s, except for a single bit set. For example, a DMA request triggers one channel, but all 0s or several 1s are possible. One DMA request could activate several channels, and the channel execution sequence can be controlled by the channel priorities and numbers, as explained in the next sections. The following table illustrates an example configuration.

NOTE

From the table, the DMA request 0 is programmed to simultaneously trigger channels 0, 1, and 31. Also, DMA requests 30-47 are not used in this example. The remaining channels 2 to 30, are configured to be triggered by DMA requests 29 to 1, respectively.

Table 43-4. Channel Enable RAM Programming Example

DMA Request Number	Channel																																
	31																																0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 43-4. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																														
	31																														0
11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 43-4. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																													
	31																													0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

43.4.2.3.4 Channel Overflow

A channel overflow occurs when a DMA request requires service from channel *n* by setting bit *n* of the register EP, but this bit is already set, meaning channel *n* is already pending. This can come from an overrun/underrun condition.

This detection is possible only when the DMA requests are pulses, because a level-based DMA request stays high until it is serviced, even though an underrun or overrun condition occurs, thus preventing another edge detection of the DMA request.

The channel overflow information is saved in the 32-bit CHNERR register (1 bit per channel). You can configure the SDMA to trigger an interrupt to the ARM platform when there are 1s in CHNERR. Every bit of CHNERR is masked with the corresponding bit of INTRMASK and if it gives a 1, the corresponding bit of INTR is set, triggering the interrupt.

43.4.2.3.5 Runnable Channels Evaluation

The EP register is used in conjunction with several other 32-bit registers to determine the channels that are runnable.

Registers EO, DO, HO and HE, are controlled by the ARM platform. EP is controlled by the DMA requests and their mapping to channels.

Several channels may be runnable at any given time. The i^{th} channel is runnable if (and only if) the condition below is true:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

After reset, the HE[i], HO[i], EP[i], and EO[i] bits are all cleared whereas the DO[i] bits are all set. The functions associated with DO are not available for this device. When DO[i] is set, the scheduler condition becomes:

(HE[i] or HO[i]) and (EP[i] or EO[i])

The registers in these equations are controlled as follows:

- ARM platform (host) channel enable flag HE[i] may be set or cleared by the ARM platform with the HSTART and STOP_STAT registers. It can also be cleared by the i^{th} channel script.

  Typical usage is for the ARM platform to set this flag to activate the channel. The flag is cleared by the SDMA core when the transfer is done.

- Externally triggered channel pending flag EP[i] is set by the scheduler when the channel was activated by a DMA request. It can be cleared by the i^{th} channel script.
- The ARM platform channel override flag HO[i] may be set or cleared by the ARM platform. When set, it enables the i^{th} channel to run without the involvement of the ARM platform.

  Typical usage is for the ARM platform to set this flag for channels that do not need ARM platform supervision such as channels that are controlled by DMA request events (EP).

- DO should always be set to 1 so that the runnable channel evaluation considers only HO, HE, EP, and EO.
- Externally triggered channel override flag EO[i] may be set or cleared by the ARM platform. When set, it prevents the i^{th} channel from stopping and stalling on incoming peripheral DMA requests. This is the case when the channel is not handling data transfers with peripherals (for example, a memory to memory transfer).

The SDMA can clear the HE[i], and EP[i] bits by means of a done or notify instruction. The done instruction causes a reschedule; thus, enabling another channel to preempt the current one, while the notify instruction does not. The done and notify instructions can clear either HE[i] or EP[i] (never more than one at a time).

Table 43-5. Runnable Channel Selection Control

Register	Set by	Cleared By
HO	Write to HOSTOVR register	Write to HOSTOVR register
HE	Write to HSTART register	Write to STOP_STAT register or by the channel script with the done or notify instructions.
DO	Write to DSPOVR register	Write to DSPOVR register
EO	Write to EVTOVER register	Write to EVTOVER register
EP	Set by external DMA request event input.	By the channel script with the done or notify instructions

43.4.2.3.6 Next Channel Decision Tree

The next channel number is computed from the runnable channels list, the current channel number, and their respective priorities.

It is re-evaluated every cycle, but is only used when the current channel yields or terminates by executing a yield, yieldge, or done instruction.

The decision tree is based on the selection of the runnable channel that has the highest priority.

The highest priority channel is selected according to the following rules:

- Runnable channels are sorted by priority.
- If one of the channels with the highest priority had been preempted by a channel with a higher priority, but did not want to yield to a channel of the same priority (for example, it executed a yield, not a yieldge), it is elected as the next channel.
- The channels that belong to the highest priority group are sorted by their number and the channel that has the highest number in this group becomes the next channel. For example, if priorities are the same, channel 31 will be selected before channel 30.

When the current channel requires a reschedule with a yield(ge) or a done instruction, the context switch decision is based on the instruction parameter, the current channel number and priority, and the next channel number and priority. The possible cases are all listed in the following table. The grayed cells correspond to unusual cases that should not occur with a typical usage of the SDMA.

Table 43-6. Channel Switching Decision with a yield, yield(ge), or done

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
yield (done 0)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Current
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
yieldge (done 1)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Next[1]
			Current < Next	Next[1]

Table continues on the next page...

Table 43-6. Channel Switching Decision with a yield, yield(ge), or done (continued)

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
done (done>1)	Not runnable	Not runnable	none	none[2]
	Runnable	Not runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)
	Not runnable	Runnable	none	Next[1]
	Runnable	Runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)

1. Current channel script execution is stopped, its context is saved; the next channel context is restored and its script execution resumes
2. Current channel context is saved and SDMA enters IDLE mode
3. Current channel context is saved, then restored, and the current channel script resumes execution

Finally, when the SDMA is in IDLE mode and a runnable channel is elected as the next channel, its context is immediately restored and the script execution resumes.

The *combinatorial-decision* tree supports dynamic modifications of the EP, EO, HE, HO, and DO flags as well as dynamic modifications of the channel priorities. The propagation times are detailed in Scheduler Pipeline Timing Diagram.

The decision tree status is available in the PSW register, which is continuously updated. It contains the next channel priority, the next channel number, the current channel priority, and the current channel number. When a priority is read as 0, it means the channel is not runnable.

A few examples of decisions are presented below:

- Channel 31 is running with priority 5, channels 13 and 24 are pending with the same priority 5; channel 24 is eligible as the next channel since 24 > 13.
- Channel 31 is running with priority 7, channels 13 and 24 are pending with priority 5; channel 31 is the next channel because its priority is greater than the other pending channels.
- Channels 7, 23, and 29 are pending with the same priority. Channel 7 is active and runs a yieldge; it is preempted by channel 29. After a period of time, channel 29 runs a yieldge, it is then preempted by channel 23 that is the selected channel since channel 29 is the current channel. Later, channel 23 runs a yieldge and is preempted

by channel 29. Channels 23 and 29 will go on switching after every yieldge until one of them terminates. It is only at that point that channel 7 becomes eligible again.

- Channel 11 is running with priority 3, and channel 15 is pending with priority 4. When the channel 31 script executes a yield instruction, it gets preempted by channel 15; then channels 6 and 18 with priority 3 become pending. Because channel 11 was preempted after executing a yield and there is no pending channel with a strictly greater priority, it is eligible as the next channel (although its number 11 < 18).

43.4.2.3.7 Scheduler State Diagram

The Figure 43-7 summarizes the behavior of the SDMA scheduler with details about the exact mechanism of the priority decision tree. It is important to understand the scheduler is a hardwired pipeline, which means all the stages are performed simultaneously every cycle, but a change on any given stage is reflected on the next stage after the delays presented in Scheduler Pipeline Timing Diagram.

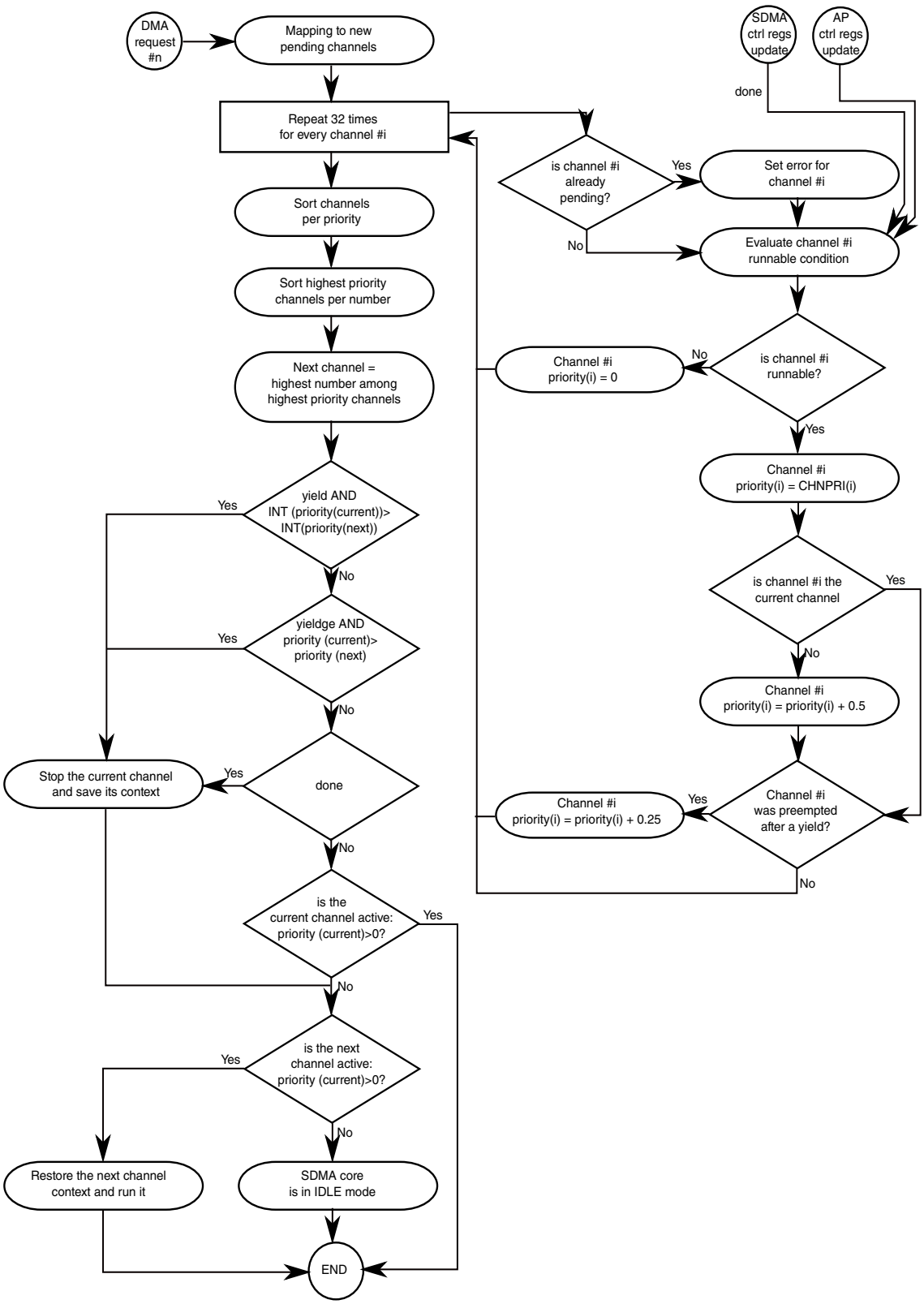


Figure 43-7. Scheduler State Diagram

43.4.2.3.8 Scheduler Pipeline Timing Diagram

The SDMA scheduler process of DMA-request and control-register modifications is not immediate.

The figure below shows the exact delays of all the tasks. The reference clock is the SDMA core clock.

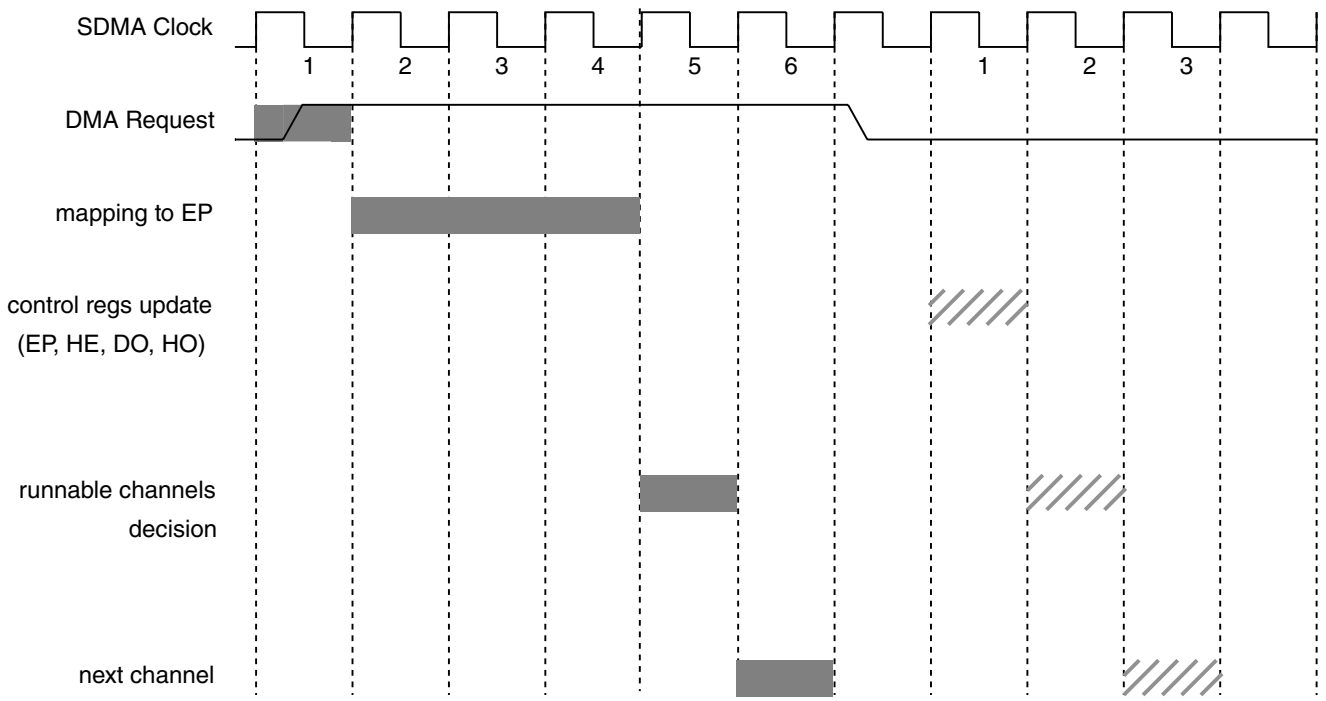


Figure 43-8. Scheduler Timing Diagram

Two numbers can be inferred from this timing diagram. First, it takes six SDMA core clock cycles to update the next channel from a DMA request. Second, it takes three SDMA core clock cycles to update the next channel from a direct modification of the condition registers (EP, DO, HE, or HO) by any processor. The processors that can modify these bits include SDMA with a done instruction or the ARM platform with a write access through the corresponding control port on their respective peripheral bus).

43.4.2.3.9 Channel-DMA Request Mapping

The 48 DMA request inputs to the SDMA scheduler are listed in project-specific chapters. Refer to the respective chapters for this information.

43.4.2.3.10 Examples: How to Start a Channel

A channel can be started when the following equation is true for channel *i*:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

Once this equation is true, the scheduler can start this channel according to the priority of all pending channels. Several examples of configuration are listed below:

1. To start a channel triggered by ARM platform software:
   - Initially, configure HO[i]=0, DO[i]=1, and EO[i]=1 using registers indicated in Table 43-5.
   - ARM platform software triggers the channel by writing to the HSTART register to set HE[i]=1, thereby setting the above equation true.
2. To start a channel triggered by DMA request event.
   - Initially, configure HO[i]=1, DO[i]=1, and EO[i]=0 using registers indicated in Table 43-5.
   - The DMA request is asserted to trigger the channel by setting EP[i]=1, which makes the above equation true.

43.4.2.4 Context Switching

On execution of a done or yield(ge) instruction, the current channel may be changed either because it has finished (which necessarily happens when the done instruction is executed), or it was preempted by a higher priority channel (which is possible but not systematic when the yield(ge) is executed).

Upon a channel change the SDMA goes through a context switch procedure.

When the current channel yields or ends, the context for that channel is saved into the context RAM locations for that channel. When the next channel starts running, its context is first restored from RAM.

Since context RAM is not yet initialized by reset, there will be no context restore at the beginning of the first channel (bootload channel) run after reset. It is expected that the bootload channel will be used to initialize the context for all other channels. When the bootload channel finishes running or yields, SDMA will enter its SAVE state and save that channel's context into RAM. Then, if the bootload channel is called again later, the context will be restored from RAM when the channel starts again.

The context structure for each channel is defined in Context Switching-Programming and Table 43-11. There will be one context area reserved for each channel. When a channel ends or yields, the SDMA core registers are automatically saved into the context RAM and later restored from the context RAM when the channel is next run. The total RAM space reserved for 32-channel contexts is either 3K or 4K depending on whether the SMSZ bit is set in the CHN0ADDR register, which enables an additional 8 words of scratch RAM for each context.

43.4.2.4.1 Context Switch Modes

The exact procedure to save the context of the old channel, and to restore the context of the new channel depends on the context switch mode selected by the ARM platform in the CONFIG control register.

The following are the context switch modes:

- By default, the "dynamic" context switch is set. This mode provides the most efficient context switch for an average of eight cycles to stop the current channel, save its context, restore the next channel context, and resume its execution. It consists of saving modified registers of the current channel in the background (for example, during the channel execution)-which leaves very few registers to save when the switch is decided-resuming execution of the next channel as soon as possible (for example, when the minimal set of registers is restored), and continuing the restore phase during this execution.
- In "dynamic with no loop" mode, the same principle is followed except the modified registers are only saved in the background when the loop flag is not set. This mode offers almost the same effectiveness as the previous one, but it prevents the system from accessing the RAM during loops to save power. This is the recommended mode for an efficient context-switch when the loop bodies are short.
- In "dynamic power" mode, no background saving is performed, which reduces power consumption to the minimum. The modified registers are only saved when the context switch starts. The restore phase is the same as before. This is the mode that achieves the optimal power consumption at the cost of a slower context-switch.
- In a "static" context switch, all the registers are saved when a context switch is decided, and all the registers are restored before starting the execution of the new channel. This mode enables a predictable behavior of the context switch since all the registers are restored prior to the channel start and all registers are saved after the channel termination.

NOTE

Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized during the context SAVE phase when the channel is done or yields. Subsequent calls to the same channel or different channels may use any of the dynamic context modes. This will ensure that all context locations for the bootload channel are initialized, and prevent undefined values in context RAM from being loaded during the context restore if the channel is re-started later.

43.4.2.4.2 Context Switch Procedure

The Program Control Unit goes into the *save* state, the current context is spilled into memory, and the next channel context is restored according to the context-switch mode that was selected by the ARM platform.

The context switch procedure is as follows:

1. Load the current context's spill base address.
2. Spill the modified registers of the current channel to memory according to the selected context switch mode while the channel is running.

   On a done or yield(ge) that causes the channel preemption, the PCU goes into the *save* state. In *static* mode, all the registers are saved; whereas, in either *dynamic* mode, the registers that were modified but not yet saved are then saved, and the PCU registers and flags are finally saved.

3. Put the SDMA core into *sleep* and wait for new channels to be serviced. This step is skipped if there are pending channels when the current channel is saved.

   As soon as there is at least one pending channel, the PCU goes into its *restore* state to restore the context of the channel that was elected by the scheduler.

   Once a channel is elected, it remains the current channel until its script requests a rescheduling operation with a done or yield(ge) instruction. That means the current channel cannot be modified by the ARM platform, even if it is no more runnable or if its priority is modified.

   The ARM platform can however force a reschedule by writing the corresponding bit in the CONFIG register, which has the same effect as if the script had executed a done instruction. That feature should only be used to stop the SDMA in emergency cases.

4. Load the context base-address of the new channel.

   In "static" mode, all the registers are restored. In either "dynamic" modes, only the PCU registers are restored.

   The new channel is running. In "static" mode, no more activity regarding context restoring or saving is performed. In either "dynamic" modes, the registers are restored in the background every time an access to the context RAM is possible, and priority is given to restoring the registers that are required by the next instruction to be executed. When a register has not been restored and the next instruction needs it, this instruction gets stalled until the register was restored.

In "dynamic" and "dynamic with no loop" modes, background saving of dirty registers is performed every time an access to the context RAM is possible and allowed by the context switch mode.

NOTE

The contents of a channel context space in the context RAM depends on the selected context switch mode. In "dynamic" and "dynamic with no loop" modes, the contents of the context RAM tend to match the contents of the SDMA registers (except for the PCU registers and flags that are never saved in the background). In "dynamic power" and "static" modes, the contents of the context RAM remain unchanged until the channel terminates with a done or gets preempted.

43.4.2.4.3 Context Map in Memory

Refer to Context Switching-Programming.

43.4.3 Functional Units

The functional units are small systems that are used by the SDMA core to handle data transfers between the core and a bus domain external to the SDMA.

The SDMA core is able to control and exchange data with these systems by sending instructions and reading or writing data from/to the functional units' registers via the FUBUS. This is done with the ldf and stf instructions.

The following sections provide introductions to the available functional units. Functional Units Programming Model provides descriptions the functional units' behaviors.

43.4.3.1 Burst DMA Unit

The burst DMA unit enables the SDMA core to perform data transfers to and from the ARM platform memory.

It is optimized for accessing SDRAM-like devices. It does not provide control to assign a privilege level to the DMA access. The burst DMA unit provides the SDMA with means to do the following:

- Perform up to 8-beat read and write bursts to the ARM platform memory, which optimizes throughput when accessing SDRAM-type devices because of an internal, 36-byte FIFO
- Access the ARM platform memory at once or twice the SDMA core frequency
- Copy data from one ARM platform memory location to another ARM platform memory location at the ARM platform bus speed, which provides a very high throughput
- Control the method for addressing the ARM platform memory (automatic increment of addresses or frozen addresses-the former aimed at accessing RAM-like memory and the latter aimed at accessing single-address FIFOs)
- Enable or disable automatic prefetch when reading data from the ARM platform memory. When the prefetch mode is selected, the burst DMA automatically triggers external bursts to fill its FIFO without waiting for the SDMA core to request the corresponding data, greatly improving throughput.
- Rely on the DMA to automatically flush its FIFO content when there is enough data to generate an 8-beat burst to the ARM platform memory. Or, it forces a flush when a data transfer must terminate.
- In the former case, the SDMA core may only be stalled when it tries writing data and there is not enough room left in the FIFO. In the latter case, the core is stalled until the data is effectively written to the ARM platform memory.

  In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory. This error status is retrieved by a later access to the burst DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory is caught.

- Handle address alignment issues between the ARM platform memory map and the SDMA core data. This enables the core to read or write 32-bit data from the burst DMA, whereas the corresponding ARM platform address is not 32-bit aligned. This drastically improves the SDMA scripts' efficiency since the same loop that transfers 32 bits at a time can be used regardless of the start and end addresses in the ARM platform memory space.

This unit structure and registers are described in Burst DMA Structure and Burst DMA Registers.

43.4.3.1.1 Burst DMA Structure

The burst DMA is essentially made up of a 36-byte FIFO, address registers, and a controlling state-machine. The 36-byte FIFO enables eight-word buffering with address alignment, and the state-machine manages clock adaptation when required.

The burst DMA is depicted in the figure below.

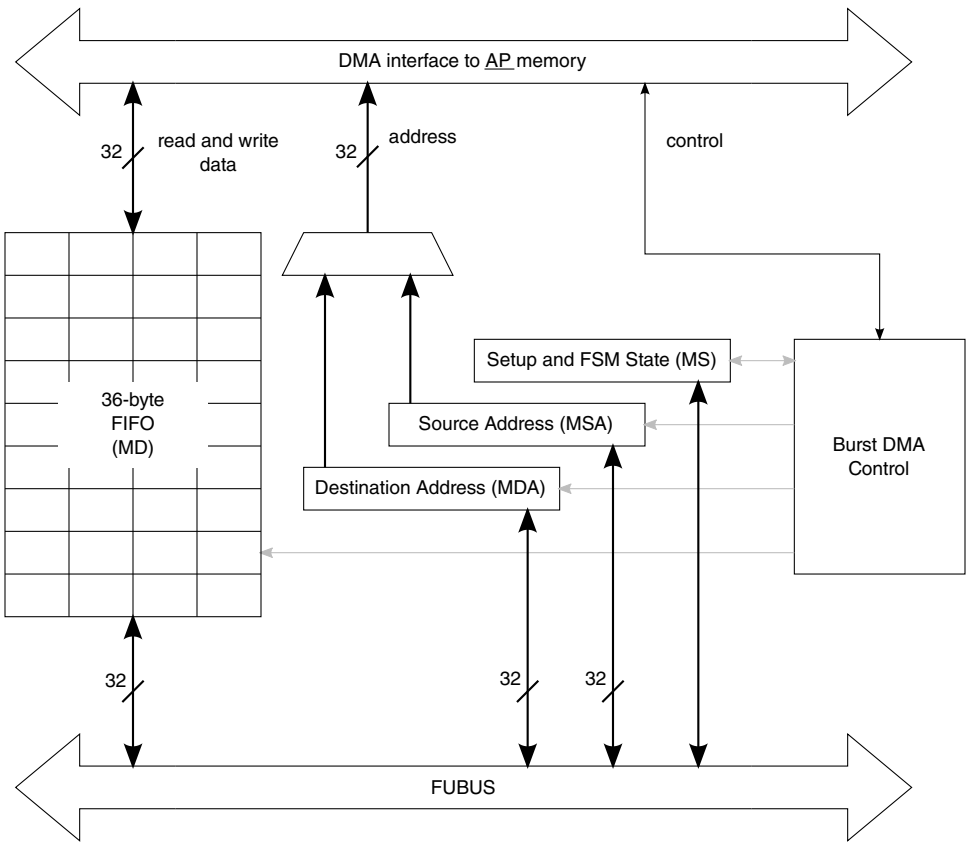


Figure 43-9. Burst DMA Structure

43.4.3.1.2 Burst DMA Registers

There are four registers, as follows, that may be accessed from the SDMA core:

- MSA (Memory Source Address) - Holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads new data from the FIFO.
- MDA (Memory Destination Address) - Holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes new data into the FIFO.
- MD (Memory Data) - Labels the 36-byte FIFO access point: Reading a byte, halfword, or word from MD respectively retrieves the first 1, 2, or 4 bytes of the

FIFO (for example, the bytes that were stored first by the DMA state-machine when transferring data from the ARM platform memory).

- When the FIFO does not hold as many bytes as required by the SDMA core, the core is stalled until the missing bytes are read from the ARM platform memory. In the case of prefetch mode, the DMA controller decides when it should start a burst to ARM platform memory in order to reduce the risk to not have the required data for the future accesses of the core. When there is no prefetching, a burst is triggered when the required data is not available in the FIFO.

  Writing a byte, halfword, or word to MD stores 1, 2, or 4 bytes, respectively, at the end of the FIFO (for example, these bytes are transmitted to the ARM platform memory after all the other bytes that were previously stored in the FIFO). When the FIFO does not have enough room left to hold the written data, the SDMA core is stalled until a sufficient amount of FIFO contents are flushed out to the ARM platform memory. Flushing is decided by the DMA controller when there are enough bytes in the FIFO to perform the largest allowed burst to ARM platform memory (the exact size depends on the burst start address and the AHB 1 Kbyte boundary rule). However, the SDMA core has the ability to force the flushing operation at any time, for example, when at the end of the data transfer, prior to channel closure.

- MS (Memory Setup) - Contains the state of the burst DMA control, the two flags that define whether each address register is incremented after every access to the external memory, and another flag that is set when a bus error occurred.

43.4.3.1.3 Burst DMA Data Transfers

Three typical usages have been identified that involve the burst DMA: the data transfer startpoint, the endpoint, or both.

Every case requires a different procedure, as listed in the following sections:

43.4.3.1.3.1 Data Retrieval from the ARM platform Memory

The following steps retrieve data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the source address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the source address register itself (MSA).
- Read data from the FIFO using the *ldf MD* instruction as many times as needed. If an error occurred during the fetch from ARM platform memory, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from the FIFO.

43.4.3.1.3.2 Storing Data Into the ARM platform Memory

The following steps store data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the destination address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the destination address register itself (MDA).
- Store data into the FIFO using the *stf MD* instruction as many times as needed.
- When the transfer is finished and if the DMA worked in automatic flush mode, force the flush of the FIFO. This instruction is stalled until all the FIFO data is effectively sent to the ARM platform memory and the error status of the transfer is available in the DF flag.

43.4.3.1.3.3 Transferring Data Between Two ARM platform Memory Locations-Burst DMA Unit

The following steps copy data between two ARM platform memory locations using the burst DMA unit:

- Set up the MS flags to reflect the modes for the source and destination addresses (all the combinations are possible), then initialize the source address register (MSA) and the destination address register (MDA). Both addresses must be word-aligned.
- Use as many *stf MD* instructions with the *COPY* flag as needed. Every instruction triggers a burst read of a given number of words from the source address (this number is provided to the burst DMA via the SDMA core general purpose register, which is referenced in the *stf* instruction). Once all the data is loaded into the FIFO, the DMA empties it with a write burst of the same count to the destination address. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the burst DMA to check the error status.

43.4.3.2 Peripheral DMA Unit

The peripheral DMA unit is the second functional unit that connects the SDMA to the ARM platform memory.

Unlike the burst DMA, it does not support burst transfers and is optimized for accessing peripherals. It does not provide control to assign a privilege level to the DMA access. Its feature list comprises the following:

- Access to the ARM platform peripherals or memory at once or twice the SDMA core frequency

- Data copy from one ARM platform memory location to another ARM platform memory location at memory bus speed, improving throughput
- Control of the method for addressing the ARM platform memory (automatic increment or decrement of addresses or frozen addresses, the first ones aimed at accessing RAM-like memory and the last one aimed at accessing single-address FIFOs)
- Selectable automatic prefetch when reading data from the ARM platform memory. In prefetch mode, the peripheral DMA automatically fetches another data-without waiting for the SDMA core to request it-when its data register is empty, which improves the throughput
- Selectable automatic flush. In this mode, the SDMA core may only be stalled when it tries writing data and the previous write operation is not finished yet; whereas, in forced flush mode, the core is stalled until the data is effectively written to the ARM platform memory.
- In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory or the peripheral. This error status is retrieved by a later access to the peripheral DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory has been caught.

  This unit structure and registers are described in Peripheral DMA Structure and Peripheral DMA Registers.

43.4.3.2.1 Peripheral DMA Structure

The peripheral DMA is made up of a 32-bit data register, two address registers, and a controlling state-machine. The state-machine manages clock adaptation, when required.

It is shown in the following figure.

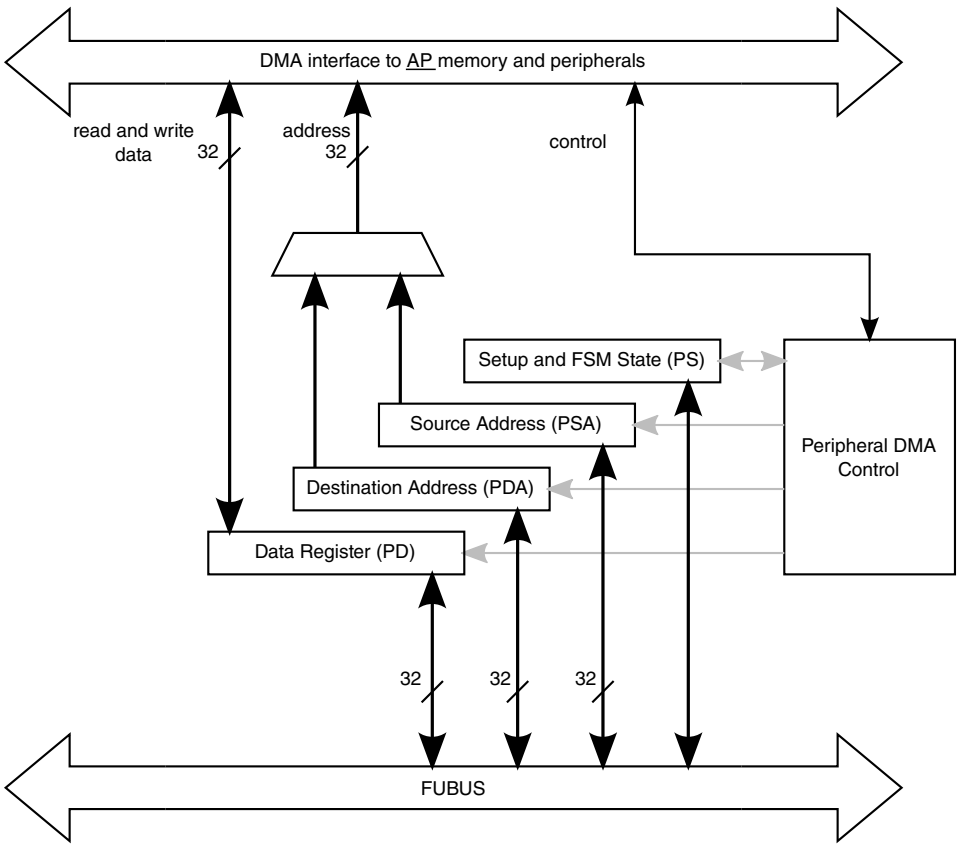


Figure 43-10. Peripheral DMA structure

43.4.3.2.2 Peripheral DMA Registers

According to Figure 43-10, the peripheral DMA has four registers that may be read or written by the SDMA core:

- *PD (Peripheral Data)* is the DMA 32-bit data register.
- *PSA (Peripheral Source Address)* holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads a new data from PD.

- *PDA (Peripheral Destination Address)* holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes a new data into PD.
- *PS (Peripheral Setup)* contains the state of the peripheral DMA control, two configuration fields that define the way address registers are modified after every data access, two additional configuration fields that define the data size to access the source and destination devices, and another field that contains the latest transfer error status.

43.4.3.2.3 Peripheral DMA Data Transfers

There are three typical usages that involve the peripheral DMA, whether it is the data transfer start-point, endpoint, or both.

Every case requires a different procedure, as described in Data Retrieval from the ARM platform Memory or Peripheral, Storing Data into the ARM platform Memory or Peripheral, and Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit.

43.4.3.2.3.1 Data Retrieval from the ARM platform Memory or Peripheral

The following steps retrieve data from ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the source (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the source address register itself (PSA) with an address that is aligned to the programmed data size.
- Read data from PD using the ldf PD instruction as many times as needed. If an error occurs during the fetch from the ARM platform memory or peripheral, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from PD.

43.4.3.2.3.2 Storing Data into the ARM platform Memory or Peripheral

The following steps store data to ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the destination (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the destination address register itself (PDA) with an address that is aligned to the programmed data size.

- Store data into PD using the *stf PD* instruction as many times as needed.
- When the transfer is finished and if the peripheral DMA worked in automatic flush mode, force the flush of PD. This instruction is stalled until PD contents are effectively sent to the ARM platform memory or peripheral, and the error status of the transfer is available in the DF flag.

43.4.3.2.3.3 Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit

The following steps copy data between two ARM platform memory locations using the peripheral DMA unit:

- Set up the PS fields to reflect the modes and data size for the source and destination addresses (all the combinations of addressing modes are possible, but both data sizes must be identical), then initialize the source address register (PSA) and the destination address register (PDA). Both addresses must be aligned with the programmed data size.
- Use as many *stf PD* instructions with the *COPY* flag as needed. Every instruction triggers a single read from the source address; a single write of the received data immediately follows. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the peripheral DMA to check the error status.

43.4.4 SDMA Security Support

The SDMA provides support to SDMA software to block unauthorized updates to the scripts in RAM.

SDMA supports the following Security modes:

- Open Mode: has full control to load scripts and context into SDMA RAM. This is the default mode.
- Locked Mode: The ARM platform loads scripts and channel contexts at startup when it is still executing known safe software. When finished, it locks the SDMA to prevent further updates to RAM and selected registers. More details described in Locked Mode.

43.4.4.1 Locked Mode

The LOCK bit in the SDMA_LOCK register provides support for SDMA scripts to freeze RAM contents after the initial bootload routine to prevent future unauthorized updates to SDMA RAM.

After initial RAM contents are uploaded, ARM platform software can set the LOCK bit to secure the RAM contents to prevent future updates by an unauthorized. After the LOCK bit is written with a '1', the SDMA is "locked" until reset.

The LOCK bit can be read in the SDMA's internal memory map in the LOCK register (see Section SDMA LOCK (SDMAARM_SDMA_LOCK)). SDMA scripts which load information into RAM can check the value of the LOCK bit to determine if an upload to RAM is allowed. If not allowed, the script can refuse to allow the request to copy data into the RAM to continue. The exact use of the LOCK bit in SDMA scripts for security control will be described in SDMA software documentation (see SDMA Scripts).

While SDMA is locked, attempts to write to the SDMA_LOCK, CHN0ADR, ILLINSTADDR, and ONCE_ENB registers will be ignored. All registers remain readable. Writes to other registers are still allowed.

Once the SDMA is locked, the LOCK bit can only be cleared by a reset. A hardware reset will always clear the LOCK bit. A software reset initiated by writing to the RESET register will only clear the LOCK bit if the SRESET_LOCK_CLR bit in the SDMA_LOCK register is set. Since SDMA_LOCK register cannot be updated if SDMA is locked, the SRESET_LOCK_CLR bit must be configured before setting the LOCK bit. The SREST_LOCK_CLR bit will also be cleared by resets that clear the LOCK bit.

The SDMA RISC core uses the ILLINST and CHN0ADDR registers as pointers to determine where to jump to after an illegal instruction or upon boot after a reset. The LOCK bit prevents updates to these registers to protect against unauthorized changes to these pointers.

While SDMA is locked, the ONCE_ENB register cannot be written to prevent the OnCE under ARM platform control from being used to gain access to SDMA internal memory. If ARM platform control of the OnCE is enabled before setting the LOCK bit, the ARM platform can use the ONCE for debug purpose after LOCK is set.

43.4.5 OnCE and PCU Debug States

The SDMA has two different debug modes in which the OnCE performs debug instructions.

Refer to Figure 43-4 for an example of the PCU states in debug. The following are the two debug states:

- When a channel is running (that is, when CCR and CCPRI are different from 0, which can be read in the PSW register), SDMA can execute a SoftBkpt instruction from the channel script or receive a debug request. When either happens, the SDMA enters its "Classical" *Debug* state, which is described in OnCE and Real-Time Debug.
- When a channel is not running, the SDMA can be in *Sleep* state or in *Sleep after Reset* state. If a debug request is sent to the core, it enters its *Debug in Sleep* state. This debug mode works similarly to the "Classical" *Debug* state, except it returns to the original state (*Sleep* or *Sleep after Reset*) when the debug mode is left via the exec_core instruction of the OnCE. From this *Debug in Sleep* state, the SDMA can execute a program whereas no channel is running. If a new debug request is sent to the core or if a SoftBkpt is executed, it comes back to this *Debug in Sleep* state.

The OnCE is provided with several instructions that can be executed when the core is in either debug state. The following table summarizes the behavior of these OnCE debug instructions. There exists other secondary OnCE instructions that are described in OnCE and Real-Time Debug.

Table 43-7. SDMA in Debug Mode

Instruction	Debug	Debug in Sleep
exec_once	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug in Sleep* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.
run_core	run_core <instruction> SDMA executes the <instruction>, leaves the *Debug* state and continues executing the channel script from the position where it stopped. This command must not be used with an instruction that modifies the PC value.	run_core <instruction> SDMA executes the <instruction> and returns to its *Sleep* or *Sleep after Reset* initial state. This command must not be used with an instruction that modifies the PC value.
exec_core	exec_core <instruction> It is similar to run_core except it requires an instruction that changes the PC value (jump, branch...): the SDMA jumps to the new PC value, leaves the *Debug* state and starts executing instructions from this new PC value.	exec_core <instruction> If the previous state was *Sleep after Reset*, the SDMA returns to this state, and Chn0Addr value overrides the PC value. Otherwise, the SDMA jumps to the new PC value and starts executing instructions from this new PC.

NOTE

The feature exec_core in *Debug in Sleep* after *Sleep after Reset* was added for the Channel boot (channel 0) to allow the debugger to return to *Sleep after Reset* state with a new PC

value. The SDMA will be ready to boot at the Chn0Addr address.

43.4.6 SDMA Clocks and Low Power Modes

The SDMA receives several root clocks from the SoC clock controller block and performs adaptive clock gating to optimize its power consumption. From a user standpoint, clock gating and power mode selection are fully automatized inside the SDMA.

Root clock control is available from the SoC clock controller block.

There are numerous clock sources that are used in the SDMA. They belong to one of two possible clock domains listed in the following table, and have frequency constraints within each domain. Clocks are considered asynchronous between domains.

Within the ARM platform/SDMA clock domain, all clocks must come from the same DPLL. The ARM platform DMA interfaces (peripheral DMA and burst DMA) receive their clock from the ARM platform DMA clock source whose frequency can be once or twice the frequency of the SDMA core clock. The DMA interfaces are designed to work at the ARM platform DMA frequency, but the SDMA core is physically limited to a maximum 104 MHz frequency. Since this is lower than the maximum ARM platform DMA frequency, the SDMA core clock is tied to the ARM platform peripheral clock frequency.

The ARM platform Peripheral Bus Clock source must be an exact sub-frequency of the SDMA Core clock source (any integer value greater or equal to 1).

Table 43-8. Clocking Scheme

Clock Domain	Source Clock	Comments
ARM platform	SDMA core (SDMA main core)	Source clock for the core and all its operations; this clock is thus used by most of the SDMA sub-blocks.
	ARM platform DMA	DMA interface for the peripheral DMA and the burst DMA. It is balanced with the main clock source, and its frequency is either once or twice the main clock frequency.
	ARM platform peripheral	Connection to the ARM platform peripheral bus. It is a sub-frequency of the main clock frequency.
JTAG	TCK	Clock for JTAG access, limited to maximum of 1/8 of the SDMA core clock frequency.

The JTAG clock is sampled by the SDMA main clock to determine its rising edge. This simplifies design and clock management, but it also adds a ratio constraint between those two clocks. It is guaranteed the JTAG interface works properly when the frequency of TCK is lower than $1/8^{th}$ of the frequency of the SDMA main clock (which is about 8 MHz when the SDMA core clock frequency is 66 MHz).

43.4.6.1 Clock Gating and Low Power Modes

The SDMA automatically performs power saving without requiring user involvement. It implements two levels of automatic clock gating.

43.4.6.1.1 Coarse Clock Gating

Every sub-block clock comes from one of the five available sources, and is gated with the sub-block specific enabling condition.

The following table displays the sub-block clocks and their source. It also indicates the relationships that may exist between different sub-blocks clock enables.

Table 43-9. Sub-blocks Clocks

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Core	SDMA Main Core	The core sub-block clock is running when the core is not in one of its sleep states (Sleep or Sleep after Reset) or there is a pending channel. Typically, the core sub-block clock is stopped once all the channels are processed and the core enters its sleep state. A new pending channel awakes the core sub-block clock.	None
Memories	SDMA Main Core	The clock activation only occurs during a core access.	Disabled when Core sub-block clock is disabled or no memory access in progress
Scheduler	SDMA Main Core	Its clock only runs when scheduling is needed: for example, when there are pending channels, upon reception of a DMA request, and anytime the ARM platform modifies the channel running conditions.	None
ARM platform Control	SDMA Main Core & ARM platform peripheral	The ARM platform peripheral clock is solely used to determine the frequency ratio with the SDMA main clock. The control registers' clock is based on *SDMA main clock*; it is active when the ARM platform or the SDMA modifies the contents of one of these registers.	None
Burst DMA	SDMA Main Core & ARM platform DMA	The burst DMA has two clocks: The first clock is derived from the SDMA main core clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled	Disabled when Core sub-block clock is disabled

Table continues on the next page...

Table 43-9. Sub-blocks Clocks (continued)

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
		during active phases of data transfers (for example, these clocks are turned off when the burst DMA is not used by the running channel script).	
Peripheral DMA	SDMA Main Core & ARM platform DMA	The peripheral DMA has two clocks: The first clock is derived from SDMA main clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the peripheral DMA is not used by the running channel script).	Disabled when Core sub-block clock is disabled
OnCE	SDMA Main Core	The OnCE clock is derived from main source clock. It is disabled by default. In order to use the OnCE, its clock must be explicitly turned on, either by enabling the OnCE access from the ARM platform peripheral bus (register ONCE_ENB), or by driving the clk_gating_off input pin high. This is a SDMA input whose driver depends on the SoC implementation (typically a JTAG controller). The OnCE also receives the TCK input, which is the JTAG clock. It does not use it as a functional clock; the TCK input is sampled instead. Refer to Synchronization Implementation.	When enabled, all other clocks are systematically on (clock gating is off)

43.4.6.1.2 Refined Clock Gating

The SDMA implements a second level of clock gating on a register-per-register basis.

Unlike the first level that covers all the SDMA flip-flops, except the synchronizers (only five flip-flops are always running), the second level is only available for eligible registers, which amounts to about 90% of the SDMA flip-flops.

These gated registers are only clocked when the hardware logic detects a new data loading. This additional gating further reduces dynamic power consumption.

43.4.6.1.3 Low Power Modes and User Control

Power savings are automatically managed by the SDMA hardware without any user involvement; however, one can distinguish three different power modes: SLEEP, RUN, and DEBUG.

The following table describes these modes, and shows how to switch from one mode to another.

Table 43-10. Power Modes

Power Mode	Sub-blocks							Comments
	Core	Memories	Scheduler	ARM platform Control	Burst DMA	Peripheral DMA	OnCE	
SLEEP	off[1]	off	wait[2]	wait	off	off	off	Set when the PCU state is either *Sleep* or *Sleep after Reset* and the SDMA is not in DEBUG mode. This is the default mode after reset.
RUN	on[3]	wait	wait	wait	wait	wait	off	Set for the other PCU states that are reachable out of debug: *Program*, *Data*, *Change of Flow*, *Error in Loop*, *Debug*, *Functional Unit*, *Save,* or *Restore*.
DEBUG	on	on	on	on	on	on	on	Set regardless of the PCU state when clock gating is turned off to use the OnCE features (either *clk_gating_off* pin high or ONCE_ENB[0] set).

1. *off*: no clock
2. *wait*: only clocked when accessed or stimulated
3. *on*: clock is always running

It is possible to control the SDMA power mode. The procedures to force the SDMA into either mode are described in SLEEP Mode.

43.4.6.1.3.1 SLEEP Mode

This is the default mode after reset; therefore, resetting the SDMA forces this mode.

However, the common procedure is as follows:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Disable all channels (via the STOP_STAT control register, and the HO, DO, EO if necessary).
- Wait for the active channels to complete or force a reschedule via the reschedule bit in the RESET register.
- The SDMA is in SLEEP mode making it possible to completely shut off its clock from the chip level clock controller using the procedure described in Stop Mode Response.

43.4.6.1.3.2 RUN Mode

This is the default mode when a channel is running:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Activate at least one channel (via the HSTART control registers, a DMA request, and/or the HO, DO, EO register bits).

43.4.6.1.3.3 DEBUG Mode

The DEBUG mode must be set when one needs to use the debugging facilities of the SDMA.

- Ensure the SDMA clocks are running from the CCM.
- Set the *clk_gating_off* pin high or use the SDMA to set ONCE_ENB[0].

43.4.6.1.4 Stop Mode Response

The SDMA receives a stop request from the chip level clock controller. This request may be asserted when the chip enters the stop low power mode.

If the SDMA is running when the request is received, then the SDMA will complete all pending channels before returning to the SLEEP state. The SDMA sends an acknowledgement to the clock controller when the SLEEP state is entered indicating that the SDMAs clocks can be turned off.

43.4.6.2 Reset

After reset (either received from the reset block or a software reset required by the ARM platform), the SDMA is in IDLE mode. It will start its boot code located at address 0 once a channel is activated.

Activating a channel can be done by the ARM platform after programming a positive priority and setting the channel bit in the EVTPEND register.

There will not be a context RESTORE for the first channel (bootload channel) called after a reset because the context data in RAM has not been initialized. Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized. Subsequent calls to the same channel or different channels may use any of the dynamic context modes

43.4.7 Software Interface

Appendix A fully describes the SDMA Application Programming Interface (API).

43.4.8 Initialization Information

This section discusses the following:

- Hardware Reset
- Channel Script Execution
- Initialization and Script Execution Setup Sequence

43.4.8.1 Hardware Reset

After reset, the program RAM, context RAM, data RAM, and RAM containing the channel enable registers (CHNENBLn) have unpredictable contents.

The active register set is assigned to channel 0 and the PC is initialized to all zeros. However, since the channel enable register is all zeros, there are no active channels and the SDMA is halted waiting for the boot channel to start.

The ARM platform will have to setup the SDMA in order to boot it. The CONFIG register must be initialized to determine the DMA/core clock ratio (1 or 2). Channel Enable Registers must also be initialized.

To start up the SDMA, the ARM platform first creates some channel control blocks (CCB) and buffer descriptors (BD) in ARM platform memory for the boot channel (channel 0) and then initializes the channel 0 pointer register (SDMA_MC0PTR) to the address of the first control block. Data Structures for Boot Code and Channel Scripts provides an overview of the data structure for the CCB and BD's. The SDMA_HSTART, SDMA_HOSTOVR and SDMA_EVTOVR registers are then configured according to Runnable Channels Evaluation to allow channel 0 to run.

Upon being enabled, the SDMA begins executing the script located at the address indicated by the Channel 0 Boot Address register (SDMA_CHN0ADDR) in the program memory. The reset value of SDMA_CHN0ADDR points to the default bootload script in ROM. This ROM script will read the channel 0 pointer register (SDMA_MC0PTR) to determine the location of the Channel Control Block (SDMA_CCB) in ARM platform memory. The script will then begin fetching by DMA the first channel control block which contains a pointer to the location channel 0 Buffer Descriptor chain which is also fetched via DMA. If the buffer descriptor contains a valid command, the script interprets the command in each buffer descriptor and proceeds to implement the command and move on to the next buffer descriptor control block. The buffer descriptor commands for

channel zero are typically set up to load SDMA's program RAM, Data RAM, and initial values for the channel contexts. Some channel scripts expect particular parameters to be passed

There are two ways to make the SDMA boot on a user-defined script. The OnCE (either via its JTAG interface or its ARM platform Control interface) can be used to download any code in the SDMA RAM and force the SDMA to boot on that code. Also, the SDMA_CHN0ADDR register in the ARM platform programming model can be modified to point to user code in RAM which would need to either have been loaded via the ONCE or default bootload routine (ex before a S/W reset).

43.4.8.2 Channel Script Execution

The execution of an SDMA script depends on both the instructions that make up the script, the data context upon which it operates, and commands or parameters allowed to the buffer. All these items must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate context, but may share scripts and locations in data RAM.

Channels are initialized by the ARM platform by using channel 0 to download any required scripts and data values and the channels initial context. The context contains all the initial values of the SDMA core registers. This includes the Program Counter (PC) which is set to the start of the desired script in SDMA program memory.

The ARM platform selects which trigger conditions that must occur for the channel to start by configuring the SDMA_CHNENBL, SDMA_HOSTOVR and SDMA_EVTOVR registers. The trigger events include ARM platform setting HE (SDMA_HSTART) or a hardware DMA request asserts an event input to SDMA. The channel can become active according to its priority compared with other runnable channels when the selected trigger(s) cause the condition described in Runnable Channels Evaluation to evaluate as true.

The specific parameters to be passed to each script in the buffer descriptor or context are documented in the software documentation for each script. Please refer to SDMA Scripts for complete script documentation. Buffer Descriptor Format provides an overview of the buffer descriptor format.

43.4.8.3 Initialization and Script Execution Setup Sequence

To summarize, the following steps are minimally required to setup SDMA and run channel scripts.

- Perform Hardware Reset. The program RAM, context RAM, data RAM and SDMA_CHNENBLn registers have unpredictable contents after this reset.
- Initialize SDMA_CHNENBLn registers to map DMA request events to desired channels.
- Configure SDMA_CHNPRIn registers to select priority for runnable channels. A non-zero priority is required for the channel to run.
- Configure the SDMA_CONFIG register to select DMA to SDMA core clock ratio .
- Set up channel control blocks and buffer descriptors in ARM platform to specify the loading of SDMA program RAM and channel contexts for each SDMA channel to be used. Reference Data Structures for Boot Code and Channel Scripts.
- Configure SDMA_MC0PTR register with base address of ARM platform Channel Control Block base address.
- Initialize SDMA_CHNENBLn registers to map DMA request events to associated channel. Reference Mapping DMA Requests to Pending Channels.
- Configure SDMA_CHNPRIn registers to set priority for each channel to be run.
- For each channel to be run, configure SDMA_HOSTOVR (HO) and SDMA_EVTOVR (EO) registers to select which events (hardware and/or software trigger events) must occur for the channel to be runnable. Reference Runnable Channels Evaluation.
- Set bit 0 of the SDMA_HSTART register to set HE[0] and allow Channel 0 to run (assumes EO[0] andDO[0] were both set in previous step). This will cause SDMA to load the program RAM and channel contexts configured previously.
- Wait for Channel 0 to finish running. This is indicated by HI[0]=1 in the SDMA_SDMA_INTR register, or by optional interrupt to the ARM platform.
- Set the LOCK bit in the SDMA_SDMA_LOCK register to prevent un-authorized uploads of data to SDMA RAM.
- Additional channel scripts can now be run by enabling the selected software or hardware trigger event according to Runnable Channels Evaluation.

43.4.9 SDMA Programming Model

This section describes the programming model for the SDMA RISC engine, including its processor, memory, and internal control registers.

All addresses are related to the internal SDMA memory map, which is completely different from the ARM platform memory maps. The ARM platform processor has no access to any hardware resource described, except when those resources are described in ARM Platform Memory Map and Control Register Summary. .

43.4.9.1 State and Registers Per Channel

The SDMA can be seen as a set of 32 identical devices that are able to perform one data transfer channel each. Only one channel can work at a time, but every channel state is available at any time.

This chapter lists the components of every channel state.

43.4.9.2 General Purpose Registers

Each channel has eight general purpose registers of 32 bits for use by scripts. General register 0 has a dedicated function for the loop instruction, but otherwise can be used for any purpose.

43.4.9.3 Functional Unit State

Each channel context has some state that is part of the functional units.

The specific allocation of this state is part of the functional unit definition that is described in Burst DMA Unit Programming, Peripheral DMA Unit Programming .

This state must be saved/restored on context switches.

43.4.9.3.1 Program Counter Register (PC)

The PC is 14 bits. Since instructions are 16 bits in width and all memory in the SDMA is 32 bits in width, the low order bit of the PC selects which half of the 32-bit word contains the current instruction.

A low order bit of zero selects the most significant half of the word.[1]

43.4.9.3.2 Flags

Each channel has the following four flags:

- The T bit reflects the status of some arithmetic and test instructions. It is set when the result of an addition or a subtraction is zero and cleared otherwise. It is also the copy of the tested bits. Finally, it can also be set when the loop counter (GReg0) reaches zero. When the last instruction of the hardware loop is an operation that can modify the T flag, its effect on T is discarded and replaced by the GReg0 status.

1. For example, big-Endian.

- Two additional bits, SF and DF, are used to indicate error conditions resulting from loading data sources and storing to destinations, respectively. Access errors set these bits, and successful transactions clear them. They can also be cleared by specific instructions (CLRF and loop). The source fault (SF) is updated by the loads LD and LDF; the destination fault (DF) is updated by the stores ST and STF.
- Access errors are caused by several conditions including writing to the ROM, writing to a read-only memory mapped register, accessing an unmapped address, or any transfer error received by a peripheral when it is accessed.

  The SF and DF flags have a major impact on the behavior of the hardware loop: If SF or DF is set when starting a hardware loop and it is not masked by the loop instruction, the loop body will not be executed. Inside the loop body, if a load or store sets the corresponding SF or DF flag, the loop exits immediately. Testing the status of the T flag at the end of the loop (as well as testing both SF and DF) tells if the loop exited abnormally as any anticipated exit prevents GReg0 from reaching the zero value and thus setting the T flag. This is also valid if the fault occurs at the last instruction of the last loop.

- The last flag is the loop mode flag, LM, which is composed of two bits. The most significant bit indicates when the processor is currently operating in loop mode. It is set by the loop instruction and is cleared after execution of the last instruction of the last loop. The least significant bit is set when the program counter points to the last instruction of a loop on the last path. It is used for a channel that is restored with this configuration to know that the next program counter is EPC. As with the dynamic context switch Greg0, which indicates when the program must get out of the loop, it can be restored only on the last instruction of the loop. This, however, is too late to fetch the next instruction after the loop.

43.4.9.3.3 Return Program Counter (RPC)

The RPC is 14 bits. It is set by the jump to the subroutine instructions and used by the return from the subroutine instructions.

Instructions are available to transfer its contents to and from a general register.

43.4.9.3.4 Loop Mode Start Program Counter (SPC)

The SPC is 14 bits. It is set by the loop instruction to the location immediately following it.

43.4.9.3.5 Loop Mode End Program Counter (EPC)

The EPC is 14 bits. It is set by the loop instruction to the location of the next instruction after the loop.

43.4.9.4 Context Switching-Programming

Each channel has a separate context consisting of the eight general purpose registers and additional registers representing the state of the functional units.

The active registers and functional units contain the context of the active channel. The context of inactive channels is stored in SDMA RAM, which is part of the SDMA address space.

In a function of the selected context switching mode (Context Switching), modified registers by the program can be saved in the channel RAM space while the program is going on. In every cycle, a write access to the RAM is possible.

On a done or yield(ge) instruction, SDMA goes into "real" context switching. In one of the dynamic modes, modified registers not previously saved, as well as the PC-Loop registers, are stored into the context area of the channel that will be closed. The new PC-Loop registers are loaded from the context area of the new channel. All other registers are restored while the program is executed, giving priority to registers used by the decoded instruction. Therefore, in the best case, only the PC and Loop registers should be saved and restored during this context-switching phase, which only requires five SDMA cycles.

In static mode, the context switch stores all registers in the old channel RAM space, and restores all registers from the new channel RAM space. It requires 26 SDMA cycles.

The address of the context memory for channel *i* is *CONTEXT_BASE* + 24**i* or *CONTEXT_BASE* + *32*i w*here CONTEXT_BASE equals 0x0800. The table below presents the layout of a channel context in memory:

Table 43-11. Layout of a Channel Context in Memory for SDMA

OFFSET	31	30	29-16	15	14	13-0
0	SF	-	RPC	T	-	PC
1	LM		EPC	DF	-	SPC
2	GR0					
3	GR1					
4	GR2					
5	GR3					
6	GR4					
7	GR5					

Table continues on the next page...

Table 43-11. Layout of a Channel Context in Memory for SDMA (continued)

8	GR6
9	GR7
10	MDA (burst DMA)
11	MSA (burst DMA)
12	MS (burst DMA)
13	MD (burst DMA)
14	PDA (peripheral DMA)
15	PSA (peripheral DMA)
16	PS (peripheral DMA)
17	PD (peripheral DMA)
18	
19	
20	Reserved[1]
21	Reserved[1]
22	Reserved[1]
23	Reserved[1]
24	Scratch RAM (optional)
25	Scratch RAM (optional)
26	Scratch RAM (optional)
27	Scratch RAM (optional)
28	Scratch RAM (optional)
29	Scratch RAM (optional)
30	Scratch RAM (optional)
31	Scratch RAM (optional)

43.4.9.5 Address Space

The SDMA has four internal buses which are listed here.

- The Instruction bus reads instructions from the memory. Its address map is described in Instruction Memory Map.
- The Data bus (DMBUS) accesses the same memories as those visible on the Instruction bus, some memory-mapped registers (scheduler status and OnCE registers), and up to 14 peripherals. Its address map is described in Data Memory Map.

- The Functional Units bus (FUBUS) accesses the , Burst DMA, Peripheral DMA . The addressing mechanism is further detailed in Functional Units Programming Model.
- The Context Switch bus reads/writes registers into context-switch RAM space. It is a 64-bit bus dedicated for accessing this RAM space for updating the context of the running channel. While the program is going on, this bus has the lowest priority compared to the Instruction and Data buses, except for restoring a register needed for the decoded instruction to be executed. On the save part of a context switch (when the PCU is in its slave state), this is the only one used. On the restore part, the Instruction bus has the priority to read the next instruction at the restored PC and otherwise the Context Switch bus is used. It is not possible to control the actual data transfers that occur on this bus.

43.4.9.5.1 Instruction Memory Map

The instruction memory map is based on a 14-bit address bus and a 16-bit data (instruction) bus. Each address corresponds to a 16-bit data location.

Instructions are fetched from either program ROM or program RAM. An SDMA script is able to change the contents of the program RAM, which is also visible from the data bus.

The first two instruction locations (at 0 and 1) are special. Location 0 is where the PC is set on reset. Location 1 is where the PC is set upon the execution of an illegal instruction. It is expected that both of these locations will contain a jmp to handle routines.

Table 43-12. SDMA Instruction Memory Space

Device	SDMA Address (Hex)	Base Address Label	Block Name	WS	Description
ROM	0x0000 ↓ 0x07FF	SDMA_IBUS_ROM_ADDR	-	0	4 Kbyte internal ROM with boot code and standard routines.
RAM	0x1000 ↓ 0x1FFF	SDMA_IBUS_RAM_ADDR	-	0	8 Kbyte internal RAM with channels context and user data/routines.

43.4.9.5.2 Data Memory Map

All of the data accessible to SDMA scripts make up the data memory space of the SDMA.

This address space has several components:

- ROM (also visible on the Instruction bus)
- RAM (also visible on the Instruction bus)

- Shared Peripherals Registers
- SDMA Internal Registers (scheduler, OnCE, and registers that are also accessible by the ARM platform)

SDMA scripts can read and write to the context RAM, data RAM, shared peripheral registers, and internal registers.

The address range is 16 bits and the data width is 32 bits. When accessing peripheral registers (USB and so on), the data width may be different. The exact address map for the peripherals depends on the project (as presented in each respective chapter).

Data access is performed with *ld* and *st* instructions that take the address from a general purpose register in the core (GReg*n*). The mapping between the general purpose register contents and the address bus is given in the following table:

Table 43-13. GReg*n* to DMBUS Address Mapping

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
sz	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
address															

Grayed bits are simply discarded but they must be cleared to ensure forward-script compatibility.

- sz (bit 31) indicates the peripheral data width: 0 is used for a 32-bit peripheral and 1 is used for a 16-bit peripheral.
- address (bits 15 down to 0) is the address of the accessed resource (internal memory, internal register, or shared peripheral).

Table 43-14. SDMA Data Memory Space

Device	SDMA Address (Hex)	Size	Description
ROM	0x0000 → 0x03FF	4 Kbyte	4 Kbyte internal ROM with boot code and standard routines
Reserved	0x0400 → 0x07FF	4 Kbyte	4 Kbyte Reserved
RAM	0x0800 → 0x0FFF	8 Kbyte	8 Kbyte internal RAM with channels contexts and user data/routines
per1	0x1000 → 0x1FFF	16 Kbyte	*peripheral 1* memory space (4 Kbyte peripheral's address space)
per2	0x2000 → 0x2FFF	16 Kbyte	*peripheral 2* memory space (4 Kbyte peripheral's address space)
per3	0x3000 → 0x3FFF	16 Kbyte	*peripheral 3* memory space (4 Kbyte peripheral's address space)
per4	0x4000 → 0x4FFF	16 Kbyte	*peripheral 4* memory space (4 Kbyte peripheral's address space)
per5	0x5000 → 0x5FFF	16 Kbyte	*peripheral 5* memory space (4 Kbyte peripheral's address space)
per6	0x6000 → 0x6FFF	16 Kbyte	*peripheral 6* memory space (4 Kbyte peripheral's address space)
Registers	0x7000 → 0x7FFF	16 Kbyte	Memory mapped registers
per7	0x8000 → 0x8FFF	16 Kbyte	*peripheral 7* memory space (4 Kbyte peripheral's address space)

Table continues on the next page...

Table 43-14. SDMA Data Memory Space (continued)

Device	SDMA Address (Hex)	Size	Description
per8	0x9000 → 0x9FFF	16 Kbyte	*peripheral 8* memory space (4 Kbyte peripheral's address space)
per9	0xA000 → 0xAFFF	16 Kbyte	*peripheral 9* memory space (4 Kbyte peripheral's address space)
per10	0xB000 → 0xBFFF	16 Kbyte	*peripheral 10* memory space (4 Kbyte peripheral's address space)
per11	0xC000 → 0xCFFF	16 Kbyte	*peripheral 11* memory space (4 Kbyte peripheral's address space)
per12	0xD000 → 0xDFFF	16 Kbyte	*peripheral 12* memory space (4 Kbyte peripheral's address space)
per13	0xE000 → 0xEFFF	16 Kbyte	*peripheral 13* memory space (4 Kbyte peripheral's address space)
per14	0xF000 → 0xFFFF	16 Kbyte	*peripheral 14* memory space (4 Kbyte peripheral's address space)

43.4.10 SDMA Initialization

Appendix A describes the setup of the SDMA . This section provides a quick description of several initialization procedures.

NOTE

There may be differences with the actual implementation in the API.

43.4.10.1 Hardware Reset-SDMA

After reset, the RAM that holds contexts, data, scripts, and the DMA request-channels matrix has unpredictable content.

The core registers are all reset to 0, including the PC; the PCU state is *Sleep after Reset*. No channel can be activated because all of the priorities are also reset to 0.

43.4.10.2 Standard Boot Sequence

The following is the standard boot sequence:

1. Initialize the CONFIG register-detailed in Configuration Register (SDMAARM_CONFIG)-to determine the ARM platform DMA/core clock ratio (1 or 2)
2. Initialize the DMA request-channels matrix (seeChannel Enable RAM (SDMAARM_CHNENBL*n*)).
3. Program the channel control registers-Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel

BP Override (SDMA_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND)-according to the channel allocation.
4. Perform any necessary setup as required by the standard boot script in ROM (this is described in Appendix A).
5. Trigger channel 0 with the Channel Start (SDMAARM_HSTART) register, which starts the execution of the ROM script starting at address 0. This boot downloads channel scripts and contexts in RAM.

43.4.10.3 User-Defined Boot Sequence

The following is a user-defined boot sequence:

1. Initialize the Configuration Register (SDMAARM_CONFIG)Channel Enable RAM (SDMAARM_CHNENBL*n*), Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel ARM platform Override (SDMAARM_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND).
2. Use the OnCE (either via its JTAG interface or its ARM platform control registers) to download any code in the SDMA RAM. Accessing the Memory describes how to write data to the RAM via the OnCE.
3. Use the OnCE instructions to make the PC default value point to the new boot script start address, or rely on the ROM startup script, which first jumps to the address in Channel 0 Boot Address (SDMAARM_CHN0ADDR). (This register default address points to the standard boot script.)

43.4.10.4 Script Loading and Context Initialization

The execution of an SDMA script depends on both the instructions that make up the script and the data context upon which it operates. Both must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate data context, but may share scripts and locations in the data RAM.

The ARM platform manages the space in program RAM and data RAM. It also manages the assignment of SDMA channels to the device drivers that need them. Channels are initialized by the ARM platform via the channel 0 boot script. The boot channel downloads any required scripts with their data and the channels' initial contexts. Every context contains all the initial values of the registers, including the PC. Then the ARM platform can enable any channel that becomes active and begins fetching and executing instructions from its script.

43.4.11 Instruction Description

The following sections introduce the instruction of the SDMA.

Instruction set details are available in Instruction Set.

43.4.11.1 Scheduling Instructions

The following are scheduling instructions:

- done-The instruction causes certain scheduling or interrupt bits to be set or cleared, which may cause a change in the schedule-ability of the running channel. Then the instruction causes the SDMA to evaluate the current scheduling priorities and to choose the highest priority ready channel. If this channel is not the current channel, a context switch will take place. If there are no runnable channels, the SDMA will enter the stopped mode. The done 5 has a special usage reserved for debug, as explained in Debug Instructions.
- yield-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater than the current channel priority.
- yieldge-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater or equal to the current channel priority.
- notify-The notify instruction affects the scheduling bits, but does not cause rescheduling.

43.4.11.2 Conditional Branch Instructions

The conditional branch instructions of an 8-bit displacement, which is sign-extended and added to the current PC (which points to the next instruction) if the condition is satisfied.

Otherwise, control passes to the next sequential instruction.

- BF-Branch if False. The branch is taken if the T bit in the processor status is zero (false).
- BT-Branch if True. The branch is taken if the T bit in the processor status is one (true).

- BSF-Branch if Source Fault. The branch is taken if the SF bit in the processor status is one.
- BDF-Branch if Destination Fault. The branch is taken if the DF bit in the processor status is one.

43.4.11.3 Unconditional Jump Instructions

There are two varieties of unconditional control transfers: an absolute transfer and a through-register transfer.

Absolute transfers have a 14-bit address field that replaces the current PC.

- JMP-Jump. Causes the processor to jump to an absolute address encoded in the instruction itself.
- JSR-Jump to Subroutine. Causes the processor to jump to a subroutine, the address of which is encoded in the instruction itself.
- JMPR-Jump through Register. Causes the processor to jump to an absolute address contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.
- JSRR-Jump to Subroutine through Register. Causes the processor to jump to a subroutine, the address of which is contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.

43.4.11.4 Subroutine Return Instructions

The following are subroutine return instructions:

- RET-Return from Subroutine. The RET restores the contents of RPC to PC.
- LDRPC-Load from RPC to Register. THe LDRPC instruction is meant to be used when more than one level of subroutines are required. It stores the contents of RPC in any General register.

43.4.11.5 Loop Instruction

The following is a loop instruction:

LOOP-Enters Loop Mode. Before entering loop mode, the loop instruction can optionally clear the fault flags (SF and/or DF) based on a 2-bit field in the instruction. This feature is linked to the fact that setting SF or DF in loop mode will cause an immediate exit of the loop.

43.4.11.6 Miscellaneous Instructions

The following are miscellaneous instructions:

- CLRF-Clear Fault Flags. This instruction clears any combination of SF and DF.
- MOV r,s-This moves data from GReg[s] to GReg[r].
- LDI r,immediate-This loads GReg[r] with a zero-extended immediate value.

43.4.11.7 Logic Instructions

The following are logic instructions:

- XORr,s-This performs an exclusive or between GReg[r] and GReg[s], and stores the result in GReg[r].
- XORIr,immediate-This performs an exclusive or between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- ORr,s-This performs an or between GReg[r] and GReg[s], and stores the result in GReg[r].
- ORIr,immediate-This performs an or between GReg[r] and a zero-extended immediate value and, stores the result in GReg[r].
- ANDNr,s-This performs an and between GReg[r] and the negated GReg[s], and stores the result in GReg[r].
- ANDNIr,immediate-This performs an and between GReg[r] and the negated zero-extended immediate value, and stores the result in GReg[r].
- ANDr,s-This performs an and between GReg[r] and GReg[s], and stores the result in GReg[r].
- ANDIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].

43.4.11.8 Arithmetic Instructions

Arithmetic instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the result is zero, otherwise it is cleared.

- ADD r,s-This performs the addition of GReg[r] and GReg[s], and stores the result in GReg[r].
- ADDI r,immediate-This performs the addition of GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].

- SUB r,s-This performs the subtraction of GReg[s] from GReg[r], and stores the result in GReg[r].
- SUBIr,immediate-This performs the subtraction of a zero-extended immediate value from GReg[r], and stores the result in GReg[r].

43.4.11.9 Compare Instructions

Compare instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the comparison is true, otherwise it is cleared.

NOTE

Only one version of the immediate form is implemented. Non-equality comparisons to immediate values will require two instructions.

- CMPEQ r,s-This sets T when registers GReg[r] and GReg[s] are equal.
- CMPEQIr,immediate-This sets T when register GReg[r] and the zero-extended immediate value are equal.
- CMPLTr,s-This sets T when register GReg[r] is less than and not equal to GReg[s]. The comparison is signed.
- CMPHS r,s-This sets T when register GReg[r] is greater than or equal to GReg[s]. The comparison is signed.

43.4.11.10 Test Instructions

Test instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if any bit in the result is one, otherwise it is cleared.

- TSTr,s-This performs an and between GReg[r] and GReg[s], and sets T if the result is not zero.
- TSTIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and sets T if the result is not zero.

43.4.11.11 Byte Permutation Instructions

These instructions shuffle the bytes in a register. For the purpose of describing these instructions, have the bytes in a register be numbered from the most significant as b_3, b_2, b_1, b_0.

- RORBr-The rotate right byte. The result is b_0, b_3, b_2, b_1.

- REVBr-The reverse bytes in word. The result is b_0, b_1, b_2, b_3.
- REVBLOr-The reverse, two low-order bytes. The result is b_3, b_2, b_0, b_1.

43.4.11.12 Bit Shift Instructions

The following are bit shift instructions:

- ROR1r-The rotate right 1 bit. This instruction does a circular right shift of 1 bit.
- LSR1r-The logical shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by a 0.
- ASR1r-The arithmetic shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by itself.
- LSL1r-The logical shift left 1 bit. This instruction shifts all bits to the left by 1. The low order bit is replaced by zero.

43.4.11.13 Bit Manipulation Instructions

- BCLRIr,n-The bit clear is immediate; clears bit number *i* in register *r*.
- BSETIr,n-The bit set is immediate; sets bit number *i* in register *r*.
- BTSTIr,n-The bit test is immediate; tests bit number *i* in register *r* (T becomes equal to the selected register bit).

43.4.11.14 SDMA Memory Access Instructions

All memory accesses are 32 bits.

Any memory location that is implemented with less than 32 bits (for example, peripheral registers) causes unimplemented bits to be read as 0s.

All memory accesses will cause either the SF or DF flags in the processor status to be set if they cause a fault.

What constitutes a fault, especially when accessing peripheral registers, is a property of the memory location.

- LDr,(b,d)-The load instruction creates an address by adding the displacement field (d) to the contents of the base register (b). The SDMA location at the resulting address is read and placed in the destination register (r).
- STr,(b,d)-The store instruction creates an address in the same manner as the load instruction. The register (r) is stored in the SDMA location at the resulting address.

43.4.11.15 Functional Unit Instructions

The functional unit instructions have an 8-bit field that is placed on the functional unit bus.

Some of these bits are used to select which functional unit should be involved in the transfer. The remaining bits are decoded by the selected functional unit so their specific use depends on the functional unit. See Functional Units Programming Model.

There are two functional unit instructions, as follows:

- LDFr,fub-The 8-bit field is placed on the functional unit bus and a read is issued to the selected functional unit. As a result of this instruction, the SF may be set in the processor status.
- STFr,fub-The 8-bit field is placed on the functional unit bus and a write is issued to the selected functional unit. As a result of this instruction, the DF may be set in the processor status.

43.4.11.16 Illegal Instructions

All instruction encodings that are illegal cause the following actions:

- The current PC (which points to one beyond the offending instruction) is put in the EPC register.
- The loop mode bit is cleared.
- The PC is set to the value stored in the Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR) register (the default value is 0x0001).

ILLEGAL-Although any instruction other than those indicated in the SDMA specification will trigger the illegal instruction mechanism, the ILLEGAL instruction code is preferred as it will always be kept as *illegal* in the possible future versions of the SDMA core.

43.4.11.17 Debug Instructions

The following are debug instructions:

- SOFTBKPT-The software breakpoint instruction causes the core to stop and enter debug mode. The core can then be accessed and started by the OnCE debug block only.

- done 5-This instruction is used for debugging, as it copies the contents of the PCU registers and flags to the context memory. Information on this instruction is described in Saving the Context.
- CpShReg-This instruction copies the context memory into the PCU registers and flags. Modifying the corresponding memory location before executing this instruction enables you to have the channel continue from a new instruction address. This instruction is described in Restoring the Context.

43.4.12 Functional Units Programming Model

The functional unit instructions cause an 8-bit code, found in the low eight bits of the instruction, to be asserted on the functional unit control bus.

Some of these bits are used to select one of several functional units. Functional units which can be selected include SDMA registers such as MSA and MSD which are not mapped in the SDMA memory map, and are accessible only through the functional unit bus. These Functional Unit Registers are listed in the following table. In order to establish a programming convention, assume the selection bits are some number of the most significant bits of the 8-bit code. Furthermore, some number of the least significant bits is decoded by a given functional unit to establish the type of operation to perform.

Table 43-15. Functional Unit Registers

Functional Unit	Register	Register Name	Section/Page
Burst DMA Unit Programming	SDMSA	Memory Source Address Register	Memory Source Address Register (MSA)
	MDA	Memory Destination Address Register	Memory Destination Address Register (MDA)
	MD	Memory Data Buffer Register	Memory Data Buffer Register (MD) (Write) Burst DMA Write (stf) (Read) Burst DMA Read (ldf)
	MS	Memory State Register	State Register (MS)
Peripheral DMA Unit Programming	PSA	Peripheral Source Address Register	Peripheral Source Address Register (PSA)
	PDA	Peripheral Destination Address Register	Peripheral Destination Address Register (PDA)
	PD	Peripheral Data Buffer Register	Peripheral Data Register (PD) (Write) Peripheral DMA Write (stf)-Write Mode

Table continues on the next page...

Table 43-15. Functional Unit Registers (continued)

Functional Unit	Register	Register Name	Section/Page
			(Read) Peripheral DMA Read (ldf)-Read Mode
	PS	Peripheral State Register	Peripheral State Register (PS)

More information regarding the functional units can be found in Peripheral DMA Unit, and Burst DMA Unit.

43.4.12.1 Burst DMA Unit Programming

The DMA instructions control the DMA state machine and may cause a DMA cycle on the associated memory bus.

There are four registers associated with the burst DMA unit: a Memory Source Address register (MSA), a Memory Destination Address register (MDA), a Memory Data buffer (MD), and a state register (MS). The burst DMA has two different uses:

- A data transfer between External Memory Interface and SDMA general register
- A data transfer in copy mode where blocks of data are transferred from the source address to the destination address

43.4.12.1.1 Memory Source Address Register (MSA)

The source address register contains the pointer into EXTMC memory associated with the next read data transfer. It has byte granularity.

Reading the register with the ldf instruction has no side effects, and gives the address value in the EXTMC memory of the next data that is read by the SDMA during an ldf MD instruction.

Writing the source address register has two side effects: If the prefetch bit is set, a DMA read cycle (8-word read access) is issued with the new address. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MSA to guarantee all the data is effectively written to memory.

The MSA register has two modes of programming:

- Frozen-In frozen mode, the MSA register is not modified after DMA accesses.
- Incremented (default mode)-In incremental mode, MSA is incremented by the number of bytes transferred during read cycles.

43.4.12.1.2 Memory Destination Address Register (MDA)

The destination address register contains the pointer into EXTMC memory associated with the next write data transfer. It has byte granularity.

Reading the MDA register with the ldf instruction has no side effects. It gives the address value in the EXTMC memory where the next SDMA data (stf r,MD instruction) is stored when MD FIFO is flushed.

Writing the destination address register has one side effect. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MDA to guarantee all the data is effectively written to memory.

The MDA register has two modes of programming:

- Frozen-In frozen mode, the MDA register is not modified after DMA accesses.
- Incremented (default mode)-The MDA register is incremented by the number of bytes transferred during write cycles.

43.4.12.1.3 Memory Data Buffer Register (MD)

The data buffer register consists of a bank of 36 bytes that behave like FIFO.

This FIFO stores the eight words received when a read burst is triggered by the DMA (DMA is in read mode).

The MD register is in write mode after a writing in MDA or after an stf MD instruction.

In that case, a burst write access is automatically triggered when there are more than eight words in MD. For bandwidth optimization, any transfers between DMA and the EXTMC controller are based on burst accesses.

An ldf r,MD|SIZE instruction that reads the data buffer may cause a DMA cycle, as follows:

- If there are less bytes in the FIFO than the size parameter of the instruction. For instance, if only two bytes are available in MD and a 4-byte read is requested, a burst read access is executed to complete the two bytes.
- If the prefetch bit is set, and after reading there is enough space in the FIFO to store a full burst, a burst read access is triggered.

An stf r,MD|SIZE instruction that writes to the data buffer may cause a DMA cycle if the number of written bytes in MD is higher than 32 (eight words) or if the flush bit is set.

When DMA is used for data transfer between SDMA and EXTMC (reading or writing), no immediate error is possible because the block manages a data misalignment issue; therefore, it is allowed to read/write a word to/from a half-word address. However, the addresses (source or destination) must belong to the EXTMC memory mapping. The only potential error, in this mode, would be the error sent back by the EXTMC controller when an access to a super-user page is detected. The whole transfer on the DMA associated bus will be considered successful when there are no errors seen on the bus during the transfer. In copy mode, an immediate error could be returned to SDMA as described in Burst DMA Unit Error Management.

43.4.12.1.4 State Register (MS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. MS is also accessed to set-up the conditional yielding feature.

The initialization value of this register is 0 and it consists of the following:

Table 43-16. SDMA_MS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	spriv	stype	0	0	dpriv	dtype
W																
R	0	0	0	0	y	d	e		0	0	n					
W																

Table 43-17. SDMA_MS Field Descriptions

Field	Description
31-22	Reserved
21 spriv	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
20 stype	Source Mode. Indicates if MSA has to be incremented (or not) during accesses. 0 Frozen-MSA is not modified. 1 Incremented-MSA is incremented by the number of transferred bytes during read access.
19-18	Reserved
17 dpriv	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved
16	Destination Mode. Indicates if MDA has to be incremented (or not) during accesses.

Table continues on the next page...

Table 43-17. SDMA_MS Field Descriptions (continued)

Field	Description
dtype	0 Frozen-MDA is not modified. 1 Incremented-MDA is incremented by the number of transferred bytes during write access.
15-12	Reserved
11 y	Conditional Yielding selector. When selected, theyield/yieldge instructions will not switch channels if the Burst DMA is in Write Mode, and it has less than four bytes in its FIFO. This is aimed at reducing the number of inefficient FIFO flushes due to context switches. 0 Always yields 1 Yields conditionally (when there are less than four bytes in the FIFO in write mode)
10 d	Access Direction or DMA Mode. DMA is in write mode when data was written into MD by stf MD instructions, or if a previous DMA cycle on the external bus was a write access. Writing MDA or MSA changes the DMA mode to the respective value. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by SDMA with an ldf MD instruction. Reading MDA or MSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 error read burst
7-6	Reserved
5-0 n	Number of bytes in the MD FIFO.

43.4.12.1.5 Burst DMA Write (stf)

When received from a stf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 43-18. STF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s		p	freeze	r			spriv
MDA								dpriv
MD			f	cpy			sz	
MS								

Table 43-19. STF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MSA)	Prefetch Flag 0 No prefetch 1 Prefetch required from new MSA
5 f (MD)	Forced Flush Flag 0 Automatic flush 1 FIFO contents are flushed (including the new written data).
4 freeze (MSA/MDA)	Address Freeze Mode 0 Address is normally incremented. 1 Address is frozen.
4 cpy (MD)	Copy Mode selection 0 Write Mode 1 Copy Mode
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD/MS)	Transfer Size 00 size 0 (no data stored in the FIFO) 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
0 spriv (MSA)	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
0 dpriv (MDA)	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved

The possible write instructions are listed in the table below (unused bits should always be cleared).

Table 43-20. Burst DMA STF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	stf r,MSA	Writes content of the SDMA general register (r) to the source address register. MSA is in incremented mode.

Table continues on the next page...

Table 43-20. Burst DMA STF Instruction List (continued)

Binary	Assembly	Comments
00_0_1_00_00	stf r,MSA|FR	Writes content of the SDMA general register (r) to the source address register. MSA is in frozen mode.
00_1_0_00_00	stf r,MSA|PF	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access. MSA is in incremented mode.
00_1_1_00_00	stf r,MSA|PF|FR	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access.
00_0_0_01_00	stf r,MDA	Writes content of the SDMA general register (r) to the destination address register. MDA is in incremented mode.
00_0_1_01_00	stf r,MDA|FR	Writes content of the SDMA general register (r) to the destination address register. MDA is in frozen mode.
00_1_0_10_00	stf r,MD|SZ0|FL	No data transfers between the SDMA and MD, but all valid written data of the MD is flushed to the memory. An acknowledge or error is sent back to the SDMA core on transfer completion.
00_0_0_10_01	stf r,MD|SZ8	8-bit (byte) transfer to write buffer MD
00_1_0_10_01	stf r,MD|SZ8|FL	8-bit (byte) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_10	stf r,MD|SZ16	16-bit (half-word) transfer to write buffer MD
00_1_0_10_10	stf r,MD|SZ16|FL	16-bit (half-word) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_11	stf r,MD|SZ32	32-bit (word) transfer to write buffer MD
00_1_0_10_11	stf r,MD|SZ32|FL	32-bit (word) transfer to write buffer MD and flush after transfer. All valid written data of MD is flushed to memory.
00_0_1_10_00	stf r,MD|CPY	No data transfer between SDMA and MD but starts a copy transfer whose length is given by the 4 LSB of *r* register. (Maximum burst length is eight words.)
00_0_0_11_11	stf r,MS	32-bit (word) transfer to status register MS
00_0_0_11_00	stf r,MS|SZ0	Clears the error flag (if set). Other MS bits are unchanged; this instruction is also known as clref MS.

NOTE

When a flush bit is set, the SDMA flushes the FIFO including the newly written data. An acknowledge is sent to the core before the flush completes (except if size 0 is used). The goal of this flush bit is to force a flush, but it is recommended to use it only when needed (for example, when finishing a row of pixels during 2D data transfers). Indeed, if this bit is omitted and if there are more than 32 bytes in the FIFO, a burst write access is automatically triggered.

Since all the stf r,MD instructions (including the copy mode) acknowledge the SDMA core before the store is effective (except if size 0 is used), it is recommended to perform an ldf

from MS before terminating a channel in order to check the final error status. (The ldf from MS will stall the core until all the data was flushed out and the transfer status is known.)

After every stf MD instruction, the MDA is incremented by the number of bytes that are written in MD, except when it is programmed in frozen mode.

43.4.12.1.6 Burst DMA Read (ldf)

When received from an ldf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 43-21. LDF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s				r			
MDA								
MD			p				sz	
MS								

Table 43-22. LDF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD)	Transfer Size 00 reserved 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

The table below lists the possible write instructions (unused bits should always be cleared).

Table 43-23. Burst DMA LDF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	ldf r,MSA	Copies the source address register value into an SDMA general register. It gives the memory address of the next data that will be read with an ldf MD instruction.
00_0_0_01_00	ldf r,MDA	Copies the destination address register value into an SDMA general register. It gives the memory address where the next incoming data will be flushed.
00_0_0_10_01	ldf r,MD|SZ8	8-bit (byte) read
00_1_0_10_01	ldf r,MD|SZ8|PF	8-bit (byte) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_10	ldf r,MD|SZ16	16-bit (half-word) read
00_1_0_10_10	ldf r,MD|SZ16|PF	16-bit (half-word) read. If after this reading, and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_11	ldf r,MD|SZ32	32-bit (word) read
00_1_0_10_11	ldf r,MD|SZ32|PF	32-bit (word) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_11_00	ldf r,MS	Copy the status register value into an SDMA general register.

NOTE

Read data is 0-extended before writing in the SDMA general registers. When reading the MD register, the DMA takes data from the FIFO if it is available. If part or whole data is not in the FIFO, an external burst read access is performed to provide the missing data. The SDMA is stalled as long as the required read data is not complete.

After every reading, MSA is incremented by the number of read bytes from MD FIFO, except when MSA is programmed in frozen mode.

43.4.12.1.7 Prefetch/Flush and Auto-Flush Management-Burst DMA Unit

The prefetch and auto-flush management enables the SDMA RISC machine to go on while a DMA access is performed.

When the RISC core requires a prefetch (p = 1) to the Burst DMA, it will receive an immediate transfer acknowledge before the DMA has finished the external access. This enables the RISC core to do other things like accessing another DMA machine.

The basic principle in prefetch mode is for the DMA to anticipate data reads from the SDMA RISC engine by fetching external bursts of data as soon as there is enough space in the DMA FIFO to store it. If ever the RISC engine required data that is not available in the FIFO, the read acknowledge is delayed until the data is available, but it does not have to wait until the burst completes.

The auto-flush basic principle is similar: An automatic flush is triggered every time there are eight words to be written in the FIFO. If the FIFO is full and the RISC engine requires another write, it is stalled until the burst has started and enough space was freed in the FIFO to store that new data. This means the SDMA RISC engine does not have to wait for the completion of a burst to receive its acknowledge and continue its processing.

In particular, an auto-flush is executed when DMA is in write mode and if the following is true:

- If the FIFO is empty and the first write is to a word-aligned address of any size (ex: the 2 LSB of MDA[1:0]= 0x0), the auto-flush is triggered immediately after the write of the 32'nd byte.
- If the FIFO is empty, and if MDA is an odd byte address (1, 3, 5, 7,...) and an stf MD|SZ8 is executed, the byte is flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is a half-word address (2, 6, 0xA,...) and an stf MD|SZ16 is executed, the two bytes of the incoming data are flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is not a word-aligned address (ex 1, 2, 3, 5, 6, 7, 9,...), and an stf MD|SZ32 is executed, the first 1 to 3 bytes will be flushed up to the next word aligned address. Afterwards, an auto-flush will be triggered each time the FIFO receives 32-bytes.
- Therefore, if an stf MD|SZ32 is executed with MDA equal to 0x1 and with an empty MD FIFO, the bytes located at addresses 1, 2, and 3 are flushed, and the byte located at address 4 remains in MD FIFO. This solves the misalignment issue. Additionally, the next write instructions (stf) complete the FIFO until it contains eight words; then a burst write is executed by the DMA to empty the FIFO. Protocol on the external bus does not support bursts of different data types (byte, half-word, or word).

  For example, consider the case where data is written using a byte access, stf MD|SZ8. The value of MDA during the very first byte write determines when the auto-flush will occur as follows:

- If MDA=0x0, the flush occurs following the write of byte 32
- If MDA=0x1, the flush occurs following the write of byte 1, byte 3 and byte 35.
- If MDA=0x2, the flush occurs following the write of byte 2 and byte 34.

- If MDA=0x3, the flush occurs following the write of byte 1 and byte 33.
- If MDA=0x4, the flush occurs following the write of byte 32

The flush command forces the DMA to flush all MD valid bytes to the EXTMC controller. An acknowledge is sent immediately to the SDMA, and any potential error is reported on a future access. It is thus essential to conclude a transfer with a last read from MS, which will stall the core until all data was flushed out and returned to the transfer status (acknowledge or error).

NOTE

During this kind of auto-flush (which occurs only at the beginning of a misaligned write transfer) no acknowledge is sent back to the SDMA, which is stalled until a flush is completed.

43.4.12.1.8 Data Alignment and Endianness-Burst DMA Unit

43.4.12.1.8.1 Burst DMA in Read Mode

For every read access to MD, the data returned to the SDMA core and the new FIFO state depends on the MSA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is fetched externally on a 32-bit bus, but the valid bytes only are stored in the FIFO and left-aligned (for a transfer of consecutive words, it is only the first word that may be truncated). The following table shows the FIFO byte alignment strategy and the corresponding MSA, the returned data, and the new FIFO state for any access size of an internal read from MD.

Table 43-24. FIFO Read Configuration

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
00	x0 x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 and so on...	sz8	00 00 00 x0	01	x1 x2 x3 y0 y1 y2 y3 z0
		sz16	00 00 x0 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz32	x0 x1 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
01	x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 and so on...	sz8	00 00 00 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz16	00 00 x1 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz32	x1 x2 x3 y0	01	y1 y2 y3 z0

Table continues on the next page...

Table 43-24. FIFO Read Configuration (continued)

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
					z1 z2 z3 t0
10	x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 and so on...	sz8	00 00 00 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz16	00 00 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz32	x2 x3 y0 y1	10	y2 y3 z0 z1 z2 z3 t0 t1
11	x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 t2 and so on...	sz8	00 00 00 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz16	00 00 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
		sz32	x3 y0 y1 y2	11	y3 z0 z1 z2 z3 t0 t1 t2

43.4.12.1.8.2 Burst DMA in Write Mode

For every write access to the MD, the new FIFO state depends on the MDA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is stored in the FIFO according to the internal access size and the former MDA value. The following table shows the FIFO byte alignment strategy corresponding to MDA, as well as the new FIFO state for any access size of an internal write to MD.

Table 43-25. FIFO Write Configuration

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
00	tt uu vv ww ?? ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	01	tt uu vv ww x0 ?? ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	10	tt uu vv ww x0 x1 ?? ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	00	tt uu vv ww x0 x1 x2 x3 ?? ?? ?? ??
01	tt uu vv ww	sz8	?? ?? ?? x0	10	tt uu vv ww

Table continues on the next page...

Table 43-25. FIFO Write Configuration (continued)

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
	xx ?? ?? ?? ?? ?? ?? ?? and so on...				xx x0 ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	11	tt uu vv ww xx x0 x1 ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	01	tt uu vv ww xx x0 x1 x2 x3 ?? ?? ??
10	tt uu vv ww xx yy ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	11	tt uu vv ww xx yy x0 ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	00	tt uu vv ww xx yy x0 x1 ?? ?? ?? ??
		sz32	x0 x1 x2 x3	10	tt uu vv ww xx yy x0 x1 x2 x3 ?? ??
11	tt uu vv ww xx yy zz ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	00	tt uu vv ww xx yy zz x0 ?? ?? ?? ??
		sz16	?? ?? x0 x1	01	tt uu vv ww xx yy zz x0 x1 ?? ?? ??
		sz32	x0 x1 x2 x3	11	tt uu vv ww xx yy zz x0 x1 x2 x3 ??

NOTE

If the FIFO mode changes from a write to a read mode, all remaining written bytes in MD are lost but no error is returned. Typically, this happens if an ldf MD is executed after stf MD instructions. Before a mode change, it is recommended to force the flush of a potential remaining byte by a stfMD|SZ0|FL instruction. In the same way, if a FIFO mode changes from a read to a write mode, all prefetched data present in the FIFO is lost and no error is returned.

43.4.12.1.8.3 Endianness-Burst DMA Unit

Big and Little Endian are supported by the Burst DMA, but data is always stored in MD in Big Endian.

Byte manipulation is performed when data is exchanged with an Burst controller (for example, during read or write burst accesses).

43.4.12.1.9 Burst DMA Unit Copy Mode

A mechanism is available to perform fast ARM-to-ARM transfers.

Data does not flow through the SDMA core: It is kept in the DMA FIFO. This mechanism is selected when writing MD with a special option in the instruction code (copy flag).

It is possible to transfer up to eight words in one SDMA instruction (this does not mean in one cycle). In this mode, every time an stf MD|CPY is executed, a read burst is executed and directly followed by a write burst transfer. Burst transfers are limited to eight words. The size of the transfer (in words)-given by the SDMA general register (4 LSB)-is also limited to eight. The following SDMA code shows how 100 bytes could be copied from the MSA address to the MDA address. This is sample code only.

Burst DMA copy mode example

```
          ldi r0,@src
          stf r0,MSA                          // Source address setup
          ldi r1,@dst
          stf r1,MSA                          // Destination address setup
          ldi r0,0x64                         // data transfer counter
          ldi r1,0x8
MAIN_XFER:
          cmphs r0,r1                         // Is r0 >= 0x8
          bf LAST_XFER                      // If not, jump to last transfer label
          stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.
          subi r0,0x8                         // Decrement counter
          jmp MAIN_XFER                     // return to main transfer loop
LAST_XFER:
          stf r0,MD|CPY
```

The main transfer loop is executed 12 times; then r0 equals 4 and the last transfer loop is run.

In this mode, an acknowledge is transmitted to the core as soon as the read burst can start; thus, a first copy instruction returns an immediate acknowledge and subsequent copy instructions will be acknowledged as soon as the previous copy has finished.

43.4.12.1.10 Burst DMA Unit Error Management

Another point to consider is the management of errors.

Because the DMA immediately sends an acknowledge to the RISC core (except for the stf MS|SZ0|FLS instruction), it assumes no error will occur. If an error occurs, it is flagged (transfer error acknowledge) for the following DMA access.

This should not be a problem if the DMA is used properly. The MD accesses are meant to stall the SDMA as little as possible to optimize throughput and hide calculation time. Therefore, final access to MS should be performed before closing a channel. This access waits until any pending operation is finished in the burst DMA and gather any remaining error.

In copy mode, an error could be immediately returned to the SDMA on execution of the ldf copy or stf copy instruction. It happens when MSA or MDA are not word addresses (for example, 0[4]). This is because copy mode must only be used for transferring a large packet of aligned data.

When an error is received during a *read* transfer to the external bus, which may occur during the burst accesses, the MD FIFO contains the valid beats of the burst, and the error flag of MS is set to 2'b11 (error read burst). It is possible to read MS ("n" field) to know how much valid data remains in MD and when MD is empty (after ldf instructions). The next read MD instruction sets the MS error flag to 2'b10 (error mode), and an error is sent back to the SDMA core. In error mode, it is possible to read MSA, which gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, gives rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

In "error read burst" mode, writing MDA, MSA, or MD, or starting a copy transfer by a stf MD|COPY instruction will cancel the error mode. The following table shows when an immediate error is sent back according to the executed instruction.

Table 43-26. Possibilities in ERROR READ BURST Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA (|U |PF) stf rn, MDA stf rn,MD|COPY	NO	Error mode is reset. MSA, MDA, or MD are updated and a DMA cycle may start. For the stf MD|COPY, a copy loop is executed.
stf rn, MS	NO	MS is updated.

Table continues on the next page...

Table 43-26. Possibilities in ERROR READ BURST Mode (continued)

DMA Instruction	Immediate Error	Comments
ldf rn, MS ldf rn, MSA ldf rn, MDA	NO	MS, MSA, and MDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	YES/NO	Immediate error if there is no more data available for read in the FIFO.

When an error is received during a *write* transfer, the error is reported to the next DMA access. In this case, an error is sent to the SDMA core and the DMA goes to its error mode. Reading MS gives the number of bytes that remain in MD; reading MDA gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, give rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

Table 43-27. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA stf rn, MDA	Yes	Any attempt to modify MD, MSA, MDA will raise an immediate error and burst DMA remains in error mode. When address registers are write-accessed, an error is returned.
stf rn, MS	No	This is the only way to exit error mode. MS[9:8] must be reset by an stf MS|SZ0 instruction.
ldf rn, MS ldf rn, MSA ldf rn, MDA	No	MS, MSA, and MDA could be read in error mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	Yes	Whatever the DMA direction (read or write), an ldf rn triggers an immediate error.

43.4.12.1.11 Conditional Yielding-Burst DMA Unit

The standard SDMA transfer is based upon a hardware loop that has the following structure:

Hardware Loop

```
loop
load Rn,source            // can be ldf or ld
<computation>            // can be done through functional units
store Rn,dest              // can be st or stf
done 0               // yield
```

This structure needs to be kept independent of the functional units' particularities regarding the context switch. However, there can be variations in the context switch's efficiency, which can depend on the number of data received up to that point, and on the data itself.

The DMA, with its 8-word burst capability, has a preferable context switch period when its address register is 8-word aligned: It is the only moment that occurs once every eight loops when the succession of bursts is not broken by the context switch. When this is not the case, a context switch requires the storing (or loading) of less than eight words, which requires separate accesses and is far less efficient. The rest of the 8-word packet is stored (or loaded) after the context restore, and this is done as separate accesses.

The proposed solution is a conditional yielding, which occurs only when the DMA is in an optimum state. It does not require any modification to the scripts. The condition is decided at the DMA level.

The DMA can be programmed in two modes-conditional or always-true-for every channel, which provides complete flexibility. By default, the DMA is not in conditional mode.

The DMA condition is computed from the FIFO fill level and the various modes, as follows:

- When copy mode is selected, regardless of the transfer direction ('read' or 'write'), the condition is always true.
- In read mode, the condition is always true.
- In write mode, the condition is true when there are four bytes or less in the FIFO; it is false when there are more than four bytes. The 4-byte limit comes from the possibility of saving those bytes as MD with absolutely no impact on the bus accesses.

The aim at conditional yielding is to avoid splitting bus accesses (especially bursts).

43.4.12.2 Peripheral DMA Unit Programming

The peripheral DMA unit is connected to the Multi-Layer DMA Crossbar Switch of the ARM platform.

Its goal is to perform data transfers between any blocks connected to the DMA bus of this platform. These blocks are either peripherals or memories. The peripheral DMA could be seen as the ARM platform DMA controller.

The DMA performs data transfers in three modes:

- Read mode, where data is read from peripherals or from memory connected to the ARM platform and copied in a SDMA general register.
- Write mode, where data of a general register has to be written in a peripheral or a memory.
- Copy mode, where data is read from a peripheral (or memory) at a source address (PSA) and automatically written to a peripheral (or memory) at a destination address (PDA).

In copy mode, no SDMA general register is involved as transferred data only goes through the data register of the DMA.

The peripheral DMA has three addressing modes: frozen, incremented, and decremented, as follows:

- Frozen mode-When source or destination addresses are frozen, their value is not modified after a transfer. This mode is typically used for addressing peripheral FIFOs located at a fixed address.
- Incremented mode-When source or destination addresses are in incremented mode, after every transfer they are incremented by the number of bytes transferred.
- Decremented mode-In decremented mode, addresses are decremented by the number of bytes transferred.

The peripheral DMA registers are as follows:

- Two, 32-bit address registers (PSA and PDA) that respectively contain the source address for a read access and the destination address for a write access
- A 32-bit status register (PS) that contains information on the peripheral DMA configuration, such as the number of valid bytes in the data register, the error flag, the source and destination address mode, and so on.
- A 32-bit data register (PD) that stores data involved in a data transfer

43.4.12.2.1 Peripheral Source Address Register (PSA)

The source address register contains a pointer to a source peripheral or a memory associated with the next read data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PSA) to store the address value
- A 2-bit register (stype) to store the source address mode (frozen, incremented, or decremented)
- A 2-bit register (ssize) to store the source target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects and gives the address value of the next data that will be read by the SDMA during an ldf MD instruction. Writing the source address register may have side effects. If there is valid write data in the data register and the source address is changed, the write data is discarded. If the prefetch bit is set, a DMA read cycle is issued with the new address.

When PSA is to be written, you must specify the source target address mode, providing its size (byte, half-word, or word). This enables omission of the size field in all ldf MD instructions. When DMA performs a read cycle, its size is given by the value of the PSA source size register (ssize). If source is a memory in incremented mode, first programmed in word mode (stf PSA|SZ32|I), and if an SDMA script needs to read bytes from this memory, the size of the source target must be updated before executing new accesses. The source address mode and its size are given by labels added to the stf PSA instruction as described in the write section. The ssize and stype registers are part of the DMA status register (PS).

Writing to PSA may issue an immediate error if the source size is not compatible with the value to be written into the PSA register. For instance, writing a 2 in PSA and specifying that it is memory-accessed in word mode creates an immediate error.

43.4.12.2.2 Peripheral Destination Address Register (PDA)

The destination address register contains a pointer to a source peripheral or a memory associated with the next write data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PDA) to store the address value
- A 2-bit register (dtype) to store the destination address mode (frozen, incremented, or decremented)
- A 2-bit register (dsize) to store the destination target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects, and gives the address value of the next data that will be written by SDMA during an stfMD instruction. Writing the destination register has no side effect. Similar to the PSA register, the destination address mode and source are specified in the stf PDA instruction and may also generate an error in case of incorrect programming.

43.4.12.2.3 Peripheral Data Register (PD)

The data register of the peripheral DMA is a 32-bit register. When the destination address is correctly set up, any writing to PD will automatically flush the new input data.

The number of SDMA bytes that will be transferred is given by the PDA size register. Unlike other SDMA DMAs, PD is not a FIFO: It is not used to accumulate bytes that from the SDMA and must be packed before being sent to external memories. In read mode, and if the source address is correctly set up, an ldf instruction will empty PD. If a prefetch is required along with the instruction, the DMA will initiate a new read transfer.

Reading PD in prefetch mode only stalls the SDMA when the prefetched data is not yet available. Writing PD only stalls the SDMA if the previous write operation was not completed. As soon as the previous operation is over, the acknowledge is sent back to the SDMA RISC engine.

An error flag-part of PS-is set when an external access fails. The error is thus reported to the next SDMA instruction that involves the peripheral DMA.

43.4.12.2.4 Peripheral State Register (PS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer.

Although all PS fields can be written by an stf instruction, it is recommended to access only the error bit (to reset it). Modifying other PS fields will provide an un-guaranteed DMA behavior.

The initialization value of PS is 0, and it consists of the following structure:

Table 43-28. PS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ssize		stype		dsize		dtype	
W																
R	0	0	0	0	0	d	e		0	0	0	0	0	n		
W																

Table 43-29. PS Field Descriptions

Field	Description
31-24	Reserved
23-22 ssize	Source Target Size. Determines the size of the read transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

Table continues on the next page...

Table 43-29. PS Field Descriptions (continued)

Field	Description
21-20 stype	Source address Mode. Determines whether PSA is incremented, decremented, or kept unmodified after every read from the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
19-18 dsize	Destination Target Size. Determines the size of the write transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
17-16 dtype	Destination address Mode. Determines whether PDA is incremented, decremented, or kept unmodified after every write on the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
15-11	Reserved
10 d	Direction Flag or DMA Mode. DMA is in write mode when data was written into PD by stf PD instructions, or if a previous DMA cycle on the external bus was a write access. Writing PDA or PSA does not change the DMA mode. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by the SDMA with an ldf PD instruction. Reading PDA or PSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 Error read
7-3	Reserved
2-0 n	number of bytes in PD

NOTE

dtype, dsize, stype, and ssize are updated when PSA and PDA are written.

43.4.12.2.5 Peripheral DMA Write (stf)-Write Mode

When written by an stf instruction, the function code bits are interpreted as follows:

Table 43-30. STF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s		p	ar	am		sz	
PDA								
PD			pdsel					
PS			pssel					

Table 43-31. STF Code Bits Field Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PSA)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
3-2 am (PSA/PDA)	Address Mode. Determines how PSA or PDA is modified after every read or write access to the PD. 00 Frozen-Address registers are not modified after the transfer. 01 Incremented-Address registers are incremented by the number of transferred bytes. 10 Decremented-Address registers are decremented by the number of transferred bytes. 11 Updated-PSA and PDA are not modified. Either address mode is not modified, but the width of the data path is updated by the sz field.
1-0 sz	Transfer Size 00 *reserved* 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
5-0 pdsel	PD access selector 001000 is the only valid option
5-0 pssel	PS access selector 111111 writes to PS 001100 only clears the error flag in PS

Due to the large number of possible stf instructions, the following table provides only a short list of all the possible write instructions:

Table 43-32. Peripheral DMA STF Instruction List

Binary	Assembly	Comments
11_00_00_01 11_00_00_10 11_00_00_11	stf Rn, PSA|SZ8 |F stf Rn, PSA|SZ16|F stf Rn, PSA|SZ32|F	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is frozen.
11_10_00_01 11_10_00_10 11_10_00_11	stf Rn,PSA|SZ8 |F|PF stf Rn,PSA |SZ16|F|PF stf Rn,PSA |SZ32|F|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • 1, 2, or 4 bytes are *fetched* from the peripheral source. • Source address is frozen.
11_00_01_01 11_00_01_10 11_00_01_11	stf Rn, PSA|SZ8 |I stf Rn, PSA|SZ16|I stf Rn, PSA|SZ32|I	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1,2 or 4 after read PD.
11_10_01_01 11_10_01_10 11_10_01_11	stf Rn, PSA|SZ8 |I|PF stf Rn, PSA|SZ16|I|PF stf Rn, PSA|SZ32|I|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1, 2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_10_01 11_00_10_10 11_00_10_11	stf Rn, PSA|SZ8 |D stf Rn, PSA|SZ16|D stf Rn, PSA|SZ32|D	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD.
11_10_10_01 11_10_10_10 11_10_10_11	stf Rn, PSA|SZ8 |D|PF stf Rn, PSA|SZ16|D|PF stf Rn, PSA|SZ32|D|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_11_01 11_00_11_10 11_00_11_11	stf Rn, PSA|SZ8 |U stf Rn, PSA|SZ16 |U stf Rn, PSA|SZ32 |U	• *Update* source pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost.
11_10_11_01 11_10_11_10 11_10_11_11	stf Rn, PSA|SZ8 |PF|U stf Rn, PSA|SZ16 |PF|U stf Rn, PSA|SZ32 |PF|U	• *Update* source pointer, which becomes a pointer to a target accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost. • 1, 2, or 4 bytes are *fetched* from the memory source.
11_01_00_01 11_01_00_10 11_01_00_11	stf Rn, PDA|SZ8 |F stf Rn, PDA|SZ16|F stf Rn, PDA|SZ32|F	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is frozen.
11_01_01_01 11_01_01_10 11_01_01_11	stf Rn, PDA|SZ8 |I stf Rn, PDA|SZ16|I stf Rn, PDA|SZ32|I	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA + 1, 2, or 4 after write PD.

Table continues on the next page...

Table 43-32. Peripheral DMA STF Instruction List (continued)

Binary	Assembly	Comments
11_01_10_01 11_01_10_10 11_01_10_11	stf Rn, PDA|SZ8 |D stf Rn, PDA|SZ16|D stf Rn, PDA|SZ32|D	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA-1, 2, or 4 after write PD.
11_01_11_01 11_01_11_10 11_01_11_11	stf Rn, PDA|SZ8 |U stf Rn, PDA|SZ16 |U stf Rn, PDA| SZ32 |U	• *Update* destination pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PDA value is not modified by Rn • bytes present in PD are lost
11_00_10_00	stf Rn, PD	• Write "dsize" bytes of Rn in PD and automatically flush to destination target
11_11_11_11	stf Rn, PS	• Write status register
11_00_11_00	stf Rn,clrefPS	• Clear error flag if set

NOTE

When writing PD, size information is not important: It is embedded in the dsize field of PDA register. If dsize is 1, 2, or 4, then one, two, or four bytes from Rn is written to the PD register, and automatically flushed out to the destination target.

43.4.12.2.6 Peripheral DMA Read (ldf)-Read Mode

When received from an ldf instruction, the function code bits are interpreted as follows.

Table 43-33. LDF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s			ar	a			
PDA								
PD			p	cpy				
PS			pssel					

Table 43-34. LDF Code Bits Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA

Table continues on the next page...

Table 43-34. LDF Code Bits Descriptions (continued)

Field	Description
	1 PDA
4 cpy (PD)	Copy Mode 0 standard access 1 copy mode access
3 a	Register Set selection 0 PSA or PDA 1 PD or PS
5-0 pssel	PS access selector 111111 is the only valid option to read PS

Table 43-35. Peripheral DMA LDF Instruction List

Binary	Assembly	Comments
11_0_0_0_000	ldf Rn, PSA	Reads 32-bit of PSA value
11_0_1_0_000	ldf Rn, PDA	Reads 32-bit of PDA value
11_0_0_1_000	ldf Rn, PD	Reads programmed source size bytes of PD (0-extended)
11_1_0_1_000	ldf Rn, PD|PF	Reads programmed source size bytes of PD (0-extended), and starts a prefetch at PSA address.
11_0_1_1_000	ldf Rn, PD|COPY	Starts a copy transfer from the source target at the PSA address to the destination target at the PDA address. No data transmits through Rn, but Rn contents are lost (Rn is loaded with PD temporary contents that are *not* the copied data).
11_111111	ldf Rn, PS	Reads 32-bit of PS value

NOTE

When reading PD, size information is not important: It is embedded in the ssize field of the PSA register. If ssize is 1, 2, or 4, the one, two, or four bytes is transferred from PD to Rn. Read data is 0-extended.

43.4.12.2.7 Peripheral DMA Unit Copy Mode

Like burst DMA, the peripheral DMA unit has a copy mode that is used when data transfers do not involve SDMA general registers.

Data is read from the source target at a PSA address, stored in PD, and then automatically flushed to the destination target at the PDA address. Copy mode is only available for transfers that involve two targets of the same data path width.

Since copy mode is invoked with an ldf instruction, the *loaded* general purpose register loses its previous contents. (However, the new contents are unpredictable as they depend on temporary values that are seen on the external DMA bus.)

43.4.12.2.8 Error Management

Peripheral DMA generates two kinds of errors: the immediate error that sanctioned incorrect register programming; and the error triggered by the previous access and stored in the error flag of PS until a DMA instruction is executed.

43.4.12.2.8.1 Immediate Errors

The following table lists all incorrect DMA register setups.

Table 43-36. Immediate Errors with Peripheral DMA

Rn[1:0] values	DMA instruction	Comments
0x01 0x11	stf Rn, PSA|SZ16|F stf Rn, PSA|SZ16|I stf Rn, PDA|SZ16|F stf Rn, PDA|SZ16|I	If PSA points to a half-word peripheral or to a half-word address in memory, its value must be 0 modulo 2.
0x01 0x10 0x11	stf Rn, PSA|SZ32|F stf Rn, PSA|SZ32|I stf Rn, PDA|SZ32|F stf Rn, PDA|SZ32|I	If PSA points to a word peripheral or to a word address in memory, its value must be 0 modulo 4.
PSA[1:0]-PDA[1:0]	DMA instruction	Comments
0x01 0x10 0x11	stf Rn, PSA|SZ32|U stf Rn, PDA|SZ32|U	When PDA or PSA is updated and becomes a pointer to a word address in memory, its content must be 0 modulo 4.
0x01 0x11	stf Rn, PSA|SZ16|U stf Rn, PDA|SZ16|U	When PDA or PSA is updated and becomes a pointer to a half-word address in memory, its content must be 0 modulo 2.
Read/Write PD instruction	Comments	
stf Rn,PD ldf Rn,PD	If PDA size (dsize) has never been set up before an stf PD instruction (dsize=0) If PSA size (ssize) has never been set up before an ldf PD instruction (ssize=0)	
ldf Rn,PD|CPY	Copy mode is possible only between two targets whose data path width is identical. It is P8↔P8, P16↔P16, or P32↔P32 regardless of the way the address registers are incremented.	

43.4.12.2.8.2 Data Transfer Errors

When PSA and PDA are correctly set up, the only error that may arise for an ldf PD or stf PD instruction would be the error of the previous DMA cycle.

Error handling is driven by a single consideration: When an error occurred during a data read on the DMA interface, this error should appear as a transfer error to the core when the core attempts to retrieve the data that was not successfully read from the accessed device (memory or peripheral).

When an error occurred during a write access to the DMA interface, the data is still available in PD and should not be destroyed by subsequent core accesses: The core must be warned about the error issue.

There are three error handling mechanisms for each case: Read Error (First Phase), Write Error and Read Error (Second Phase), and Copy Mode Errors handling.

43.4.12.2.8.3 Read Error (First Phase)

If an error occurred during a prefetch command, the peripheral DMA enters its ERROR READ mode (PS[9:8]=11). In this mode, the error is reported on the next ldf PD instruction and writing PSA, PDA, or PD will cancel the error flag.

The block returns no error mode and instructions are normally executed (a DMA cycle may be triggered). Similarly, initiating a copy transfer will reset the error flag and start a copy transfer. The following table details which instructions can be executed in this mode.

Table 43-37. Possibilities in ERROR READ Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA (IU IPF) stf rn, PDA ldf rn,PD|COPY	NO	Error mode is reset, PSA or PDA are updated, or a write cycle is started. For the ldf PD|COPY, a copy loop is executed.
stf rn, PS	NO	PS is updated.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Error of the previous read access is reported here and the peripheral DMA enters its ERROR mode.

43.4.12.2.8.4 Write Error and Read Error (Second Phase)

The peripheral DMA enters its ERROR mode (PS[9:8]=10) when the previous DMA write cycle failed, or, as explained in Read Error (First Phase), when an ldf PD is executed while the block is in ERROR READ mode. When a DMA cycle failed, address registers (PSA, PDA) are not modified and continue to point to the problematic address. In ERROR mode, stf instructions may raise an immediate error, and ldf instructions will not (as detailed in the table below).

Table 43-38. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA stf rn, PDA	YES	Any attempt to modify PD, PSA, or PDA will raise an immediate error, and the peripheral DMA stays in ERROR mode. When address registers are write accessed, an error is returned.
stf rn, PS	NO	This is the only way to exit the ERROR mode. PS[3] must be reset by an stf PS instruction.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Whatever the DMA direction (read or write), an ldf rn, PD instruction will show an immediate error.

43.4.12.2.8.5 Copy Mode Errors

Because copy mode is a write access that follows a read access, there are two possible cases of bus error.

When the read access incurs a bus error, the peripheral DMA behaves exactly as described in Read Error (First Phase) and Write Error and Read Error (Second Phase) : It enters its ERROR READ mode, and so on.

When the error occurred during the write access of the copy transfer, the DMA enables the core to retrieve the data that was read because it is assumed the read from the peripheral removed the data from its source device. Therefore, the data to be flushed is still in PD. Any subsequent access to PD triggers an error to the core, which should execute its error handling procedure.

Once the ERROR mode is left (after writing to PS), it is possible for the core to retrieve the data in PD with an ldf instruction or try to flush PD contents once again (for example, when the error was due to a full FIFO and the script waited for the FIFO to be emptied) with another ldf instruction in copy mode. This latter instruction detects that there is valid data in PD, tries to flush it, and thus skips the read phase of the copy instruction. This is a different behavior from the usual stf PD instruction that overwrites PD with the selected General Purpose register contents. The same mechanism can be used any time PD holds data that is not written because of a bus error on the DMA interface; when the data was written via a copy instruction, or via the usual stf PD instruction.

43.4.12.2.8.6 Error Check Example

The following code illustrates an example checking for both immediate and data transfer errors on a store to the PD register. The first bdf instruction checks for an immediate error, but if a data transfer error occurred it is reported until the next instruction to access the Peripheral DMA. A second check of the error flags is done after the ldf PS

instruction. The value of PS here can be ignored. The act of reading any register in Peripheral DMA while it is in an error mode that returns the error to the core to set either the SF or DF flag. Any error returned on an ldf command sets the SF flag and any error returned on an stf instruction sets the DF flag. This can create a situation as shown in the example where a bus error during a DMA write which would normally be considered as a destination fault is reported as a source fault because the error was reported to the SDMA core during an ldf instruction.

Peripheral DMA Error Check

```
    clrf    0            // Clear SF and DF flags
 stf    R4, PD        // Write data to memory
bdf    error_routine    // Check for immediate error from write to PD.
ldf    r3, PS        // Read PS (PS value in R3 can be ignored)
    bsf    error_routine    // Check for bus error from "stf R4,PD"
            // SF is set because it is a ldf instruction, even though
            // the original error was a destination fault
```

43.4.12.2.9 Peripheral DMA Unit Prefetch/Flush Management

There is no flush bit because every time data is stored in PD by a stf PD instruction-assuming PDA is correctly programmed-it is automatically flushed to the destination.

An acknowledge is returned in the cycle of the DMA instruction, and the SDMA is only stalled by an instruction that addresses the peripheral DMA when the previous DMA access is not over.

43.4.12.3 OnCE and Real-Time Debug

The On-Chip Emulation block (OnCE) is the debug interface to the SDMA.

It supports the access to all core internal devices (registers, memory, and so on), and provides a set of mechanisms that control the core. The OnCE is accessed by JTAG ports at the chip's board level, or by the host via its peripheral bus.

To reduce the size of the hardware material involved, all tasks supported by the OnCE are performed on the SDMA core. The architecture of the SDMA OnCE is relatively simple and very flexible.

The commands supported by the SDMA OnCE are listed in the following sections.

43.4.12.3.1 Memory and Register Access

A set of mechanisms is provided to access SDMA memory and register locations. Both reading and writing are allowed. The access is supported if the processor is in debug mode.

Those registers can also be accessed through the ARM platform Control interface when the OnCE is controlled by the ARM platform, as described in the "Using BP" section.

43.4.12.3.2 Hardware Breakpoints

An event detection unit is implemented to support memory breakpoints. The unit watches the data exchanged between the SDMA memory bus and the core.

A debug request is sent to the core when matching conditions occur. The unit supports mixed conditions based on address range, access type, and data value. Event detection unit configuration registers are memory mapped in the SDMA space (see ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)): You can modify them through a regular memory access or the ARM platform control interface.

43.4.12.3.3 Watchpoints

One output pin is provided to monitor matching trigger conditions that are defined in the event detection unit.

43.4.12.3.4 Software Breakpoints

The SDMA instruction set contains a software breakpoint. Upon executing a software breakpoint instruction, the core suspends normal execution and enters debug mode.

No hardware step execution mode is implemented in the OnCE, but this feature may be implemented at the software level with this instruction.

43.4.12.3.5 Core Control

Commands are provided to monitor and control processor activity. You can halt the core, rerun the core from another address location, and get processor status.

Any hardware breakpoint on the instruction bus is not supported, but this feature may be implemented by inserting a software breakpoints program.

43.4.13 The OnCE Controller

The OnCE controller receives commands from the ARM platform or from the JTAG controller. Each command is interpreted before being sent to the core.

43.4.13.1 OnCE Commands

A small set of commands supports the communication between the OnCE and the external world.

This command set enables you to perform any of the following tasks: control processor activity, save core context, and execute an SDMA instruction from the OnCE. Combined together, these tasks perform more complex commands.

A full OnCE command contains a 4-bit instruction (the OnCE command opcode) and a variable length data field (the OnCE data). During command execution, the OnCE data is transferred in a OnCE internal register before being exchanged with the SDMA. Some data values are also exported. This mechanism creates a link between the processor and the external world. Nine commands are defined: The following table presents their formats.

Table 43-39. OnCE Command Opcode Values

Instruction Opcode	Name	Action	Register	Data Field Size	Mode
0000	rstatus	Reads the OnCE status register	STATUS	16-bit	normal/debug
0001	dmov	Updates general register GReg1	GREG1	32-bit	debug
0010	exec_once	Runs the instruction from the SDMA instruction register	INSTRUCTION	16-bit	debug
0011	run_core	Returns to normal execution	BYPASS	1-bit	debug
0100	exec_core	Returns to normal execution via a jump instruction that specifies the new address	INSTRUCTION	16-bit	debug
0101	debug_rqst	Stops the core after execution of current instruction	BYPASS	1-bit	normal
0110	rbuffer	Reads the real time buffer	RTB	32-bit	normal/debug
0111-1110	reserved	Reserved	BYPASS	1-bit	normal/debug
1111	bypass	Bypasses TARM platform controller	BYPASS	1-bit	normal/debug

Each instruction corresponds to a specific action performed on the OnCE. The nature of the associated data field is clearly identified. The dmov command is followed by a 32-bit data value (which is a data value for the SDMA); the exec_once and the exec_core commands are followed by a 16-bit data value (which is an instruction for the SDMA); the rstatus command is followed by a 16-bit control value (which is the content of the OnCE status register); the rbuffer command is followed by a 32-bit data value. The

debug_rqst and the run_core commands are followed by a single bit data field (this is a bypass value). Finally, the bypass instruction enables the SDMA JTAG TAP controller to be daisy-chained with another JTAG TAP controller. This is a JTAG-only feature. The set of commands is simple, but enables you to perform any possible task on the SDMA during a debug process.

43.4.13.2 Sending Commands to the OnCE Controller

The JTAG access is the standard access to the OnCE, but sometimes the JTAG is not available to fix some bugs (if the chip is in production for instance), an additional access is then required. Therefore, one ARM platform access to the OnCE is provided.

43.4.13.2.1 Using the JTAG Interface

A serial access is performed through the five JTAG pins TCK, TRST, TMS, TDI, and TDO. A Test Access Port controller is provided to decode the TMS control signal.

It produces shift-enable signals (shift_ir and shift_dr), and updates enable signals (update_ir and update_dr). It is fully compliant with the IEEE 1149.1 testability (JTAG) standard.

During the shift_ir state, the command opcode is shifted into the OnCE controller (for example, the signal from the TDI pin is shifted into the command register and the TDO pin receives the signal shifted out). After transferring the four bits of the command, an update_ir signal is asserted and the command is decoded. The target data register is now clearly identified and the corresponding control signal is produced, as follows: bypass enable signal (bp_en), instruction enable signal (inst_en), data enable (data_en), and status enable signal (stat_en).

During the shift_dr state, the TDI signal is shifted into one of the following target registers: bypass register (1 bit), SDMA instruction register (16 bits), SDMA data register (32 bits), or OnCE status register (16 bits). The TDO pin is connected to the output of the selected register to receive the signals shifted out.

The JTAG access is disabled when the ARM platform access is enabled.

43.4.13.2.2 Using the ARM platform

The ARM platform access to the OnCE is not the standard access, but it is required if the JTAG is not available.

For example, if the SDMA ROM is out of use on a chip in production, and the ARM platform needs to download new code and restart the SDMA, the OnCE can easily perform this operation. This type of debug operation justifies the use of an ARM platform access to the OnCE.

To drive the OnCE, the ARM platform uses some registers contained in the ARM platform Control block of the SDMA. These registers are accessed through the ARM platform peripheral bus. Most of these registers are connected to another register in the OnCE controller. Thus, accessing one of these registers is equivalent to accessing the associated register in the OnCE controller.

The set of registers in the ARM platform Control block is listed below:

- ONCE_ENB register (1 bit, read/write)-This 1-bit register enables the ARM platform access to the OnCE. When this bit is set, the signals from the JTAG are ignored. When it is cleared, all writing operations to the following registers through the Host Control interface are ignored. This register is reset on a JTAG reset.
- ONCE_CMD register (4 bits, read/write)-This 4-bit register receives the command opcode. It is connected to the command register in the controller. A write access to this register causes the associated command to be executed on the OnCE. For example, after writing "0001" in this register, a dmov command is executed.

NOTE

On the ARM platform side, the rstatus and bypass commands are not supported. This register is reset on a JTAG reset.

- ONCE_DATA register (32 bits, read/write)-This 32-bit register is connected to the SDMA data register. This register is used when executing a dmov or rbuffer command.

NOTE

Before requesting a dmov command, the 32-bit data to transfer must be written in the ONCE_DATA register. At the end of the execution, the register is updated with GReg1 former value. This register is reset on a JTAG reset.

- ONCE_INSTR register (16 bits, read/write)-This 16-bit register is connected to the SDMA instruction register. This register is used when executing an exec_core or an exec_once command.

NOTE

Before requesting an exec_core or an exec_once command, the appropriate instruction must be written in the ONCE_INSTR register. This register is reset on a JTAG reset.

- ONCE_STAT register (16 bits, read only)-A read access to the ONCE_STAT register returns the content of the OnCE status register (OSTAT). This register is read only.
- The bypass register is not useful when the ARM platform controls the OnCE, therefore no register is defined in the ARM platform Control block to access the bypass register.

43.4.13.2.3 Conflicts Between the JTAG and the ARM platform Accesses

When ARM platform access to the SDMA OnCE is enabled (that is, when the bit in the ONCE_ENB register is set), the JTAG access is disabled. This guarantees that the block is not accessed at the same time on both sides.

It is possible to check whether the JTAG access to the SDMA OnCE is enabled from the JTAG port. When the JTAG access is disabled, the SDMA TDO always returns 1. The check requires the following steps:

- Execute a dmov command from debug mode (with neither 0xffffffff nor 0x0 as dmov value: 0x5a5a5a5a is good).
- Execute another dmov command (the value here is not important).
- The returned value from the latter dmov command should be the original one if the JTAG access is enabled; if it is 0xffffffff instead of the original input value, this means the JTAG access is disabled.

43.4.13.3 Executing a Command from the OnCE

All the commands defined in OnCE Commands can be accessed through the JTAG. The ARM platform can access all these commands except the rstatus command.

On the ARM platform side, the OnCE status is directly accessed by reading the ONCE_STAT register.

43.4.13.3.1 Nature of the Commands

Two types of commands may be distinguished. First, there are two commands that do not interact with the core: rstatus and rbuffer. Those commands may be requested at any time: They do not depend on the core status.

NOTE

Each of these commands exports a data value or a status value from the SDMA.

There are also commands that interact with the core: dmov, run_core, exec_core, exec_once, and debug_rqst. These commands are core status dependent, as follows:

- During user mode only the debug_rqst is taken into account.
- During debug mode, all these commands are taken into account except the debug_rqst. For example, an exec_once command requested while not in debug mode has no effect.

43.4.13.3.2 Execution Request

The SDMA starts executing a task in debug mode when requested by the OnCE controller. The execution starting time depends on the type of access used to communicate with the OnCE.

If the JTAG is used, the request is send after decoding the update_dr state in the TAP controller. Therefore, always cross this state when sending a command through the JTAG. If the OnCE is driven from the ARM platform side, the request is sent after detecting a write access to the ONCE_CMD register. All the registers involved in this operation must be loaded first.

The following is an example of an exec_core command execution from the ARM platform side: After writing '010' in the ONCE_CMD register, the OnCE controller asks the SDMA to execute the instruction contained in the ONCE_INSTR register. The instruction involved should be available in the ONCE_INSTR register before the beginning of the execution.

43.4.13.3.3 Command Execution

The following list shows the commands and details how each command is executed:

- rstatus command execution-The rstatus command exports the content of the OnCE status register (OSR). If the JTAG is used, the status information is captured in the OnCE status register during the capture_dr state, and shifted out after 16 TCK clock cycles in the shift_dr state. The rstatus command is not supported on the ARM platform side, but a status register is provided instead. The rstatus may be performed in both debug and user modes.
- dmov command execution-The dmov command accesses SDMA internal registers. Executing a dmov instruction exchanges the 32-bit data values between the SDMA data register and the general register GReg[1].
- If the JTAG is used, the content of GReg1 is captured in the SDMA data register during the capture_dr state, then it is shifted out after 32 TCK clock cycles in the shift_dr state. During the update_dr state, GReg1 is updated with the new, shifted-in 32-bit data value. If the OnCE is driven from the ARM platform side, the data values

contained in GReg1 and the SDMA data register are exchanged after detecting a write access to the ONCE_CMD register. The ONCE_DATA register must therefore be loaded first.

- exec_once command execution-The exec_once command executes the instruction loaded in the SDMA instruction register. The command may only be requested from debug mode. The SDMA returns to debug mode at the end of the execution.
- Change of flow instructions as well as instructions that may cause a context switch are not supported: The comprehensive list comprises done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions.

  No other command should be requested before the SDMA returns to debug mode. The SDMA status (for example, whether it is in debug mode or not) can be detected by polling with the rstatus OnCE command, monitoring the debug_mode pin, or checking the OnCE Status Register (SDMAARM_ONCE_STAT) register via the ARM platform control interface.

NOTE

Most of the instructions are single-cycle, which omits the step of polling the status. Loads and stores to DMA units are typical instructions that might require this polling.

If the JTAG is used, the 16-bit instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. A request is sent to the core when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the request is sent to the SDMA when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must be therefore be loaded first.

- run_core command execution-The run_core command leaves debug mode and resume normal program execution. The next instruction executed is the last instruction decoded before entering debug mode. Be sure to restore core context before re-running the core. This procedure is detailed in Restoring the Context.
- If the JTAG is used, a 1-bit bypass value is shifted in the bypass register in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the core is rerun when detecting a write access to the ONCE_CMD register.

- exec_core command execution-The exec_core command resumes program execution from any address. The 16-bit instruction provided with the exec_core overwrites the last instruction decoded before entering debug mode. This command is designed to support change of flow instructions, so that a program execution can be restarted

from any address. After executing an exec_core command, the SDMA leaves debug mode. The exec_core command is usually used with a jmp instruction.

- If the JTAG is used, the 16-bit branch instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the SDMA reruns when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must therefore be loaded first. For example, to restart the SDMA from the program address 0x100, the instruction loaded should be a jump to address 0x100 instruction.
- debug_rqst command execution-The debug_rqst command puts the SDMA in debug mode. If the JTAG is used, a 1-bit bypass value is shifted in the bypass register during the shift_dr state. A debug request is sent to the SDMA when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the debug request is sent when detecting a write access to the ONCE_CMD register. When the SDMA is already in debug mode, this command is simply ignored.
- rbuffer command execution-The rbuffer command exports the content of the real time buffer (RTB). If the JTAG is used, the content of the real time buffer (RTB) is captured in the SDMA data register during the capture_dr state. The register is completely shifted out after maintaining the shift_dr state during 32 TCK clock cycles. If the OnCE is driven from the ARM platform side, the content of the RTB is captured in the ONCE_DATA register after detecting a write access to the ONCE_CMD register.
- bypass command execution-This command is only available from the JTAG interface. It enables daisy-chaining of the SDMA JTAG TAP controller with other JTAG TAP controllers. This command does not change the SDMA state and can be executed in any mode (run, debug, or sleep). It selects the bypass register of the TAP controller.

43.4.13.4 Registers Descriptions

See SDMACORE, and SDMAARM, for detailed information on each register.

43.4.13.4.1 Event Cell Counter Register (ECOUNT)

The event cell counter register is a 16-bit register that contains the number of times minus one that an event detection occurs before generating a debug request.

This register should be written before attempting to use the event detection counter during an event detection process. The event cell counter register is cleared on a JTAG reset.

43.4.13.4.2 Event Cell Address Registers (EAA or EAB)

The event cell contains two address registers-the event cell address register (a), called EAA, and the event cell address register (b), called EAB. Every address register is a 16-bit register that stores a user-defined address value. This value computes one of the following address conditions: addra_cond or addrb_cond. Every address register is cleared on a JTAG reset.

43.4.13.4.3 Event Cell Address Mask Register (EAM)

The event cell address mask register is a 16-bit register that contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before comparing addresses.

NOTE

There is a common address mask value for the two address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The event cell address mask register is cleared on a JTAG reset.

43.4.13.4.4 Event Cell Data Register (ED)

The event cell data register is a 32-bit register that contains a user-defined data value. This data value is an input for the data comparator, which generates the data_cond condition.

The event cell data register is cleared on a JTAG reset.

43.4.13.4.5 Event Cell Data Mask Register (EDM)

The event cell data mask register is a 32-bit register that contains a user-defined data mask value. This mask is applied to the data value latched from the memory bus before comparing data.

Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. The event cell data mask register is cleared on a JTAG reset.

43.4.13.4.6 Real Time Buffer Register (RTB)

The real Time Buffer register is a 32-bit register that stores and retrieves run-time information without putting the SDMA in debug mode.

Refer to Real Time Buffer for more details.

43.4.13.4.7 Event Control Register (ECTL)

The event cell control register is a 16-bit register that defines cell event occurrence conditions.

The event cell control register is cleared on a JTAG reset. See also OnCE Event Detection Unit for more details.

43.4.13.4.8 Trace Buffer (TB)

The Trace Buffer register retrieves the information in the Trace Buffer.

See Trace Buffer for more details.

43.4.13.4.9 OnCE Status Register (OSTAT)

The OnCE status register is a 16-bit register that contains processor and event detection unit status. The OSTAT is a read-only register.

Refer to OnCE Status Register (SDMAARM_ONCE_STAT) for detailed description of the individual fields in the OSTAT register.

The following figure shows the OSTAT structure.

Table 43-40. OnCE Status Register (OnCE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PST[3:0]				RCV	EDR	ODR	SWB	MST					ECDR[2:0]		

Where PST[3:0] is the SDMA core state, RCV is set when the real-time buffer (RTB) is modified. EDR, ODR, and SWB are set, respectively, when the SDMA has entered debug mode because of an external debug request, a OnCE debug_rqst command, or a software breakpoint. MST is set when the OnCE is controlled from the ARM platform control interface, and when ECDR is a three-flag set that shows the event cell condition(s) that put the core in debug mode. The OSTAT never provides more than one reason for entering debug mode.

There are two ways of accessing OSTAT content, as follows:

1. Send an rstatus command to the OnCE controller through the JTAG, or read the ONCE_STAT register through the ARM platform access. Executing the rstatus command through the JTAG can be performed in both user and debug modes.
2. Perform an SDMA read access to the location in the SDMA core memory map (OSTAT register) debug mode using the exec_once command. With this method of access, the SDMA state reflected by the PST (processor status bit) is always DATA.

The register may also be accessed by a running application.

43.4.13.5 JTAG Interface Requirements

Because the signals received from the JTAG (running on TCK) are transferred to the OnCE controller (running on the SDMA clock), a synchronization mechanism is required.

43.4.13.5.1 TCK Speed Limitation

In the JTAG top-level layer, the TDO signal is always captured on a TCK falling edge. To guarantee a stable TDO signal from the SDMA during this operation, a falling edge detection is performed on TCK.

Before being latched in the *I* flip-flop (see Figure 43-11) on TCK falling edge, the TDO signal must be stable at the input of the flip-flop. This condition is verified if the TCK period is superior to the following delay:

worst-case edge detection delay + negative-edge signal propagation delay + JTAG top-level logic propagation delay

The frequency relationship, TCK < CLK/8, limitation guarantees that all operations are performed as expected.

43.4.13.5.2 Synchronization Implementation

The figure found here shows the synchronization mechanism.

Flip-flops tck0, tck1, and tck2 perform falling- and rising-edge detections on TCK. They generate the posedge_detected and negedge_detected nets that are used to sample the TDI and TMS inputs into the respective tdi and tms flip-flops, and update the tdo flip-flop to yield the TDO output. In the design, the only signal that might go metastable is the output of the tck0 flip-flop. This signal is captured in the tck1 flip-flop and no logical operation is performed on it to minimize a metastability propagation risk.

The TDI and TMS flip-flops also cannot go metastable: The propagation time of the rising-edge detection signal through tck0, tck1, and tck2 guarantees that the TDI and TMS inputs are stable when captured in the TDI and TMS flip-flops.

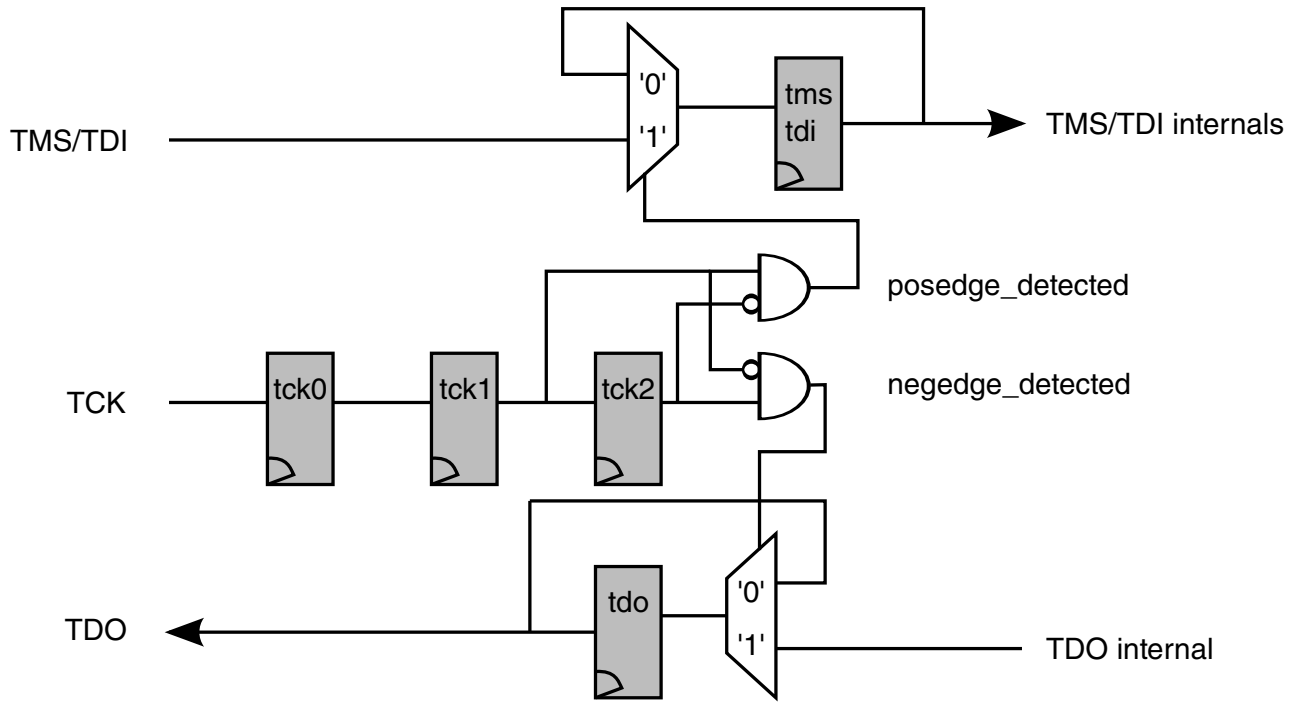


Figure 43-11. OnCE Synchronization Layer

The following figure shows synchronization timings. It takes three CLK clock cycles to synchronize TDI on the SDMA clock.

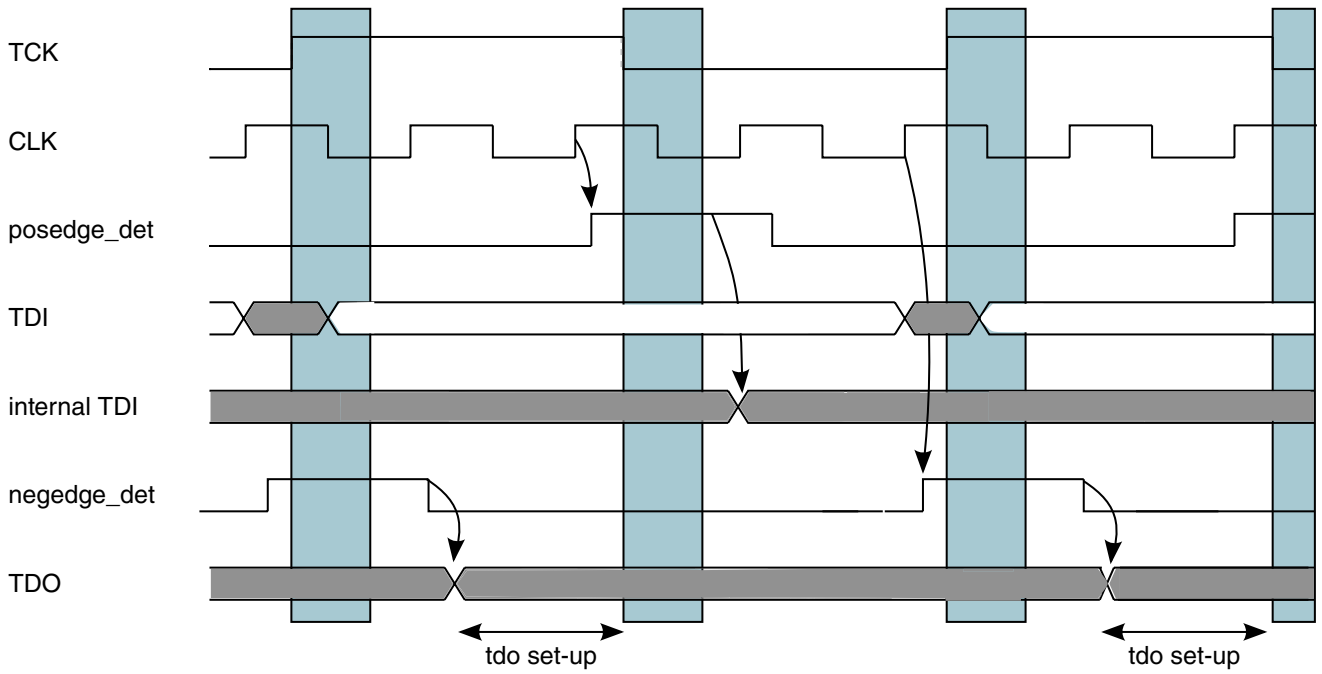


Figure 43-12. Synchronization Timings

43.4.13.5.3 JTAG Controller Start-Up Recommended Procedure

To ensure correct TAP controller initialization, it is recommended to use the following procedure:

1. Assert JTAG reset TRSTB (for example, set low).
2. Set TMS low.
3. Wait for 1 TCK clock.
4. Release JTAG reset TRSTB (for example, set high).
5. Wait for a minimum of five TCK cycles.

43.4.14 Using the OnCE

This section provides the elements necessary to run the OnCE during a debug process.

In addition to the basic set of commandsdescribed in OnCE Commands, more complex commands can be built to meet users' requirements.

43.4.14.1 Activating Clocks in Debug Mode

For power consumption issues, some clocks in the SDMA are disabled when not needed.

This is the case for instances when the SDMA is in sleep mode. Clock gating management depends on the interface used to control the OnCE.

- For the JTAG access, the SDMA clock gating must be turned off via the clk_gating_off input.
- For the ARM platform access, the SDMA clock gating is automatically turned off when the ARM platform access is enabled (see OnCE Enable (SDMAARM_ONCE_ENB)).

43.4.14.2 Getting the Current Status

Most of the commands the OnCE supports have an impact on the status of the SDMA.

It is not permissible to request the execution of an instruction on the SDMA from the OnCE while the SDMA is not in debug mode. Such a violation may cause unpredictable behavior, and it might be necessary to reset the SDMA.

Therefore, the value of the PST bits provided in the OnCE status register should always be checked before sending any request to the SDMA.

43.4.14.3 Methods of Entering Debug Mode

A debug request may be asserted at any time, but it is not always taken into account immediately. Debug mode cannot be entered in the middle of an instruction, or during the save or restore states of a context switch.

The request is ignored when the core is already in debug mode. Refer to Figure 43-4, which shows all possible transitions to the debug state, as there are several ways to enter debug mode.

43.4.14.3.1 External Debug Request During Reset

To enter debug mode after exiting reset, the external debug line has to be maintained high. This line is handled by the JTAG top-level block.

NOTE

The SDMA detects the debug requests only if the SDMA clock is running (see Activating Clocks in Debug Mode). The debug request line should be not be maintained high when the SDMA is in debug mode.

NOTE

The debug_rqst command (from the OnCE command set) is not supported during system reset.

43.4.14.3.2 Debug Request During Normal Activity

During normal activity, the SDMA enters debug mode when the following is true:

1. If the debug request line from the JTAG top-level is asserted, or
2. If the OnCE controller receives a debug_rqst command.

   The debug_rqst command can be sent by the JTAG access or by an access on the ARM platform side (if the ARM platform access is enabled).

43.4.14.3.3 Software Breakpoint Instruction

The SDMA enters debug mode at the end of the execution of a software breakpoint instruction. This instruction must be inserted in program flow executed by the core.

43.4.14.3.4 Event Detection Unit Matching Condition

If the event detection is enabled, a debug request is sent to the core after detecting a matching condition on the SDMA memory bus.

See OnCE Event Detection Unit for more details.

43.4.14.4 Executing Instructions in Debug Mode

The OnCE supports a mechanism to execute instructions in debug mode. If the SDMA is in debug mode, then the exec_once command can be used to execute an SDMA instruction from the OnCE controller. The SDMA returns to debug mode at the end of each execution.

Some instructions are not supported by the exec_once command: done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions are not supported.

NOTE

While instructions are executed in debug mode from the OnCE, the program counter of the SDMA is not incremented.

43.4.14.5 Command Sequences Examples

This section provides examples of command sequences that run the SDMA in debug mode. These sequences are available for both the ARM platform and JTAG accesses.

The following presents the syntax used in this section. The data field provided with each command is put in parenthesis with the command name. A '-' is used if the data field provided is a *don't care* value.

```
my_command(data_field);                 // executing my_command with a data field
my_command(-);                        // executing my_command with a don't care data field
```

The value returned by the command (if there is one) is referred by an assignment. In case the value returned by the command is not used, the assignment is omitted. For an ARM platform access, the value returned (it is always a data value) is obtained by reading back into the SDMA data register.

```
data_out = my_command(data_in); // returning a data value
```

To clarify the syntax, the instructions' opcodes are referred to by their names. In practice, use the corresponding 16-bit encoding.

43.4.14.5.1 Getting the SDMA Status

NOTE

Before executing any command that affects the SDMA (like dmov or exec_once), check that the SDMA is in debug mode.

Use the following snippet:

```
rstatus();        // read SDMA status until the SDMA is in debug mode
...
rstatus();
```

If the SDMA is not in debug mode, then a debug request must be generated. In this case, the SDMA enters debug mode at the end of the execution of the current instruction. Use this snippet:

```
debug_rqst(-);   // debug request
```

In the following sections, it is assumed that the SDMA was successfully put into debug mode.

43.4.14.5.2 Saving the Context

The first debug task is to save the SDMA context, which is the content of the eight general-purpose registers, the loop and PC-related registers, and the flags.

Use the general register GReg[1] as an intermediate register to export the entire context of the SDMA.

The following example shows how to save GReg[0], GReg[1], GReg[2] and GReg[3]. The sequence of commands used to export additional general registers is very similar to this.

Save GReg[0], GReg[1], GReg[2], and GReg[3]

```
GReg1_data = dmov(-);                          // the value exported is the content of
GReg[1]
exec_once("mov GReg1,GReg0");                      // puts the content of GReg[0] into
GReg[1]
GReg0_data = dmov(-);                          // the value exported is the content of
GReg[0]
exec_once("mov GReg1, GReg2");                     // puts the content of GReg[2] into
GReg[1]
GReg2_data = dmov(-);                          // the value exported is the content of
GReg[2]
exec_once("mov GReg1, GReg3");                     // puts the content of GReg[3] into
GReg[1]
GReg3_data = dmov(-);                          // the value exported is the content of
GReg[3]
```

Get the value of the internal flags (SF, DF, T, and LM), of the loop related registers (EPC and SPC), and of the PC-related registers (PC and RPC). Use a done 5, which is the formatting instruction dedicated to the debug. This instruction formats the flags and the values contained in the registers. It also writes the resulting values into the channel context memory. It should not be used when entering debug from the IDLE state (for example, with no active channel script running on the SDMA), because it will update a channel context that may belong to any channel.

```
exec_once("done 5");                   // formatting the value of flags and registers
```

At this point, the channel context should be up-to-date in memory, and debug operations should now be possible. However, the context can be exported with the following instructions:

Exporting the Context

```
dmov(ctx_base_addr);                             // loading GReg[1] with the channel
context base address
exec_once("ld GReg0,(GReg1,0)");                     // get RPC-PC into GReg0
exec_once("ld GReg1, (GReg1,1)");                    // get SPC-EPC into GReg1
Loop_data = dmov(-);                           // read back the value of Loop registers
exec_once("mov GReg1, GReg0");                     // puts the PC info into GReg1
PC_data = dmov(-);                             // reads back the content of the PC registers
```

After this sequence of operations, the entire SDMA context is exported via the OnCE.

43.4.14.5.3 Restoring the Context

At this point in the operation, restore the context of the SDMA. It can be different from the original context located in memory, and the content previously saved into the debugging application via the OnCE.

The example found hereshows how it is possible to modify the current channel context.

Modifying the Current Channel Context

```
dmov(Loop_data);                                  // put Loop former value into GReg[1]
exec_once("mov GReg0, GReg1");                        // copy to GReg[0]
dmov(PC_data);                                    // put PC former value into GReg[1]
exec_once("mov GReg2, GReg1");                        // copy to GReg[2]
dmov(ctx_base_addr);                                 // put channel context base address into
GReg[1]
exec_once("st GReg0, (GReg1,1)");                          // restore Loop context
exec_once("st GReg2, (GReg1,0)");                          // restore PC context
```

Once the context in memory is the desired context (with or without applying the previous instruction sequence), it can be restored to the *real* PC and loop registers in the SDMA core:

```
exec_once("cpShReg");                    // restore flags and PC & loop related registers
```

After this command, the SDMA core PC, RPC, SPC, EPC registers, as well as the flags contain the same data as what is stored in the context RAM for the current channel.

The following example shows how to restore the context of general registers GReg[0], GReg[1], GReg[2] and GReg[3].

Restoring the General Register Context

```
dmov(GReg3_data);                     // put GReg[3] restore value in GReg[1]
exec_once("mov GReg3, GReg1"); // restore GReg[3]
dmov(GReg2_data);                     // put GReg[2] restore value in GReg[1]
exec_once("mov GReg2, GReg1"); // restore GReg[2]
dmov(GReg0_data);                     // put GReg[0] restore value in GReg[1]
exec_once("mov GReg0, GReg1"); // restore GReg[0]
dmov(GReg1_data);                     // restore GReg[1]
```

At this point, it is possible to restart the normal program execution.

NOTE

Every SDMA core general register value can be modified by a mov instruction, which makes modification of these registers easy during debug. Unfortunately, there is no such instruction as a mov to directly modify the contents of either PCU register or flag (PC, RPC, SPC, EPC, T, LM, SF, or DF). The cpShReg instruction is meant to provide a means for changing these register contents via the context memory.

43.4.14.5.4 Accessing the Memory

In the example shown here, it is assumed that the SDMA context is entirely saved. If true, it is permissible to modify the general purpose registers during debugging activity.

To perform a memory read access, the target address is stored via the OnCE in GReg[1], then the load instruction is executed on the SDMA (the data loaded from the memory overwrites the address contained in GReg[1]), and then the result value is read back via the OnCE.

```
macro READ:            dmov(target_addr);                                  // put the target
address in GReg[1]
                    exec_once("ld GReg1,(GReg1,0)");                        // execute the
load instruction
                    res_data = dmov(-);                                 // exports the result
data value
```

For a memory write access, the target address is written in GReg[0], and the value to store is written in GReg[1]. Then the store instruction is executed on the SDMA.

```
macro WRITE:            dmov(target_addr);                                  // puts the
target address in GReg[1]
                    exec_once("mov GReg0,GReg1");                       // puts the target
address in GReg[0]
                    dmov(target_data);                                  // puts the target
data in GReg[1]
                    exec_once("st GReg1,(GReg0,0)");                        // performs the
store operation
```

This sequence is shown as an example; however, many other sequences are possible.

NOTE

This sequence of commands can also be applied to memory-mapped registers.

43.4.14.5.5 Resuming Program Execution

Before resuming program execution, it is assumed that the SDMA context is properly restored. There are two ways to restart the SDMA.

Start by executing the last instruction fetched before entering debug mode, as follows.

```
run_core(-);                          // resume execution from where we stopped before
```

If necessary, restart the execution from a different address. In this case, use the exec_core command. The data field provided with this command must be the encoding of a jump instruction.

```
exec_core("jmp start_addr"); // rerun the SDMA from another address
```

In these two examples, the SDMA exits debug mode and keeps executing the code fetched from the memory.

43.4.14.5.6 Single Stepping in RAM

To execute a program step-by-step from the RAM, insert software breakpoints in the program flow at appropriate places so that the SDMA only executes one instruction before returning to debug mode.

First, read the next instruction to execute in the RAM. Then, depending on the value of this instruction, compute the address where a software breakpoint instruction should be inserted. The instruction at the corresponding address must be saved, and, the software breakpoint instruction is inserted. After restarting the SDMA, there is only one instruction executed before meeting the software breakpoint.

The following example shows the macro functions READ and WRITE, which correspond to the sequence of commands (described above) used to access the memory.

NOTE

The data read from the memory are 32-bit values, while the instructions are 16-bit values only. This is why it is best to only use addresses divided by two when accessing the memory.

READ and WRITE Macro Functions

```
next_instr = READ(run_addr/2);                          // read the next instruction to execute
// the tool now has to compute the address where the breakpoint
// instruction should be inserted, this address is the "bkpt_addr"
instr_save = READ(bkpt_addr/2);                          // save the instruction before
overwriting
STORE("bkpt instruction",bkpt_addr/2);                        // store the bkpt instruction
in memory
exec_core("jmp run_addr");                              // rerun the SDMA
rstatus(-);                                           // wait for the SDMA to enter debug mode
...
rstatus(-);
STORE(instr_save,bpkt_addr/2);                           // restore the instruction
overwritten
```

In case of branched conditional instructions, a breakpoint instruction should be written at the two possible target addresses.

43.4.14.5.7 Single Stepping in ROM

No single-step mechanism is supported in ROM. The program code can be loaded in the RAM, where the single-step mechanism can be executed.

43.4.14.6 OnCE Event Detection Unit

The event detection unit watches signals from the data memory bus (DMBUS), which the SDMA core uses to access its RAM, ROM, and memory mapped registers.

A debug request is sent to the OnCE controller when user-defined conditions on address and/or data values are true.

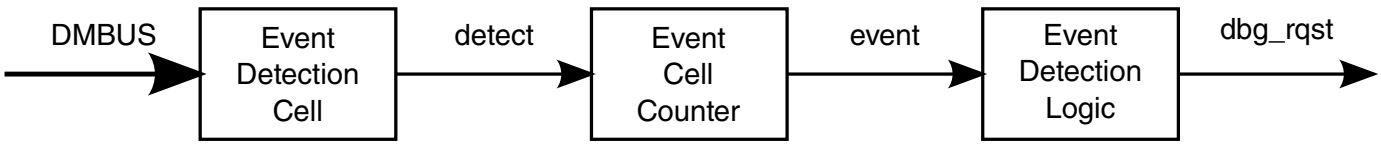


Figure 43-13. Event Detection Unit

A counter, provided with the detection cell, is decreased after an event detection. A debug request is sent to the core only when the counter reaches the value of 0. It is possible to disable the use of the counter if a debug request has to be generated after each event detection.

The event cell is the basic block that supports hardware breakpoints on an address value and/or data values coming from the SDMA memory bus. The trigger condition that generates the debug request is a mixed condition based on those values.

The following figure shows the event cell architecture. The event cell contains the address (stored in the memory address register) and the data (stored in the memory data register) used during the last memory access. There are some user-defined reference values located in memory mapped registers-the event cell addresses, the event cell address mask, the event cell data, and the event cell data mask. These registers are accessed by standard load/store instructions just like regular memory locations.

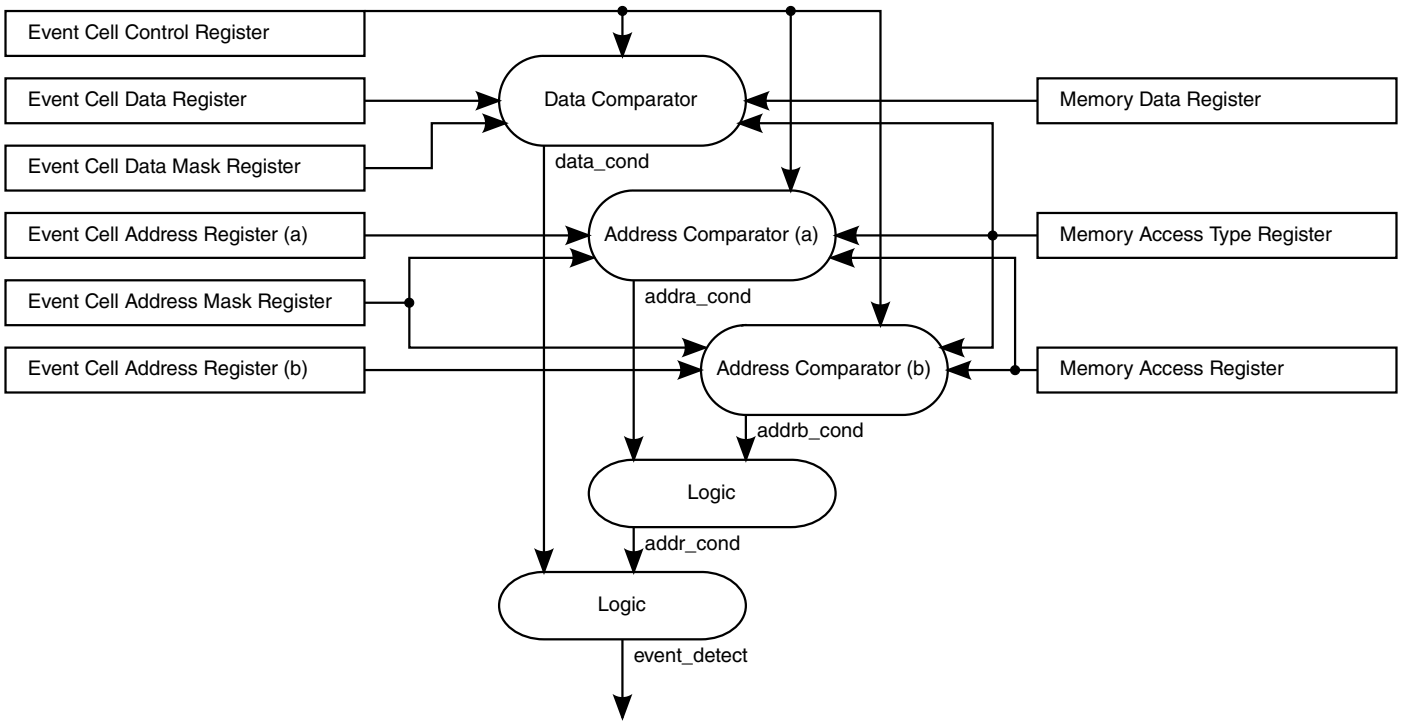


Figure 43-14. Event Cell Architecture

To define a memory breakpoint, three conditions are taken into account: The first two conditions are comparisons of the current memory address with user-defined reference addresses (these conditions are called addressA and addressB). The third condition consists of a comparison between the data received on the DMBUS and a user-defined reference data (this condition is called data). An intermediate address condition is set to express a dependency between addressA and addressB conditions.

43.4.14.7 Clock Gating and Reset

This section details how to use the clocks and handle the reset signals.

43.4.14.7.1 Clocks

Because the SDMA uses clock gating to save power, it is necessary to disable the clock gating and force the clocks to be enabled when using the OnCE.

When the OnCE is accessed through its JTAG interface, clock gating must be disabled outside the SDMA via a dedicated SDMA input port clk_gating_off. The reason why detection is not performed automatically by the SDMA internal hardware is that it would cost power to monitor activity on the JTAG interface.

When the OnCE is accessed through the ARM platform Control interface, clock gating is automatically turned off. This is done when bit 0 of the ONCE_ENB register (see OnCE Enable (SDMAARM_ONCE_ENB)) is set. A write access to this register is possible even when the OnCE clock is not running. If the ARM platform access is used, the bit in the ONCE_ENB register must be set before any attempt to access any other OnCE register.

43.4.14.7.2 Resets

The OnCE reset is different from the SDMA main reset.

Normally, activating the SDMA reset while keeping the OnCE reset inactive (when possible) enables you to reset the core without having to reprogram the OnCE.

43.4.14.8 Real Time Features

To rebuild the skeleton of a program execution, it is necessary to store the addresses of the program instructions where jumps are taken: A trace buffer is therefore provided. A real time buffer has also been added to receive data values written during a program execution.

The content of this register may be exported through JTAG ports without stopping the core.

43.4.14.8.1 Trace Buffer

The Trace Buffer is a 32-stage buffer that contains appropriate information to identify the 32 last changes of flow detected during a program execution.

The following figure shows an overview of the Trace Buffer.

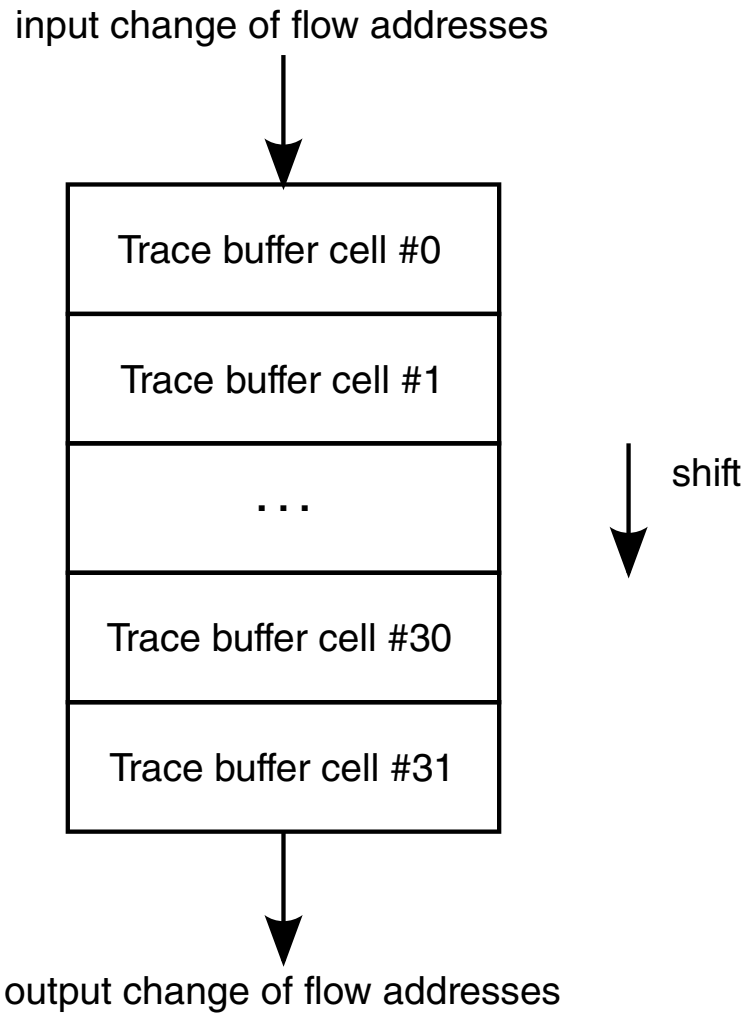


Figure 43-15. Trace Buffer

Each cell of the trace buffer contains two reference addresses and a flag. The flag is set when the addresses stored in the cell correspond to a valid change of flow; otherwise, the flag is cleared. The three most significant bits are unused.

After every change of flow detection, the address of current instruction and the address of the target instruction are stored at the top of the Trace Buffer (cell #0). The flag in the cell is set to indicate that a valid change of flow was detected. Former cell values are shifted one level down. The Trace Buffer contains the 32 last changes of flow. All the flags are reset on a software or a hardware reset, and after each transition from debug mode to user mode.

A memory mapped register of SDMA core, the Trace Buffer register (TB), is provided to read the content of the Trace Buffer. This operation should be done in debug mode. Performing a read access to the Trace Buffer register returns the content of the bottom of the Trace Buffer (cell #31). After every read access, the trace buffer is shifted one level down, and the flag at the top of the trace buffer is cleared.

A typical OnCE command sequence that retrieves the oldest change-of-flow information is a follows:

```
exec_once("mov r1, TB");                              // stores the oldest change-of-flow in
GReg1
dmov(-);                                          // retrieves GReg1 contents
```

This sequence requires the SDMA to be put in debug mode.

43.4.14.8.2 Real Time Buffer

The Real Time Buffer register (RTB) is a memory mapped register that can be accessed as a regular memory location by the SDMA core during program execution. This register is located in the OnCE.

Executing ar rbuffer command (see The OnCE Controller for further details) exports the content of this register through JTAG ports.

When a write access is performed at the memory location corresponding to the RTB, the receive flag (for example, the RCV bit) is set in the OnCE Status Register (OSR). This flag is cleared at the end of the execution of a rbuffer command.

NOTE

Every write access to the RTB memory location updates the RTB register even if the RCV flag is set. The RTB is cleared on a JTAG reset.

43.4.14.8.3 Emulation Pin

The debug_matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit.

Since it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

43.4.14.8.4 Real-Time Debug Outputs

The table found here shows the debug signals that are available at the SDMA boundaries. Their availability at chip boundaries depends on the project.

Table 43-41. Real-Time Debug Output Pins

Pin	Description
debug_core_state[3:0]	The core_state bits reflect the state of the SDMA core. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the core idle state); the "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 is executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: they are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Context Switch Saving Channel 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change of Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Context Switch Restoring Channel
debug_yield	Pulse that is active when a yield (done 0) or a yieldge (done 1) instruction is executed. 0 - 1 yield/yieldge executed
debug_core_run	Active when the SDMA core is executing instructions. 0 Debug or sleep mode

Table continues on the next page...

Table 43-41. Real-Time Debug Output Pins (continued)

Pin	Description
	1 Run mode
debug_event_channel_sel	Indicates if debug_event_channel displays current channel or last received event 0- debug_event_channel[5:0] gives the number of the current channel 1- debug_event_channel[5:0] gives the number of the last received event
debug_event_channel[5:0]	Gives the number of any DMA request as soon as it is received or the number of the current channel. The value of debug_event_channel_sel indicates if debug_event_channel displays the current channel or last received event. The signal debug_event_channel_sel must be observed to determine what information is provided on debug_event_chanel at any given time.
debug_pc[13:0]	Program Counter value; it has a meaning when the core is in run mode.
debug_mode	Set when the core is in debug. 0 - 1 Core is in debug
debug_bus_error	Set when an error was received during a load or a store (ld, st, ldf, or stf instruction) and registered in SF or DF flag. 0 No error during last load/store 1 Error during last load/store
debug_bus_device[4:0]	Indicates the device or functional unit that is accessed by the current instruction. The debug_bus_device output is always valid when in sleep mode, debug mode, or executing any instruction that does not access the functional units or the memory mapped devices, "no access" is output. 0 No access 1 MSA 2 MDA 3 MD 4 MS 5 PSA 6 PDA 7 PD 8 PS 9 RESERVED 10 RESERVED 11 RESERVED 12 RESERVED 13 CA 14 CS 15 Reserved 16 Memory (RAM or ROM) 17 Memory mapped register

Table continues on the next page...

Table 43-41. Real-Time Debug Output Pins (continued)

Pin	Description
	18 Peripheral #1 19 Peripheral #2 20 Peripheral #3 21 Peripheral #4 22 Peripheral #5 23 Peripheral #6 24 Peripheral #7 25 Peripheral #8 26 Peripheral #9 27 Peripheral #10 28 Peripheral #11 29 Peripheral #12 30 Peripheral #13 31 Peripheral #14
debug_bus_rwb	Indicates the direction of the access given by debug_bus_device 0 Write access (st or stf) 1 Read access (ld or ldf)
debug_matched_dmbus	Pulse indicating the OnCE event detection unit has detected a match on the data bus during an access to memory (RAM or ROM), a memory mapped register or a peripheral that is hooked to the SDMA. 0 - 1 data bus match detected
debug_rtbuffer_write	Pulse indicating when the real-time buffer is written by the core. 0 - 1 RTB was modified
debug_evt_chn_lines[7:0]	Eight lines that generate short pulses when DMA requests are received or channels are (re)started. Every line is controlled through two parameters defined in registers Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1) (as described in SDMAARM). The following two parameters are available for every line: • CNF-Indicates what is monitored on the line: 0 for a channel start, 1 for a DMA request reception • NUM[5:0]-Gives the number of the DMA request or channel to monitor

The matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Because it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

All real-time debug outputs are disabled by default (for example, they are stuck to 0) to avoid power consumption when they are not used. They are enabled when bit 11 (RTDOBS) of the Configuration Register (SDMAARM_CONFIG) is set. Signals provided to the system JTAG controller for SDMA debug mode status will also be enabled when the *clk_gating_off* input is asserted.

43.5 Instruction Set

43.5.1 Instruction Encoding

This section presents a short summary of the instruction codes. All context switch instructions are listed for information only; they cannot function properly out of the context switch routine.

```
x...x - don't care

rrr - destination/source general register

sss - additional source general register

bbb - general register used as address base register

ddddd - address displacement

nnnnn - bit number
uuuuuuuu - function unit command bits

pppppppp - branch displacement (signed)

iiiiiiii - 8-bit immediate

jjj - control bit to clear

ff - flag to clear
00000jjj00000000        - done (done,yield,wait)
00000jjj00000001        - notify
00000xxx00000010        - reserved
00000xxx00000011        - reserved
00000xxx00000100        - reserved
0000000000000101     - softBkpt
0000000100000101     - reserved
0000001000000101     - reserved
0000001100000101     - reserved
0000010000000101     - reserved
0000010100000101     - reserved
0000011000000101     - reserved
0000011100000101     - reserved
0000000000000110     - ret
0000000100000110     - reserved
0000001000000110     - reserved
0000001100000110     - reserved
0000010000000110     - reserved
0000010100000110     - reserved
0000011000000110     - reserved
```

```
0000011100000110     - reserved
000000ff00000111       - clrf ff
0000010000000111     - reserved
0000010100000111     - reserved
0000011000000111     - reserved
0000011100000111     - illegal
00000rrr00001000        - jmpr r
00000rrr00001001        - jsrr
00000rrr00001010        - ldrpc r
00000rrr00001011        - reserved
00000rrr000011xx          - reserved
00000rrr00010000        - revb
00000rrr00010001        - revblo
00000rrr00010010        - rorb
00000rrr00010011        - reserved
00000rrr00010100        - ror1
00000rrr00010101        - lsr1
00000rrr00010110        - asr1
00000rrr00010111        - lsl1
00000rrr001nnnnn        - bclri r,n
00000rrr010nnnnn        - bseti r,n
00000rrr011nnnnn        - btsti r,n
00000xxx10000xxx           - reserved
00000rrr10001sss           - mov
00000rrr10010sss           - xor
00000rrr10011sss           - add
00000rrr10100sss           - sub
00000rrr10101sss           - or
00000rrr10110sss           - andn
00000rrr10111sss           - and
00000rrr11000sss           - tst
00000rrr11001sss           - cmpeq
00000rrr11010sss           - cmplt
00000rrr11011sss           - cmphs
0000011011100000     - reserved
0000011011100001     - reserved
0000011011100010     - cpShReg
0000011011100011     - reserved
0000011011100100     - reserved
0000011011100101     - reserved
0000011011100110     - reserved
0000011011100111     - reserved
00000xxx11101xxx           - reserved
00000xxx11110xxx           - reserved
00000xxx11111xxx           - reserved
00001rrriiiiiiii               - ldi r,i
00010rrriiiiiiii               - xori r,i
00011rrriiiiiiii               - addi r,i
00100rrriiiiiiii               - subi r,i
00101rrriiiiiiii               - ori r,i
00110rrriiiiiiii               - andni r,i
00111rrriiiiiiii               - andi r,i
01000rrriiiiiiii               - tsti r,i
01001rrriiiiiiii               - cmpeqi r,i
01010rrrdddddbbb         - ld r,(d,b)
01011rrrdddddbbb         - st r,u
01100rrruuuuuuuu         - ldf r,u
01101rrruuuuuuuu         - stf r,u
011100xxxxxxxxxx               - reserved
011101xxxxxxxxxx               - reserved
011110ffnnnnnnnn       - Loop ff flags are reset
01111100pppppppp     - bf pc=pc+signed(pppppppp)+1
01111101pppppppp     - bt pc=pc+signed(pppppppp)+1
01111110pppppppp     - bsf pc=pc+signed(pppppppp)+1
01111111pppppppp     - bdf pc=pc+signed(pppppppp)+1
10aaaaaaaaaaaaaa     - jmp absolute
11aaaaaaaaaaaaaa     - jsr absolute
```

43.5.2 SDMA Instruction Set

This section describes all the useful instructions from the SDMA set.

Table 43-42. SDMA Instruction List

Instruction	Description	Page
ADD	Addition	ADD (Addition)
ADDI	Add with Immediate Value	ADDI (Add with Immediate Value)
AND	Logical AND	AND (Logical AND)
ANDI	Logical AND with Immediate Value	ANDI (Logical AND with Immediate Value)
ANDN	Logical AND NOT	ANDN (Logical AND NOT)
ANDNI	Logical AND with Negated Immediate Value	ANDNI (Logical AND with Negated Immediate Value)
ASR1	Arithmetic Shift Right by 1 Bit	ASR1 (Arithmetic Shift Right by 1 Bit)
BCLRI	Bit Clear Immediate	BCLRI1 (Bit Clear Immediate)
BDF	Conditional Branch if Destination Fault	BDF (Conditional Branch if Destination Fault)
BF	Conditional Branch if False	Functional Units Programming Model
BSETI	Bit Set Immediate	BSETI (Bit Set Immediate)
BSF	Conditional Branch if Source Fault	BSF (Conditional Branch if Source Fault)
BT	Conditional Branch if True	BT (Conditional Branch if True)
BTSTI	Bit Test immediate	BTSTI (Bit Test immediate)
CLRF	Clear ARM platform flags	CLRF (Clear ARM platform flags)
CMPEQ	Compare for Equal	CMPEQ (Compare for Equal)
CMPEQI	Compare with Immediate for Equal	CMPEQI (Compare with Immediate for Equal)
CMPHS	Compare for Higher or Same	CMPHS (Compare for Higher or Same)
CMPLT	Compare for Less Than	CMPLT (Compare for Less Than)
cpShReg	Update Context of PCU Registers and Flags	cpShReg (Update Context of PCU Registers and Flag)
DONE	DONE, Yield	DONE (DONE, Yield)
ILLEGAL	ILLEGAL Instruction	ILLEGAL (ILLEGAL Instruction)
JMP	Unconditional Jump Immediate	JMP (Unconditional Jump Immediate)
JMPR	Unconditional Jump	JMPR (Unconditional Jump)
JSR	Unconditional Jump to Subroutine Immediate	JSR (Unconditional Jump to Subroutine Immediate)
JSRR	Unconditional Jump to Subroutine	JSRR (Unconditional Jump to Subroutine)
LD	Load Register	LD (Load Register)
LDF	Load Register from Functional Unit	LDF (Load Register from Functional Unit)

Table continues on the next page...

Table 43-42. SDMA Instruction List (continued)

Instruction	Description	Page
LDI	Load Register with Immediate Value	LDI (Load Register with Immediate Value)
LDRPC	Load from RPC to Register	LDRPC (Load from RPC to Register)
LOOP	Hardware Loop	LOOP (Hardware Loop)
LSL1	Logical Shift Left by 1 Bit	LSL1 (Logical Shift Left by 1 Bit)
LSR1	Logical Shift Right by 1 Bit	LSR1 (Logical Shift Right by 1 Bit)
MOV	Logical Move	MOV (Logical Move)
NOTIFY	Notify to ARM platform	NOTIFY (Notify to ARM platform)
OR	Logical OR	OR (Logical OR)
ORI	Logical OR with Immediate Value	ORI (Logical OR with Immediate Value)
RET	Return from Subroutine	RET (Return from Subroutine)
REVB	Reverse Byte Order	REVB (Reverse Byte Order)
REVBLO	Reverse Low Order Bytes	Reverse Low Order Bytes(REVBLO)
ROR1	Rotate Right by 1 Bit	ROR1 (Rotate Right by 1 Bit)
RORB	Rotate Right by 1 Byte	RORB (Rotate Right by 1 Byte)
SOFTBKPT	Software Breakpoint	SOFTBKPT (Software Breakpoint)
ST	Store Register	ST (Store Register)
STF	Store Register in Functional Unit	STF (Store Register in Functional Unit)
SUB	Subtract	SUB (Subtract)
SUBI	Subtract with Immediate	SUBI (Subtract with Immediate)
TST	Test with Zero	TST (Test with Zero)
TSTI	Test Immediate	TSTI (Test Immediate)
XOR	Logical Exclusive OR	XOR (Logical Exclusive OR)
XORI	Exclusive OR with Immediate	XORI (Exclusive OR with Immediate)

43.5.2.1 ADD (Addition)

Operation:

```
GReg[r] ← GReg[s] + GReg[r]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: add r,s

Example: add 0,3
```

ADD GReg[3] and GReg[0] and store the result in GReg[0]

CPU Flags: T

Cycles: 1

Description: Performs the ADDition of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*. The T flag is set if the result of the operation is 0. It is cleared if the result is not 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.2 ADDI (Add with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] + immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: addi r,immediate

Example: add 6,112
```

ADD GReg[6] and decimal value 112 and store the result in GReg[6]

CPU Flags: T

Cycles: 1

Description: Adds a 0-extended immediate value to a general register; stores the result in the general register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

 ...

11111110 - 254

11111111 - 255
```

43.5.2.3 AND (Logical AND)

Operation:

```
GReg[r] ← GReg[s] & GReg[r]
```

Assembler:

```
Syntax: and r,s

Example: and 1,2
```

AND GReg[1] and GReg[2] and store the result in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.4 ANDI (Logical AND with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] & immediate
```

Assembler:

```
Syntax: andi r,immediate

Example: andi 7,45
```

AND GReg[7] and decimal value 45 and store the result in GReg[7]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between a 0-extended immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

43.5.2.5 ANDN (Logical AND NOT)

Operation:

```
GReg[r] ← ~GReg[s] & GReg[r]
```

Assembler:

```
Syntax:andn r,s

Example: andn 3,4
```

AND GReg[3] and NOT GReg[4] (bit inverted) and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the negation of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format:

Table 43-43. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	0	s	s	s

Instruction Fields:

rrr /sss - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.6 ANDNI (Logical AND with Negated Immediate Value)

Operation:

```
GReg[r] ← GReg[r] & ~immediate
```

Assembler:

```
Syntax: andni r,immediate

Example: andni 0,2
```

AND GReg[0] and decimal value -3 (inverted 32-bit value 2) and store the result in GReg[0]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between the negation of a 0-extended 8-bit immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

43.5.2.7 ASR1 (Arithmetic Shift Right by 1 Bit)

Operation:

```
GReg[r]:{b31,b30,...,b1,b0} ← GReg[r]:{b31,b31,b30,...,b1}
```

Assembler:

```
Syntax: asr1 r

Example: asr1 3
```

divide by 2 the signed value of GReg[3] and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any general register to the right and keep the same sign: The left bit (bit 31) is kept untouched.

Instruction Format:

Table 43-44. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.8 BCLRI1 (Bit Clear Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),0,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bclri r,i

Example: bclri 1,12
```

clear bit 12 in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Clear the bit of register r specified by the 5-bit immediate field

Instruction Format

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

Table continues on the next page...

0	0	0	0	0	r	r	r	0	0	1	i	i	i	i	i

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - immediate value:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

43.5.2.9 BDF (Conditional Branch if Destination Fault)

Operation:

```
if (DF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax:bdf label

Example: bdf LLL
```

Jump to LLL if DF is set, or go to the next instruction if DF is cleared; the displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: If flag DF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag DF is cleared, no jump is performed: The next instruction is located at the next PC address.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	p	p	p	p	p	p	p	p

Instruction Fields:

ppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

43.5.2.10 BF (Conditional Branch if False)

Operation:

```
if (T == 0)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler:

```
Syntax: bf label

Example: bf LLL
```

Jump to LLL if T is cleared, or go to the next instruction if T is set. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is cleared, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is set, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	0	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

43.5.2.11 BSETI (Bit Set Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),1,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bseti r,i

Example: bseti 6,5
```

Set bit 5 in GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Sets bit number i in the selected General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	0	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

43.5.2.12 BSF (Conditional Branch if Source Fault)

Operation:

```
if (SF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax: bsf label

Example: bsf LLL
```

Jump to LLL if SF is set, or go to the next instruction if SF is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag SF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag SF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

43.5.2.13 BT (Conditional Branch if True)

Operation

```
if (T == 1)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler

```
Syntax: bt label

bt LLL
```

Jump to LLL if T is set, or go to the next instruction if T is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	1	p	p	p	p	p	p	p	p

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110- 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 -  (-2)

11111111 - (-1)
```

43.5.2.14 BTSTI (Bit Test immediate)

Operation:

```
T ← GReg[r]:b(i)
```

Assembler:

```
Syntax: btsti r,i

Example: btsti 2,29
```

Test bit 29 in GReg[2] and copy its value in flag T

CPU flags: T

Cycles: 1

Description: T is loaded with the value of bit number i from the selected general register.

Instruction Format:

Table 43-45. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	1	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
0000 - 0

0001 - 1

...

11110 - 30

11111 - 31
```

43.5.2.15 CLRF (Clear ARM platform flags)

Operation:

```
 if (ff%2 == 0)

SF ← 0

if (ff/2 == 0)

DF ← 0
```

Assembler:

```
Syntax: clrf ff

Example: clrf 2
```

Clear flag SF and keep flag DF unchanged

CPU Flags: SF, DF

Cycles: 1

Description: Clears a selection of the ARM platform fault flags: SF, DF, both SF and DF or none can be cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	f	f	0	0	0	0	0	1	1	1

Instruction Fields:

ff - flags field:

```
00 - clear SF and clear DF

01 - clear DF

10 - clear

SF 11 - no clear
```

43.5.2.16 CMPEQ (Compare for Equal)

Operation:

```
T ← (GReg[s] == GReg[r])
```

Assembler:

```
Syntax: cmpeq r,s

Example: cmpeq 7,5
```

Compare GReg[7] and GReg[5] and set flag T if they are equal

CPU flags: T

Cycles: 1

Description: Subtracts the destination general register *r* from the source general register *s*, and sets T if the result is 0, clears T if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.17 CMPEQI (Compare with Immediate for Equal)

Operation:

```
T ← (GReg[r] == immediate)
```

Assembler:

```
Syntax: cmpeqi r,immediate

Example: cmpeqi 2,13
```

Compare GReg[2] and decimal value 13 and set flag T if they are equal

CPU Flags: T

Cycles: 1

Description: Subtracts the 0-extended 8-bit immediate value from the general register, and sets T if the result is 0, clears T if the result is not 0. The immediate value is the low-order byte of the instruction.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]
```

```
001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

43.5.2.18 CMPHS (Compare for Higher or Same)

Operation:

```
T ← (GReg[r] ≥ GReg[s])
```

Assembler:

```
Syntax: cmphs r,s

Example: cmphs 0,1
```

Compare GReg[0] and GReg[1] and set flag T if GReg[0] is higher than or equal to GReg[1]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is higher than or equal to the source general register *s*, clears T otherwise. The comparison is unsigned.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.19 CMPLT (Compare for Less Than)

Operation:

```
T ← (GReg[r] < GReg[s])
```

Assembler:

```
Syntax: cmplt r,s

Example: cmplt 7,4
```

Compare GReg[7] and GReg[4] and set flag T if GReg[7] is lower than GReg[4]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is lower than the source general register *s*, clears T otherwise. The comparison is signed.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	0	s	s	s

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]
```

```
011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.20 cpShReg (Update Context of PCU Registers and Flag)

Assembler:

```
Syntax: cpShReg
```

CPU Flags: none

Cycles: 1

Description: SF, RPC, T, PC,LM, EPC, DF, and SPC registers are updated according to the value of their corresponding bits in the context memory. This instruction must only be used in debug mode via the OnCE. It reverses the done *5* operation.

Instruction Format

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

43.5.2.21 DONE (DONE, Yield)

Operation:

```
if (jjj&6 == 2) HE[CCR] ← 0

if (jjj == 3) HI[CCR] ← 1

if (jjj == 4) EP[CCR] ← 0


if ((jjj == 0) && (NCP > CCP)) CCR ← NCR

else if ((jjj == 1) && (NCP >= CCP))

CCR ← NCR

else

CCR ← NCR
```

(CCR stands for Current Channel Register; NCR stands for Next Channel Register)

Assembler:

```
Syntax: done jjj

Example: done 3
```

Clear HE bit for the current channel, send an interrupt to the ARM platform for the current channel and reschedule.

CPU Flags: Unaffected

Cycles: Variable if a context switch is done, 1 otherwise

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required. Sends an interrupt to the corresponding ARM platform by setting the appropriate flag, if required (HI for the corresponding channel number). Reschedules according to the mode and the NCP (Next Channel Priority) and CCP (Current Channel Priority) values. According to the scheduling decision, the NCR (Next Channel Register) is copied to the CCR (Current Channel Register) and channel contexts are switched. If several channels with the same highest priority are pending, they are ordered by their number from 31 down to 0. The higher number is selected (for example, channel 26 is selected if channels 3, 12, 14, and 26 with the same highest priority are pending). If no flag is modified, the reschedule can allow the replacement of the current channel by another channel with a priority strictly greater than the current channel priority (yield). Or, it can allow the replacement of the current channel by another channel with a priority greater than or equal to the current channel priority (yieldge). In the latter case, the selected channel will always be the first one with the same priority, starting from channel number 31 down to channel 0 (the current channel does not belong to the set of selectable channels).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	0

jjj - Channel Flags field:

000 - No channel flags affected: Reschedule only if the next channel priority is greater than current channel priority (yield)

001 - No channel flags affected: Reschedule only if the next channel priority is greater than or equal to the current channel priority (yieldge)

010 - Clear HE for the current channel and reschedule 011 - Clear HE, set HI for the current channel and reschedule 100 - Clear EP for the current channel and reschedule

101 - Reserved for debug to copy relevant registers into context memory

110 - RESERVED

111 - RESERVED

For the scheduling rules, refer to Scheduler Functional Description. Every possible done instruction is further described as follows:

- done 0/yield is executed by a channel script when it accepts preemption by a higher priority channel;
- done 1/yieldge is executed by a channel script when it accepts preemption by a higher priority channel and it also accepts a roll-up with other channels that have the same priority;
- done 2 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed and it requires termination;
- done 3 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed, it requires termination and it needs to trigger an interrupt to the ARM platform upon closure;
- done 4 is executed by a channel script that was triggered by a DMA request, when its task is completed and it requires termination;
- done 5 is used in debug mode only; it copies the PCU registers and flags to the context memory of the current channel;

43.5.2.22 ILLEGAL (ILLEGAL Instruction)

Operation:

```
PC ← 0001
```

Assembler:

```
Syntax: illegal
```

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the Illegal instruction routine located at address 0001. All unauthorized instructions result in an Illegal instruction behavior; however, the ILLEGAL instruction must be used to guarantee software compatibility with future versions of the SDMA.

Instruction Format

Table 43-46. Instruction Format

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

43.5.2.23 JMP (Unconditional Jump Immediate)

Operation:

```
PC ← absolute_address
```

Assembler:

```
Syntax: jmp label

Example: jmp LLL
```

The assembler translates the label to the exact address

CPU Flags:Unaffected

Cycles: 2

Description: Jumps to the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaaa - address field:

```
00000000000000 - 0

00000000000001 - 1

...

11111111111110 - 16382

11111111111111 - 16383
```

43.5.2.24 JMPR (Unconditional Jump)

Operation:

```
PC ← GReg[r]
```

Assembler:

```
Syntax: jmpr r

Example: jmpr 0
```

Jump to address stored in GReg[0]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the absolute address contained in a General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.25 JSR (Unconditional Jump to Subroutine Immediate)

Operation:

```
RPC ← PC + 1

PC ← absolute_address
```

Assembler:

```
Syntax: jsr r

Example:jsr LLL
```

Jumps to subroutine starting at LLL; the assembler translates the label to exact address

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine located at the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaaa - address field:

```
00000000000000 - 0

00000000000001 - 1

...

11111111111110 - 16382

11111111111111 - 16383
```

43.5.2.26 JSRR (Unconditional Jump to Subroutine)

Operation:

```
RPC ← PC + 1

PC ← GReg[r]
```

Assembler:

```
Syntax: jsrr r

Example:jsrr 5
```

Jumps to subroutine located at address stored in GReg[5]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine at address contained in a General Register

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.27 LD (Load Register)

Operation:

```
GReg[r] ← [GReg[b] + displacement]

if (transfer_error)

SF ← 1

else

SF ← 0
```

Assembler:

```
Syntax: ld r,(b,displacement)

Example: ld 1,(2,23)
```

Loads data into GReg[1]; the data is located at address obtained by adding decimal value 23 to GReg[2]

CPU Flags: SF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to fetch on the DM bus. The data received from the bus is stored in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	r	r	r	d	d	d	d	d	b	b	b

rrr / bbb - register field:

```
000 - GReg[0]

001 - GReg[1]

...

111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0

00001 - 1

...

11111 - 31
```

43.5.2.28 LDF (Load Register from Functional Unit)

Operation:

```
GReg[r] ← [fu_address]

if (transfer_error)

SF ← 1

else

SF ←0
```

fu_address is an 8-bit field and depends on addressed functional unit

Assembler:

```
Syntax: ldf r,fu_address

Example: ldf 0,13
```

Loads data coming from the Burst DMA register MD into GReg[0]; it is a 32-bit access with no prefetch

CPU Flags: SF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and stores the data received from the bus in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the LDF instruction usage with each functional unit:

- Burst DMA Read (ldf) for Burst DMA
- Peripheral DMA Read (ldf)-Read Mode for Peripheral DMA

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit source register and action (unspecified values are reserved):

```
00000000 - MSA

00000100 - MDA

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word

00001100 - MS

00101001 - MD byte - prefetch

00101010 - MD halfword - prefetch

00101011 - MD word - prefetch

01000000 - DSA
```

```
11000000 - PSA

11001000 - PD

11010000 - PDA

11011000 - PD in copy mode (rrr contents are lost)

11101000 - PD - prefetch next data

11111111 - PS
```

43.5.2.29 LDI (Load Register with Immediate Value)

Operation:

```
GReg[r] ← immediate
```

Assembler:

```
Syntax: ldi r,immediate

Example: ldi 6,1
```

loads decimal value 1 into GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Stores a 0-extended immediate value in a General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]
```

110 - GReg[6]

111 - GReg[7]

iiiiiiii - immediate value:

00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255

43.5.2.30 LDRPC (Load from RPC to Register)

Operation:

GReg[r] ← RPC

Assembler:

Syntax: ldrpc r

Example: ldrpc 3

copies RPC to GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Stores the contents of the RPC in a General Register. That instruction may be used to have more than one level of subroutines.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	1	0

Instruction Fields:

rrr - register field:

000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

```
101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.31 LOOP (Hardware Loop)

Operation:

```
if (ff%2 == 0)
   SF ← 0
if (ff/2 == 0)
  DF ← 0
if ((GReg[0] == 0) || (SF == 1) || (DF == 1))
   PC ← PC + loop_size + 1
else
   {
      SPC ← PC + 1
      EPC ← PC + loop_size + 1
      LM ← 1
      PC ← PC + 1
   }
```

during every instruction execution in the loop:

```
if ((SF == 1) || (DF == 1))
 {
    LM ← 0
    PC ← EPC
  }
else if ((PC + 1) == EPC)
 {
    GReg[0] ← GReg[0] - 1
    if (GReg[0] == 0)
      {
         LM ← 0
         PC ← EPC
```

```
       }
     else
         PC ← SPC
   }
else
  PC ← nextPC(instruction)
```

after the execution of the last instruction of the loop body:

```
if (GReg[0] == 0)
   T ← 1
else
   T ← 0
```

Assembler:

```
Syntax: loop n{,ff}
Example: loop 3,1
```

Executes GReg[0] times the instructions comprised between PC+1 and PC+3 (included); ff=1 clears the DF flag before starting the loop. When omitted, the ff field is set to 0 (clearing both SF and DF).

CPU Flags: LM[1:0], T

Cycles: 2 when the loop count (GReg[0]) is 0 or SF or DF is set at loop start, 1+1 when the loop starts but exits abnormally (SF or DF set inside the loop which adds 1 cycle to the offending load or store to jump to EPC), 1 when the loop is executed normally

Description: The loop instruction executes a sequence of instructions several times. The number of times is given by the contents of GReg[0], the loop counter. SDMA will jump to the first instruction after the end of the loop if the value in GReg[0] is 0. Otherwise the SDMA enters loop mode. It sets the most significant bit of the LM flag that will only be reset once the last instruction of the last loop is executed. The instructions in the loop are executed GReg[0] times. The management of fault flags (SF and DF) is as follows. When entering the hardware loop, SF and DF can be cleared according to the ff field of the instruction. After that operation, if any flag is still set the loop will not be executed. The SDMA will jump to the first instruction after the end of the loop without entering loop mode. During the execution of the loop, if any fault flag is set by a LD, LDF, ST, or STF instruction, the SDMA will immediately exit loop mode and jump to the first instruction after the end of the loop. In that case, GReg0 is not decremented for that last piece of the loop body execution (even if the SF or DF flag is set at the last instruction of the loop body). The T flag reflects the state of GReg[0] after the end of the loop, which is an indicator of the complete execution of the loop. If the loop exited because of an error (SF

or DF set), GReg[0] will not be 0 at the end of the loop, hence T will be cleared. If the loop executes without fault, GReg[0] will be 0 at the end of the loop, hence T will be set. The boundary case when a source or destination fault occurs at the last instruction of the last loop is considered as an anticipated exit of the loop, which causes the T flag to be cleared. If the last instruction executed before leaving the hardware loop also tries to modify the T flag, the flag is updated according to the value of GReg[0], NOT according to the result of the last executed instruction.

Limitations:

1. 1. Jump instructions (JMP, JMPR, JSR, JSRR, BF, BT, BSF, BDF) are not allowed inside the hardware loop.
2. 2. GReg[0] cannot be written to inside the hardware loop (it can be read).
3. 3. The empty loop (0 instruction in the body) is forbidden.
4. 4. If GReg[0] == 0 at the start of the loop, which causes a jump to EPC, the T flag is not updated.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	f	f	n	n	n	n	n	n	n	n

Instruction Fields:

```
ff - flags field:

00 - clear SF and clear DF

01 - clear DF

10 - clear SF

11 - no clear
```

nnnnnnnn - loop size

```
00000000 - empty loop: forbidden value

00000001 - 1 instruction in the loop

00000010 - 2 instructions in the loop

...

11111111 - 255 instructions in the loop
```

43.5.2.32 LSL1 (Logical Shift Left by 1 Bit)

Operation:

```
GReg[r]:{b30,...,b1,b0,0} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: lsl1 r

Example: lsl1 2
```

multiplies by 2 the value in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the left. The right bit (bit 0) is set to 0. No overflow is detected by the hardware.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.33 LSR1 (Logical Shift Right by 1 Bit)

Operation:

```
GReg[r]:{0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: lsr1 r

Example: lsr1 4
```

divides by 2 the unsigned value contained in GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the right. The left bit (bit 31) is set to 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	1

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.34 MOV (Logical Move)

Operation:

```
GReg[r] ← GReg[s]
```

Assembler:

```
Syntax: mov r,s

Example: mov 4,0
```

copies GReg[0] to GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Move the contents of the source General Register s to the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.35 NOTIFY (Notify to ARM platform)

Operation:

```
 if (jjj & 4 == 0)
 {
   if (jjj&2 == 2)
      HE[CCR] ← 0
   if (jjj&1== 1)
       HI[CCR] ← 1
  }
else if (jjj == 4)
   EP[CCR] ← 0
else
```

(CCR stands for Current Channel Register)

Assembler:

```
Syntax: notify jjj

Example: notify 3
```

clears the HE bit for the current channel and sends an interrupt to the Host for the current channel

CPU Flags: Unaffected

Cycles: 1

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required, sends an interrupt to the corresponding ARM platform by setting the appropriate flag if required (HI for the corresponding channel number).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	1

jjj - Channel Flags field:

```
000 - unused

001 - set HI for the current channel

010 - clear HE for the current channel

011 - clear HE, set HI for the current channel

100 - clear EP for the current channel

101 - RESERVED

110 - RESERVED

111 - RESERVED
```

43.5.2.36 OR (Logical OR)

Operation:

```
GReg[r] ← GReg[s] | GReg[r]
```

Assembler:

```
Syntax: or r,s

Example: or 3,6
```

ORs GReg[3] and GReg[6] and stores the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.37 ORI (Logical OR with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] | immediate
```

Assembler:

```
Syntax: ori r,immediate

Example: ori 1,56
```

ORs GReg[1] and the decimal value 56 and stores the result in GReg[1]

CPU Flags: unaffected

Cycles: 1

Description: Performs an OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

43.5.2.38 RET (Return from Subroutine)

Operation:

```
PC ← RPC
```

Assembler:

```
Syntax: ret
```

CPU Flags: Unaffected

Cycles: 2

Description: Return from subroutine.

Instruction Format

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

43.5.2.39 REVB (Reverse Byte Order)

Operation:

```
GReg[r]:{B3,B2,B1,B0} ← GReg[r]:{B0,B1,B2,B3}
```

Assembler:

```
Syntax: revb r

Example:  revb 5
```

reverses bytes order in GReg[5]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse the byte order of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.40 Reverse Low Order Bytes(REVBLO)

Operation:

```
GReg[r]:{B3,B2,B0,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: revblo r

Example: revblo 0
```

reverses low order bytes in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse both low order bytes of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.41 ROR1 (Rotate Right by 1 Bit)

Operation:

```
GReg[r]:{b0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: ror1 r

Example: ror1 3
```

rotates bits to the right in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bits of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.42 RORB (Rotate Right by 1 Byte)

Operation:

```
GReg[r]:{B0,B3,B2,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: rorb r

Example: rorb 2
```

rotates bytes to the right in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bytes of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.43 SOFTBKPT (Software Breakpoint)

Operation:

Stops the current script and enters debug mode

Assembler:

```
softbkpt
```

CPU Flags: Unaffected

Description: When the core executes this instruction, it has the same effect as receiving a debug request from the OnCE or via the external debug request input: the script execution halts, the PCU enters its debug state and waits for the OnCE commands that are described in OnCE and Real-Time Debug.

Instruction Format

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

43.5.2.44 ST (Store Register)

Operation:

```
[GReg[b] + displacement] ← GReg[r]

if (transfer_error)
```

```
 DF ← 1

 else

DF ← 0
```

Assembler:

```
Syntax: st r,(b,displacement)

Example: st 7,(0,9)
```

stores the value from GReg[7] into memory at address obtained by adding decimal value 9 to GReg[0]

CPU Flags: DF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to store on the DM bus. The data sent on the bus comes from the source General Register r. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	r	r	r	d	d	d	d	d	b	b	b

Instruction Fields:

rrr / bbb - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0

00001 - 1
```

```
...

11111 - 31
```

43.5.2.45 STF (Store Register in Functional Unit)

Operation:

```
[fu_address] ← GReg[r] 0

if (transfer_error) 0

DF ← 1 0

else 0

DF ← 0
```

fu_address is an 8-bit field

Assembler:

```
Syntax: stf r,fu_address

Example: stf 3,0x2B
```

stores the 32-bit contents of GReg[3] to the Burst DMA register MD; waits until the flush to external memory is completed

CPU Flags: DF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and sends the contents of the source General Register r on the bus. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Table 43-47. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the STF instruction usage with each functional unit:

- Burst DMA Write (stf) for Burst DMA
- Peripheral DMA Write (stf)-Write Mode for Peripheral DMA

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit destination register and action (unspecified values are reserved):

00000000 - MSA in incremented mode

00000100 - MDA in incremented mode

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word

00001100 - clear MS error flag

00001111 - MS

00010000 - MSA in frozen mode

00010100 - MDA in frozen mode

00011000 - MD in copy mode - number of words in rrr

00100000 - MSA in incremented mode - start prefetch

00101000 - MD no data - flush

00101001 - MD byte - flush

00101010 - MD halfword - flush

00101011 - MD word - flush

00110000 - MSA in frozen mode - start prefetch

11000001 - PSA in frozen mode - 8-bit data width

11000010 - PSA in frozen mode - 16-bit data width

11000011 - PSA in frozen mode - 32-bit data width

11000101 - PSA in incremented mode - 8-bit data width

11000110 - PSA in incremented mode - 16-bit data width

11000111 - PSA in incremented mode - 32-bit data width

11001000 - PD

11001001 - PSA in decremented mode - 8-bit data width

11001010 - PSA in decremented mode - 16-bit data width

11001011 - PSA in decremented mode - 32-bit data width

11001100 - clear PS error flag

11001101 - PSA data width becomes 8-bit

11001110 - PSA data width becomes 16-bit

11001111 - PSA data width becomes 32-bit

11010001 - PDA in frozen mode - 8-bit data width

11010010 - PDA in frozen mode - 16-bit data width

11010011 - PDA in frozen mode - 32-bit data width

11010101 - PDA in incremented mode - 8-bit data width

11010110 - PDA in incremented mode - 16-bit data width

11010111 - PDA in incremented mode - 32-bit data width

11011001 - PDA in decremented mode - 8-bit data width

11011010 - PDA in decremented mode - 16-bit data width

11011011 - PDA in decremented mode - 32-bit data width

11011101 - PDA data width becomes 8-bit

11011110 - PDA data width becomes 16-bit

11011111 - PDA data width becomes 32-bit

11100001 - PSA in frozen mode - 8-bit data width - prefetch data

11100010 - PSA in frozen mode - 16-bit data width - prefetch data

11100011 - PSA in frozen mode - 32-bit data width - prefetch data

11100101 - PSA in incremented mode - 8-bit data width - prefetch data

11100110 - PSA in incremented mode - 16-bit data width - prefetch data

11100111 - PSA in incremented mode - 32-bit data width - prefetch data

11101001 - PSA in decremented mode - 8-bit data width - prefetch data

11101010 - PSA in decremented mode - 16-bit data width - prefetch data

11101011 - PSA in decremented mode - 32-bit data width - prefetch data

11101101 - PSA data width becomes 8-bit - prefetch data

11101110 - PSA data width becomes 16-bit - prefetch data

11101111 - PSA data width becomes 32-bit - prefetch data

11111111- PS

43.5.2.46 SUB (Subtract)

Operation:

```
GReg[r] ← GReg[r] - GReg[s]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,s

Example: sub 4,7
```

SUBtracts GReg[7] from GReg[4] and stores the result in GReg[4]

CPU Flags: T

Cycles: 1

Description: Subtracts the source General Register s from the destination General Register r, and stores the result in the destination General Register r. The T flag is set if the result of the operation is 0; it is cleared if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	0	s	s	s

Instruction Fields:

rrr / sss - register fields:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.47 SUBI (Subtract with Immediate)

Operation:

```
GReg[r] ← GReg[r] - immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,immediate

Example: sub 1,255
```

SUBtracts decimal value 255 from GReg[1] and stores the result in GReg[1]

CPU Flags: T

Cycles: 1

Description: Subtracts a 0-extended 8-bit immediate value from a General Register; stores the result in the General Register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

43.5.2.48 TST (Test with Zero)

Operation:

```
T ← ((GReg[s] & GReg[r]) != 0)
```

Assembler:

```
Syntax: tst r,s

Example: tst 2,3
```

ANDs GReg[2] and GReg[3] and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of the source General Register s and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	0	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

43.5.2.49 TSTI (Test Immediate)

Operation:

```
T ← ((GReg[r] & immediate) != 0)
```

Assembler:

```
Syntax: tsti r,immediate

Example: tsti 5,13
```

ANDs GReg[5] and decimal value 13 and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of a 0-extended 8-bit immediate value and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]
```

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]

iiiiiiii - immediate value:

00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255

43.5.2.50 XOR (Logical Exclusive OR)

Operation:

GReg[r] ← GReg[s] ^ GReg[r]

Assembler:

Syntax: xor r,s

Example: xor 0,3

XORs GReg[0] and GReg[3] and stores the result in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the eXclusive OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	0	s	s	s

Instruction Fields:

rrr / sss - register field:

000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]

43.5.2.51 XORI (Exclusive OR with Immediate)

Operation:

GReg[r] ← GReg[r] ^ immediate

Assembler:

Syntax: xori r,immediate

Example: xor 7,5

XORs GReg[5] and decimal value 5 and stores the result in GReg[7]

CPU Flags: Unaffected

Cycles: 1

Description: Performs an eXclusive OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

```
100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

43.5.2.52 YIELD, YIELDGE (DONE, Yield)

By default, unsupported assembler syntax. Can be aliased to the corresponding done instructions (yield = done 0; yieldge = done 1). Refer to the done instruction description DONE (DONE, Yield) .

43.6 Software Restrictions

43.6.1 Unsupported Burst DMA Access Sequence

The SDMA does not support triggering a pre-fetch followed by a flush of the Burst DMA without reading or writing any data. If the flush occurs while the background pre-fetch DMA operation is still in progress, it could result in un-defined behavior.

An example of the sequence which could result in undefined results is shown in the following example:

Instruction sequence not supported

```
stf r1, MSA|PF          ; Update source address, triggers data pre-fetch in the
                        ; background
mov R0,R0               ; Execute multiple assembly instructions, none of which
                        ; read
mov R0,R0               ; or write data to/from MD
stf MD|SZ0|FL           ; Flush FIFO without writing data. If the pre-fetch is still
                        ; in progress when this instruction is executed, there
                        ; could be undefined operation
```

A work-around to avoid any undesirable results is to first read MD to ensure the pre-fetch is complete before the flush is attempted.

Work-Around to previous example

```
stf r1, MSA|PF          ; Update source address, triggers data pre-fetch.
mov R0,R0               ; Execute multiple assembly instructions, none of which
                        ; read
mov R0,R0               ; or write data to/from MD
ldf r2, MD              ; dummy read of MD to ensure pre-fetch is complete
                        ; before the next instruction
stf MD|SZ0|FL           ; Flush FIFO without writing data
```

43.7 Application Notes

43.7.1 Data Structures for Boot Code and Channel Scripts

SDMA boot code downloads the different channel contexts and the scripts that will be executed on SDMA channels during the application.

The boot code is run after reset when channel 0 is started by the ARM platform. The boot code is also known as channel 0 script.

The boot code is based on the Channel Control Block (CCB) and Buffer Descriptor (BD) mechanisms that are data structures located into the ARM platform memory space. With these data structures, it is possible to instruct SDMA to download scripts and contexts but also to dump a context or a script to a destination data buffer. Channel scripts also use the CCB and BD data structures to pass instructions and/or pointers to data to be copied.

The format, processing, and field definition of the CCB and BD are defined and performed entirely by the software script rather than the SDMA hardware. An overview of the format and structure is provided here, but for complete details refer to the SDMA software documentation (see SDMA Scripts).

The CCB and BD data structures are accessed by SDMA using DMA and processed by the SDMA scripts. The ROM contains common sub-routines for processing these data structures which may be called by the bootload and channel scripts.

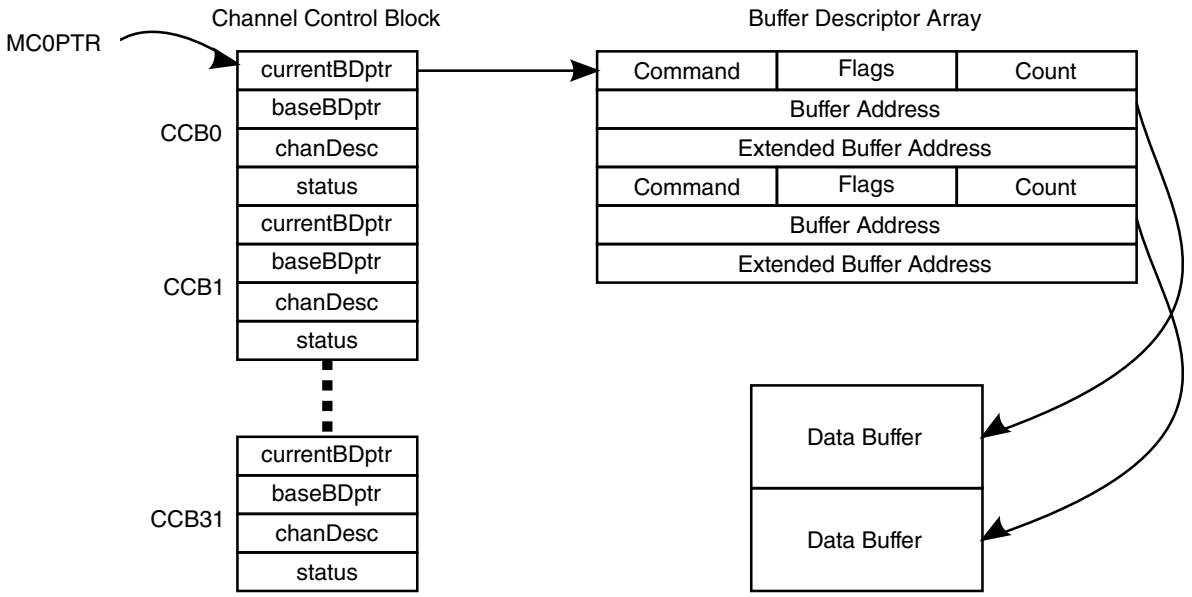


Figure 43-16. Data Structures Layout

The previous figure shows an example how these data structures are linked to pass command and pointers to data buffers. The SDMA's MC0PTR register holds the base address of the Channel 0 Control Block (CCB0). The Channel 0 control block holds a pointer to the array of buffer descriptors. The buffer descriptors are used to tell the channel 0 (boot channel) what to do as described Buffer Descriptor Format.

43.7.1.1 Buffer Descriptor Format

Buffer descriptors are three longs (32-bit words) in size as, shown in the figure found here.

A buffer descriptor describes the properties of the data buffer it points to. The buffer descriptors can be used for linear or circular data buffers in the ARM platform processor memory. The CCB contains a pointer to the base BD as well as the current BD.

Table 43-48. Buffer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command								-	-	L	R	I	C	W	D	Count															
Buffer Address																															
Extended Buffer Address																															

Table 43-49. Buffer Descriptor Field Descriptions

Field	Description
31-24 Command	Command. The command field is used to differentiate operations performed within a script when the script accesses this particular buffer descriptor. The use of this field can be defined by the script. The command values defined for the bootload script are defined in Buffer Descriptor Commands for Bootload scripts. Refer to the individual script definition in script library documents in SDMA Scripts for command field definitions for other scripts.
23	Reserved
22	Reserved
21 L	Last Buffer Descriptor: This bit is set in SDMA IPC scripts to indicate to the receiving Core that the transfer has ended. Whenever the source finishes transferring the count it wanted to transfer, it sets LAST_BIT in the destination BD, to let the destination know that transfer is over.This bit also tells the destination software that when it processes the destination BDs, they need not process any BD after the BD with the LAST_BIT set.For example, when the DSP prepares a single buffer descriptor with count equals to 25 and ARM platform prepares a single buffer descriptor with count equals 100. When 25 bytes have been transferred from DSP to ARM platform, the DSP buffer descriptor is normally closed while the ARM platform buffer descriptor will have the L bit set and the byte count updated to 25.
20 R	erroR. Indicates an error occurred on the channel's buffer descriptor requested command. Some scripts may overwrite the command field with an error code indicating the source of the error. 0 No Error 1 Error
19 I	Interrupt. When SDMA has finished to process data transfer attached to this buffer descriptor, send an interrupt to the ARM platform. 0 No Interrupt 1 Interrupt the processor when BD is complete
18 C	Continuous. This buffer is allowed to receive multiple transmit buffers or is allowed to transmit to multiple receive buffers.The Continuous bit is decoded at the end of the processing of a BD to determine if the SDMA script must open a new BD to potentially continue the data transfer. 0 No further buffer descriptors 1 SDMA should move to the next Buffer descriptor after this one
17 W	Wrap. Indicates if this buffer descriptor is the last one for the channel control block. When encountering this bit set, the SDMA scripts updates the CurrentBD pointer to point to the first Buffer Descriptor of the array. This bit is set if the ARM platform wants to organize the array of BD in a circular way (like a ring). When all BD have been processed and if Wrap bit and CONtinuous bit are set in the last BD, the SDMA script will wrap around and it will try to re-open the first BD. 0 No Error 1 Wrap to first buffer descriptor after this one is processed.
16 D	D - "Done": bit 16: indicates the "ownership" of the buffer descriptor. When D=0 the host owns the buffer descriptor; when D=1 SDMA owns the buffer descriptor. In the case of the channel 0, D=1 indicates the SDMA has not yet processed this buffer, D=0 indicates the SDMA has processed this buffer. 0 ARM platform owns the buffer. 1 SDMA owns the buffer
15-0 Count	Count. the count field (bit 15-0) indicates the size of the data to be transmitted, the size of the data buffer pointed to by the buffer descriptor. The SDMA memory structure is different for program memory (16-bits shorts/half-words) and data memory (32-bits long). For channel 0 buffer descriptors, Count is expressed in 16-bit half-words when PM is addressed and in 32-bit words when DM is addressed. Count is typically expressed in bytes for other channel scripts, but the unit is dependant on the script.
31-0	Buffer address. Address pointer to the data buffer.
31-0	Extended buffer address. Additional pointer or other information required by some scripts.

The buffer descriptors form an array of programmable size. If the last buffer descriptor is marked by the Wrap flag-bit W=1, the array of buffer descriptor is treated as a ring with some logically continuous portion owned by the ARM platform with D=0, and the remainder owned by the SDMA with D=1. The count field of the buffer descriptor indicates how much data has been transmitted.

If ARM platform has prepared 3 buffers to be filled by the SDMA script, it has also prepared 3 BD, one for each buffer. The *Cont* and *Wrap* bits are used to organize the buffers in a circular way. For example, *CONT*inous bit is set to 1 in the 2 first BDs and *Wrap* is set in the 3rd BD. The SDMA script opens and processes BD#1. Since *CONT*inous bit is set for this BD, the SDMA will open the second BD and it will process it. Each time a BD is processed, its *Done* bit is reset by the SDMA. After the 3rd BD, if *CONT*inous is not set but if *Wrap* is set, the SDMA script stops here and the next time the channel will be triggered, the script will open the BD pointed by the currentBDptr pointer of the CCB and it will correspond to the first buffer descriptor.

If the *CONT*inous bit and *Wrap* bits are both set in the 3rd BD, the script will close it and it will try to open the first BD. An error may occur at this point if the BD#1 has already been processed and its *Done* bit is 0. The SDMA script cannot process a BD with a *Done* bit to 0. It means the BD is not ready to be processed. To avoid this situation, the *CONT*inous bit should not be set for the last BD if *Wrap* is set, and the Interrupt flag must set for the last BD. It will warn the owner of the BD that all the BDs have been processed and it has to re-set to 1 the *Done* bit of all the BD's if it desires the SDMA to fill them again. Basically, if the ARM platform expects the SDMA to fill up the buffers in a circular fashion, then it's the responsibility of the ARM platform to set the *Done* bit of a buffer descriptor at an appropriate time.

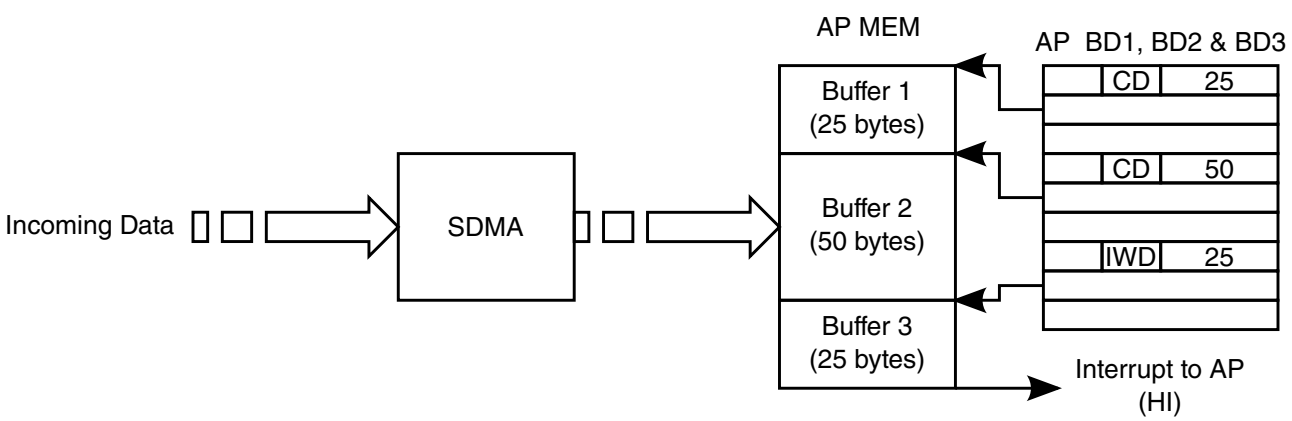


Figure 43-17. Buffer Descriptor Flow

The previous figure shows an example buffer descriptor flow. When the incoming data is stored and fills the first buffer of 25 bytes, the SDMA script opens the second BD because the CONTinuous bit was set. Then next incoming data is put in the second buffer. After receiving 50 bytes, the second buffer descriptor is also closed. The Done bit is reset and the third BD is opened. After receiving another 25 bytes, the third buffer is full and an interrupt is sent to the ARM platform because the Interrupt flag is set in the 3rd BD. The CONTinuous flag is not present the transfer is over. The next time the script will be triggered, the BD to be opened will be the first buffer descriptor since the Wrap flag was set in the 3rd BD. It is the ARM platform responsibility to set the Done bit of all the BD if it wants to use the same buffers.

43.7.1.2 Buffer Descriptor Commands for Bootload scripts

The command field of the buffer descriptor is defined separately for each script.

The following table lists the buffer descriptor commands defined for the channel 0 bootload script.

Table 43-50. Channel Zero Buffer Descriptor Commands

Command Field (binary)	Command	Description	Buffer Address	Extended Buffer Address
0000_0001 (0x01)	C0_SET_DM	Load SDMA data memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0010 (0x02)	C0_GET_DM	Copy SDMA data memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
0000_0100 (0x04)	C0_SET_PM	Load SDMA program memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0110 (0x06)	C0_GET_PM	Copy SDMA program memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
cccc_c111 (0x07 | CHN)	C0_SETCTX	Load Context for channel cccc into SDMA RAM from ARM platform memory buffer	ARM Platform memory source address	-
cccc_c011 (0x03 | CHN)	C0_GETCTXT	Copy Context for channel ccccc from SDMA RAM to ARM platform memory buffer	ARM platform memory destination address	-

The Channel 0 bootload commands are summarized as follows:

- C0_SET_[PM-DM]: load the buffer descriptor data in the SDMA local memory at the address pointed to by the "extended buffer address" field. The SDMA RAM can be seen as a Program Memory (PM, 16-bit address) or Data Memory (DM 32-bit address). When C0_SET_PM is used, the count field is expressed in "shorts" (16-bit

half words), this command can be used to download scripts. When C0_SET_DM is used, the count field is expressed in "long" (32-bit words), this command can be used to download channel contexts to the context channel area in RAM.

- C0_GET_[PM-DM]: write to the buffer descriptor's data buffer the content of the SDMA local memory from the address pointed to by the "extended buffer address" field for the length defined by the count in the buffer descriptor. C0_GET_PM is used to dump some part of the Program Memory (may be used to dump context of a channel), therefore count is expressed in "shorts"; while C0_GET_DM is used to dump to the buffer descriptor's data buffer, so the count field is in "longs."
- C0_SETCTX: load a context into the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using the channel number the script computes the offset of the context data pointer for the channel relative to the context page base to use as the destination address in SDMA memory. Then the C0_SET_DM command explained above is invoked to load SDMA RAM from memory. The counter indicates the size in words of the context structure.
- Command value: (in binary) cccc c111, where ccccc is the channel number (5 bits). For instance, 0x0F means set context for channel 1, 0xFF means set context for channel 31.

- C0_GETCTX: write to the buffer descriptor's data buffer the content of the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using this channel number, the script computes the offset of the context data pointer for the channel relative to the context page base to use as the source address for the copy. Then the C0_GET_DM command explained above is invoked to copy the context to memory. The counter indicates the size in words of the context structure.
- Command value: (in binary): cccc c011, where ccccc is the channel number (5 bits). For instance, 0x03 means get context of channel 1, 0xFB means get context of channel 31.

NOTE

To download channel context, C0_SETDM and C0_SETCTXT command can be used but the second one is easier because the channel number is embedded into the command field, whereas with the C0_SETDM, the pointer to the channel context area must be written into the extended buffer address field of the buffer descriptor.

43.7.1.3 Example of Buffer Descriptors for Channel 0.

Figure 43-19 illustrates the buffer descriptors that must be set in ARM platform memory space, before execution of boot code, to download contexts and scripts of channels 1, 4, and 10. After boot code execution, SDMA memory will be populated with the different contexts and scripts as presented in the following figure.

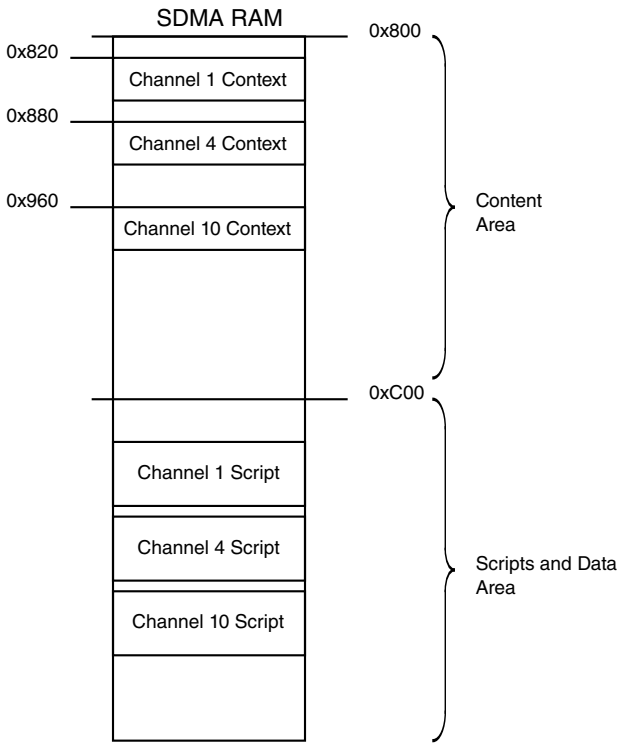
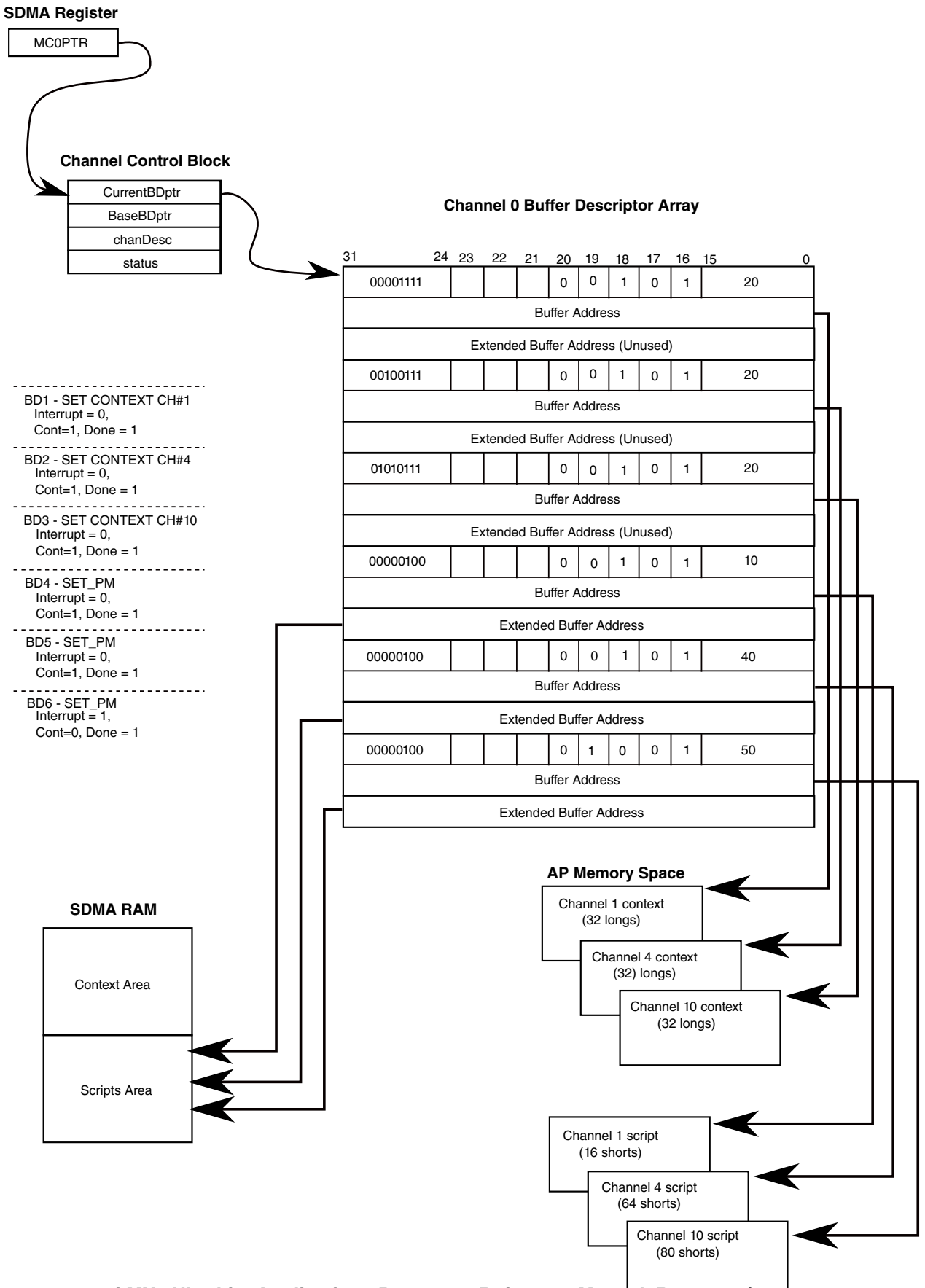


Figure 43-18. Example of SDMA RAM After Boot Session


SDMA Register
MC0PTR
Channel Control Block
CurrentBDptr
BaseBDptr
chanDesc
status
Channel 0 Buffer Descriptor Array
31 24 23 22 21 20 19 18 17 16 15 0
00001111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00100111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
01010111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00000100 0 0 1 0 1 10
Buffer Address
Extended Buffer Address
00000100 0 0 1 0 1 40
Buffer Address
Extended Buffer Address
00000100 0 1 0 0 1 50
Buffer Address
Extended Buffer Address
BD1 - SET CONTEXT CH#1
Interrupt = 0,
Cont=1, Done = 1
BD2 - SET CONTEXT CH#4
Interrupt = 0,
Cont=1, Done = 1
BD3 - SET CONTEXT CH#10
Interrupt = 0,
Cont=1, Done = 1
BD4 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD5 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD6 - SET_PM
Interrupt = 1,
Cont=0, Done = 1
SDMA RAM
Context Area
Scripts Area
AP Memory Space
Channel 1 context
(32 longs)
Channel 4 context
(32) longs)
Channel 10 context
(32 longs)
Channel 1 script
(16 shorts)
Channel 4 script
(64 shorts)
Channel 10 script
(80 shorts)

43.7.1.4 Channel Context

There are 32 channel context memory structures pointed to by the local save area pointer. These channel context memory structures are fixed.

The script in the SDMA computes the memory offset for a given channel based on the structure length and channel number. Figure below shows the structure of the channel context as it is saved in the SDMA local memory (RAM).

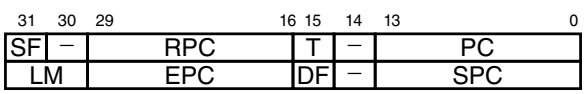
A channel context consists in 24 words, one per register. A total of 32 words are reserved for every channel. The additional 8 words are called scratch ram and they are dedicated to each channel. This memory area is commonly used for stack management.

The structure is divided in 4 areas:

- Channel status registers
- General purpose registers
- Functional units state registers reflecting the state of the ARM platform DMAs (Burst and Peripheral DMA).
- Scratch RAM

The details of the channel context status registers are described in the following figure.

The PC field of the first long register must point to the SDMA RAM address where the script that will be executed on the channel is located and this value equals the one stored in the extended buffer address of the buffer descriptor with C0_SETPM command.



SF: Source fault while loading data
RPC: Return program counter
T: Test bit: status of arithmetic and test instructions
PC: Program counter
LM: Loop mode
EPC: Loop end program counter
DF: Destination fault while storing data
SPC: Loop Start program counter

Figure 43-20. SDMA State Registers (ShPC, ShLoop)

43.7.2 Typical Data Transfer Supported by SDMA DMA Units

This section presents a library of SDMA scripts that perform data transfers through the peripheral DMA and the burst DMA units.

The ARM platform memory and peripherals are devices that either the peripheral DMA or the burst DMA can access. The scripts are given for a peripheral DMA whose address registers are programmed in incremented mode when internal memory is involved. See the following table for the summary.

Table 43-51. Typical Data Transfers Summary

Data Transfer	Peripheral DMA	Burst DMA	Comments
ARM platform External Memory ↔ ARM platform External Memory		3	Copy mode Script example, see Burst DMA Unit Copy Mode and External Memory to External Memory.
ARM platform Peripheral ↔ ARM platform Peripheral	3		Copy mode if same data path width Script example, see Peripheral to Peripheral Transfer.
ARM platform External Memory ↔ ARM platform Peripheral	3	3	Data transit through SDMA Script example, see Transfer Between Peripheral and External Memory.
ARM platform External Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Transfer Between External Memory and Internal Memory.
ARM platform Internal Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Internal Memory to Internal Memory.
ARM platform Internal memory ↔ ARM platform Peripheral	3		Data transit through SDMA Script example, see Transfer Between Peripheral and Internal Memory.

NOTE

These scripts are provided as examples of how to use DMA blocks to perform required data transfers: They are not "official" programs.

43.7.2.1 External Memory to External Memory

This section describes the SDMA script that performs data moves in external memory.

For this particular data transfer, only the burst DMA is used. It is programmed in copy mode, so no data transmits through an SDMA general register.

The SDMA core only monitors data transfer status. It is assumed source and destination address values are already present in two SDMA general registers (r1 and r2). For this example, it is also assumed that a 32-bit word-to-move for source-to-destination address is present in r0 and equals 64.

Data Moves in External Memory

```
1         stf r1,MSA                          // Source address setup

2         stf r2,MDA                          // Destination address setup

3         ldi r0,0x64                         // 64 words must be transferred from MSA to
MDA

4         ldi r1,0x8

MAIN_XFER:

5         cmphs r0,r1                         // Is r0 >= 0x8

6         bf LAST_XFER                      // If not, jump to last transfer label

7         stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.

8         subi r0,0x8                         // Decrement counter

9         jmp MAIN_XFER                     // return to main transfer loop

LAST_XFER:

10        stf r0,MD|CPY                        // perform last transfer
```

All instructions are performed in one cycle (jumps excepted). Instruction 7 triggers a copy transfer: A read burst access of 8-word starts, data is staged in MD and then a write burst of 8 words is executed. Instruction 8, 9, 5, and 6 are executed while the burst access is in progress. If this access is not complete when instruction 7 is executed a second time, SDMA stalls on this instruction as long as the previous copy transfer is not over. In this case, the instruction is no longer a one-cycle instruction.

During the main loop (MAIN_XFER), r1 always equals 8, so burst lengths are 8 words. On the last ldf |CPY instruction (10), r1 equals the reminder of r0 divided by 8; therefore, the length of bursts triggered in copy mode equal r1 value, which is between 1 and 7.

43.7.2.2 Peripheral to Peripheral Transfer

For this data transfer, only the peripheral DMA is used.

It is programmed in copy mode, so no data will transmit through the SDMA general register used in the ldf instruction, but the contents of the general register are lost. The SDMA core only monitors the transfer.

43.7.2.2.1 Source and Destination Target Have the Same Data Path Width

When the source and destination target have the same data path width, the following is true:

- Source target is a *half-word* (16-bit) peripheral located at address 0x1002.
- Destination is a *half-word* (16-bit) peripheral located at address 0x2006.

It is assumed the address values are already present in two SDMA general registers (r1, r2). The script for a transfer of 10 half-word is as follows:

Same Data Path Width for Source and Destination

```
//SETUP SECTION
1         stf r1, PSA|SZ16|F                    //r1=0x1002 Source address register setup
2         stf r2, PDA|SZ16|F                    //r2=0x2006 Destination address register
setup
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0xa                           //loop counter is 10
//MAIN LOOP TRANFER
copy_loop:

5         loop 2,0
6           ldf r7,PD|CPY                      //Reads 1 half-word from src and writes to
dest.
7           yield
8         bdf ERROR_DURING_XFER
ERROR_ADDR_SETUP:
                     //correction of PSA/PDA setup and jumps to main loop transfer
ERROR_DURING_XFER:
//flag error is set,
//PS can be read to know if error occurs during read or write access.
```

If a data transfer must occur between two word peripherals, only the setup section should be updated. The transfer itself is always performed by the hardware loop instruction.

All instructions are executed in one cycle (change of flow excepted). On instruction 6, a single read access is triggered, read data is staged in PD, and a write-to-destination is executed. When the transfers are in progress, the SDMA can execute he next instructions in parallel. If instruction 6, which performs the copy transfer, is executed while the previous access is not over, SDMA is stalled and instruction ldf is a multi-cycle instruction.

43.7.2.2.2 Source and Destination Target Have a Different Data Path Width

When the source and destination target have a different data path width, copy mode cannot be used, and any attempt to initiate a copy transfer immediately raises an error, which is stored in the SF flag.

The following example shows the SDMA code that could transfer 10 words from a *word* (32-bit) peripheral to a *half-word* peripheral whose addresses are preliminary and stored in r1 and r2.

Different Data Path Width for Source and Destination

```
//SETUP SECTION
1         stf r1, PSA|SZ32|F|PF                  //r1=0x1000 and prefetch data
2         stf r2, PDA|SZ16|F                    //r2=0x2006
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0xa                           //loop counter is 10
//MAIN LOOP TRANFER
main_loop_xfer_16_16:
5         loop 6,0
6           ldf r7,PD                         //copy 32-bit of PD in r7
7           stf r7,PD                          //store 16 LSB of r7 in PD and a flush is
executed
8           rorb r7
9           rorb r7                          //16 MSB --> 16 LSB
10          stf r7,PD                          //store 16 LSB of r6 in PD and a flush.
11          yield
```

On instruction 1, when the source address register is programmed and a data prefetch is required, a read access is executed. In parallel, the SDMA executes instructions 2 to 5. On instruction 6, the SDMA tries to read data that was fetched by instruction 1. If data is ready, the ldf will be a one cycle instruction; otherwise, the SDMA is stalled as long as the read access is not finished. Then, the 16 LSB of the read data is stored in PD and automatically flushed to the destination peripheral. In parallel, the SDMA executes the rotation instructions (8, 9), and stores the 16 MSB of the read data into PD. If a previous write access is finished, instruction 10 will be a one-cycle instruction.

The main loop transfer may appear inefficient, but due to wait states imposed to the peripheral DMA each time an external access is performed, a software pipeline is in place. During the time needed to flush PD, the SDMA executes the move and rotation operations. SDMA executes instructions in parallel with DMA accesses.

43.7.2.3 Transfer Between Peripheral and External Memory

43.7.2.3.1 Peripheral to External Memory Transfer

A transfer from a peripheral to the external memory controller involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from word peripheral to the external memory would be as follows:

Peripheral to External Memory Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1         stf r1, PSA|SZ16|F|PF                   //r1=0x1000 and prefetch 32-bit data
2         stf r2, MDA                           //r2=0x2000, setup burst DMA destination
address
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0x64                           //loop counter is 100
5
          //MAIN LOOP TRANFER
6         loop 3,0
```

```
7           ldf r1,PD|PF                    // read 32 bits of PD and initiate a new read
access.
8           stf r1,MD|32                     // store 32 bits of r1 in the MD fifo.
9           yield
10        ldf r1,PD                        // last word data is read
11        stf r1,MD|32|FL                    // to flush all remaining bytes of MD
```

On instruction 1, the source address register of the peripheral DMA is programmed and data is fetched. This data is stored in PD and the SDMA reads PD during instruction 7, which is a one-cycle instruction that is read-access finished. On the same instruction (7), a data prefetch is required and a read access to the source peripheral is executed. In parallel, the SDMA stored the previous read data into the data register of MD. When MD (which is an eight-word FIFO) is full, a burst write access is executed to empty the FIFO. As long as the next SDMA instructions do not access the burst DMA, they will be one-cycle instructions. The following figures show how the peripheral DMA and burst DMA work in parallel.

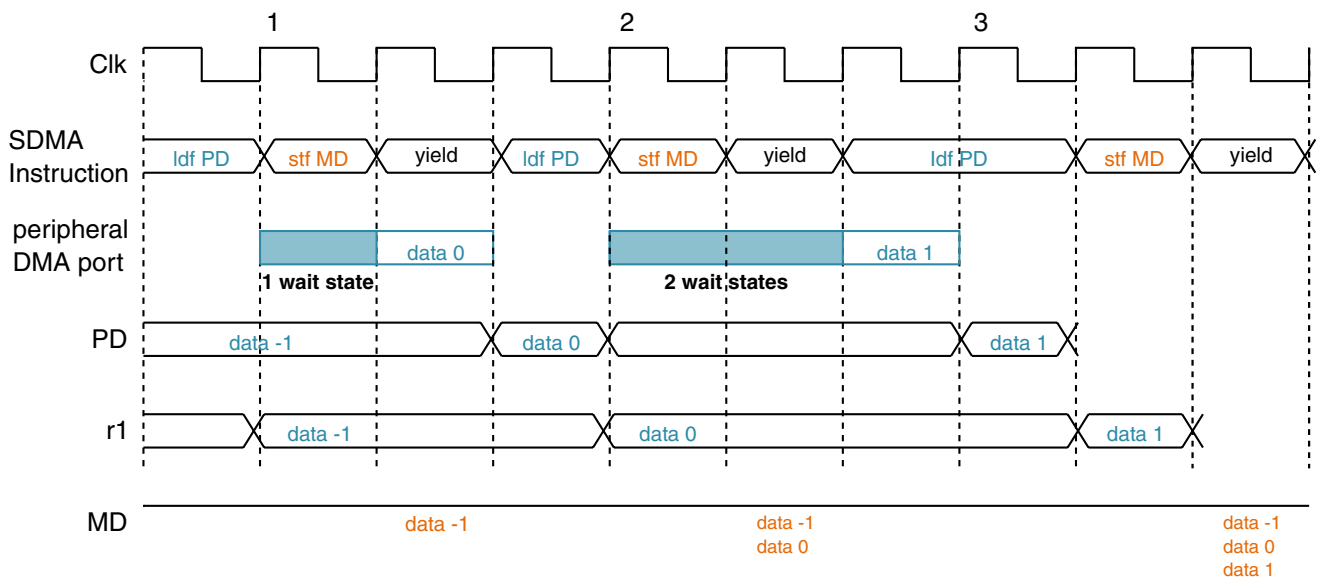


Figure 43-21. Peripheral to External Memory Example (1)

As seen in the figure above, the read access triggered by the ldf PD instruction is symbolized by the blue bar when in progress. After wait states, the read data (data 0, data 1) is stored in PD on the clk rising edge. On edge 2, data 0 is available in PD so it can be transferred to the SDMA general register r1, and then stored in MD FIFO. On edge 3, data 1 is not in PD; therefore, SDMA is stalled on the ldf instruction, which lasts two cycles. The figure below shows an example of when MD FIFO is full with data.

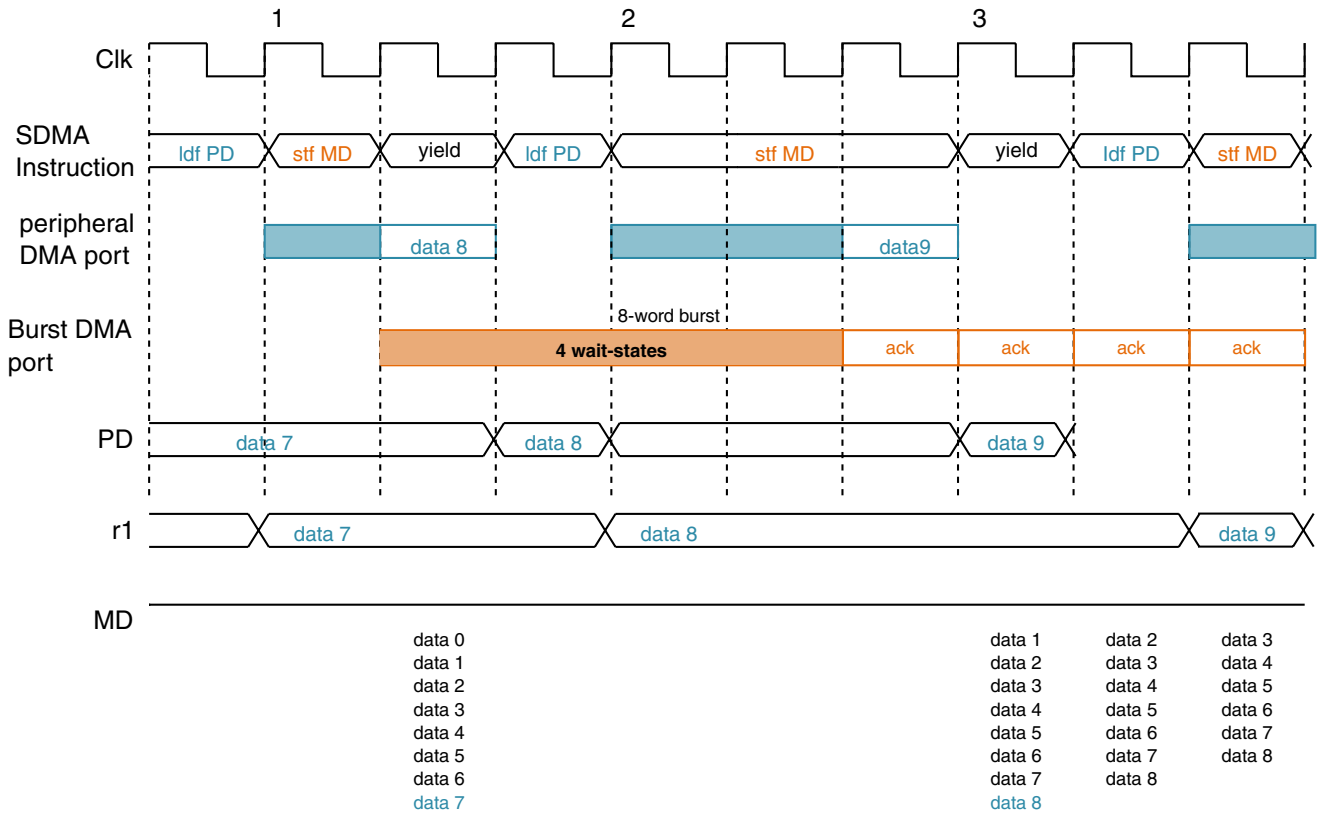


Figure 43-22. Peripheral to External Memory Example (2)

In the previous figure, the write bar means the burst DMA is performing a write burst access. The latency to have the first write acknowledge is four cycles. SDMA is stalled on instruction stf because no acknowledge was received, MD FIFO is full, and there is no empty slot to store data 9. When an acknowledge is sampled by the burst DMA, FIFO is shifted and data 8 is written. As long as there is at least one empty slot in MD FIFO, the stf MD instruction lasts one cycle.

43.7.2.3.2 External Memory to Peripheral Transfer

A transfer from the external memory to a peripheral involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from external memory to a word peripheral would be as follows:

External Memory to Peripheral Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1         stf r1, MSA|PF              //r1=0x1000 and starts a 8-word read burst
2         stf r2, PDA|SZ32|P           //r2=0x2010, setup peripheral DMA destination address
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0x64                //loop counter is 100
//MAIN LOOP TRANFER
6         loop 3,0
```

```
7           ldf r1,MD|32|PF          // read 32 bits of MD and initiate a new read access
                                  // if MD is empty after this reading.
8           stf r1,PD               // store 32 bits of r1 in the PD.
9           yield
          10 ldf r1,MD|32            // last word data is read
11        stf r1,PD                 // last write access
```

On instruction 1, a read burst of 8 words begins. Read data is staged into MD. On instruction 7 (and if data is available in MD), 32 bits are copied into r1. Then instruction 8 writes them into PD and an automatic flush is executed. The SDMA core, peripheral DMA, and burst DMA can work in parallel as long as no SDMA instruction tries to start a new write access on the peripheral DMA while the previous access is still in progress, or as long as there is data in MD when the SDMA tries to read it.

43.7.2.4 Transfer Between External Memory and Internal Memory

Since the internal memory (ARM platform RAM) is accessed via the peripheral DMA and the external memory is accessed via the burst DMA, the SDMA scripts that are described in Transfer Between Peripheral and External Memory can be reused. The exception is that the peripheral DMA address registers (PSA or PDA, depending on the script) should be programmed in incremented mode rather than frozen mode.

43.7.2.4.1 Internal Memory to Internal Memory

The internal memory can only be accessed via the peripheral DMA, so the script described in Peripheral to Peripheral Transfer can be reused with a different programming of the peripheral DMA address registers.

43.7.2.4.2 Transfer Between Peripheral and Internal Memory

For this transfer, the peripheral DMA is also used in copy mode.

The SDMA script is very similar to the one described in Peripheral to Peripheral Transfer, except for the peripheral DMA address registers programming.

43.8 ARM Platform Memory Map and Control Register Definitions

The ARM platform controls the SDMA by means of several interface registers. Those registers are described in the current section.

All registers are clocked with the SDMA clock (which means the ARM platform must ensure that the SDMA clock is running when it wants to access any register).

SDMAARM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C000	ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)	32	R/W	0000_0000h	43.8.1/2760
20E_C004	Channel Interrupts (SDMAARM_INTR)	32	w1c	0000_0000h	43.8.2/2760
20E_C008	Channel Stop/Channel Status (SDMAARM_STOP_STAT)	32	w1c	0000_0000h	43.8.3/2760
20E_C00C	Channel Start (SDMAARM_HSTART)	32	R/W	0000_0000h	43.8.4/2761
20E_C010	Channel Event Override (SDMAARM_EVTOVR)	32	R/W	0000_0000h	43.8.5/2761
20E_C014	Channel BP Override (SDMAARM_DSPOVR)	32	R/W	FFFF_FFFFh	43.8.6/2762
20E_C018	Channel ARM platform Override (SDMAARM_HOSTOVR)	32	R/W	0000_0000h	43.8.7/2762
20E_C01C	Channel Event Pending (SDMAARM_EVTPEND)	32	w1c	0000_0000h	43.8.8/2762
20E_C024	Reset Register (SDMAARM_RESET)	32	R	0000_0000h	43.8.9/2763
20E_C028	DMA Request Error Register (SDMAARM_EVTERR)	32	R	0000_0000h	43.8.10/ 2764
20E_C02C	Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)	32	R/W	0000_0000h	43.8.11/ 2764
20E_C030	Schedule Status (SDMAARM_PSW)	32	R	0000_0000h	43.8.12/ 2765
20E_C034	DMA Request Error Register (SDMAARM_EVTERRDBG)	32	R	0000_0000h	43.8.13/ 2765
20E_C038	Configuration Register (SDMAARM_CONFIG)	32	R/W	0000_0003h	43.8.14/ 2766
20E_C03C	SDMA LOCK (SDMAARM_SDMA_LOCK)	32	R/W	0000_0000h	43.8.15/ 2767
20E_C040	OnCE Enable (SDMAARM_ONCE_ENB)	32	R/W	0000_0000h	43.8.16/ 2768
20E_C044	OnCE Data Register (SDMAARM_ONCE_DATA)	32	R/W	0000_0000h	43.8.17/ 2769
20E_C048	OnCE Instruction Register (SDMAARM_ONCE_INSTR)	32	R/W	0000_0000h	43.8.18/ 2769
20E_C04C	OnCE Status Register (SDMAARM_ONCE_STAT)	32	R	0000_E000h	43.8.19/ 2769
20E_C050	OnCE Command Register (SDMAARM_ONCE_CMD)	32	R/W	0000_0000h	43.8.20/ 2771

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C058	Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)	32	R/W	0000_0001h	43.8.21/ 2772
20E_C05C	Channel 0 Boot Address (SDMAARM_CHN0ADDR)	32	R/W	0000_0050h	43.8.22/ 2772
20E_C060	DMA Requests (SDMAARM_EVT_MIRROR)	32	R	0000_0000h	43.8.23/ 2773
20E_C064	DMA Requests 2 (SDMAARM_EVT_MIRROR2)	32	R	0000_0000h	43.8.24/ 2773
20E_C070	Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)	32	R/W	0000_0000h	43.8.25/ 2774
20E_C074	Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)	32	R/W	0000_0000h	43.8.26/ 2776
20E_C100	Channel Priority Registers (SDMAARM_SDMA_CHNPRI0)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C104	Channel Priority Registers (SDMAARM_SDMA_CHNPRI1)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C108	Channel Priority Registers (SDMAARM_SDMA_CHNPRI2)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C10C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI3)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C110	Channel Priority Registers (SDMAARM_SDMA_CHNPRI4)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C114	Channel Priority Registers (SDMAARM_SDMA_CHNPRI5)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C118	Channel Priority Registers (SDMAARM_SDMA_CHNPRI6)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C11C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI7)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C120	Channel Priority Registers (SDMAARM_SDMA_CHNPRI8)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C124	Channel Priority Registers (SDMAARM_SDMA_CHNPRI9)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C128	Channel Priority Registers (SDMAARM_SDMA_CHNPRI10)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C12C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI11)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C130	Channel Priority Registers (SDMAARM_SDMA_CHNPRI12)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C134	Channel Priority Registers (SDMAARM_SDMA_CHNPRI13)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C138	Channel Priority Registers (SDMAARM_SDMA_CHNPRI14)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C13C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI15)	32	R/W	0000_0000h	43.8.27/ 2777

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C140	Channel Priority Registers (SDMAARM_SDMA_CHNPRI16)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C144	Channel Priority Registers (SDMAARM_SDMA_CHNPRI17)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C148	Channel Priority Registers (SDMAARM_SDMA_CHNPRI18)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C14C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI19)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C150	Channel Priority Registers (SDMAARM_SDMA_CHNPRI20)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C154	Channel Priority Registers (SDMAARM_SDMA_CHNPRI21)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C158	Channel Priority Registers (SDMAARM_SDMA_CHNPRI22)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C15C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI23)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C160	Channel Priority Registers (SDMAARM_SDMA_CHNPRI24)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C164	Channel Priority Registers (SDMAARM_SDMA_CHNPRI25)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C168	Channel Priority Registers (SDMAARM_SDMA_CHNPRI26)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C16C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI27)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C170	Channel Priority Registers (SDMAARM_SDMA_CHNPRI28)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C174	Channel Priority Registers (SDMAARM_SDMA_CHNPRI29)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C178	Channel Priority Registers (SDMAARM_SDMA_CHNPRI30)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C17C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI31)	32	R/W	0000_0000h	43.8.27/ 2777
20E_C200	Channel Enable RAM (SDMAARM_CHNENBL0)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C204	Channel Enable RAM (SDMAARM_CHNENBL1)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C208	Channel Enable RAM (SDMAARM_CHNENBL2)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C20C	Channel Enable RAM (SDMAARM_CHNENBL3)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C210	Channel Enable RAM (SDMAARM_CHNENBL4)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C214	Channel Enable RAM (SDMAARM_CHNENBL5)	32	R/W	0000_0000h	43.8.28/ 2777

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C218	Channel Enable RAM (SDMAARM_CHNENBL6)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C21C	Channel Enable RAM (SDMAARM_CHNENBL7)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C220	Channel Enable RAM (SDMAARM_CHNENBL8)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C224	Channel Enable RAM (SDMAARM_CHNENBL9)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C228	Channel Enable RAM (SDMAARM_CHNENBL10)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C22C	Channel Enable RAM (SDMAARM_CHNENBL11)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C230	Channel Enable RAM (SDMAARM_CHNENBL12)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C234	Channel Enable RAM (SDMAARM_CHNENBL13)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C238	Channel Enable RAM (SDMAARM_CHNENBL14)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C23C	Channel Enable RAM (SDMAARM_CHNENBL15)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C240	Channel Enable RAM (SDMAARM_CHNENBL16)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C244	Channel Enable RAM (SDMAARM_CHNENBL17)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C248	Channel Enable RAM (SDMAARM_CHNENBL18)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C24C	Channel Enable RAM (SDMAARM_CHNENBL19)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C250	Channel Enable RAM (SDMAARM_CHNENBL20)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C254	Channel Enable RAM (SDMAARM_CHNENBL21)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C258	Channel Enable RAM (SDMAARM_CHNENBL22)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C25C	Channel Enable RAM (SDMAARM_CHNENBL23)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C260	Channel Enable RAM (SDMAARM_CHNENBL24)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C264	Channel Enable RAM (SDMAARM_CHNENBL25)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C268	Channel Enable RAM (SDMAARM_CHNENBL26)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C26C	Channel Enable RAM (SDMAARM_CHNENBL27)	32	R/W	0000_0000h	43.8.28/ 2777

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C270	Channel Enable RAM (SDMAARM_CHNENBL28)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C274	Channel Enable RAM (SDMAARM_CHNENBL29)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C278	Channel Enable RAM (SDMAARM_CHNENBL30)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C27C	Channel Enable RAM (SDMAARM_CHNENBL31)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C280	Channel Enable RAM (SDMAARM_CHNENBL32)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C284	Channel Enable RAM (SDMAARM_CHNENBL33)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C288	Channel Enable RAM (SDMAARM_CHNENBL34)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C28C	Channel Enable RAM (SDMAARM_CHNENBL35)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C290	Channel Enable RAM (SDMAARM_CHNENBL36)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C294	Channel Enable RAM (SDMAARM_CHNENBL37)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C298	Channel Enable RAM (SDMAARM_CHNENBL38)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C29C	Channel Enable RAM (SDMAARM_CHNENBL39)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2A0	Channel Enable RAM (SDMAARM_CHNENBL40)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2A4	Channel Enable RAM (SDMAARM_CHNENBL41)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2A8	Channel Enable RAM (SDMAARM_CHNENBL42)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2AC	Channel Enable RAM (SDMAARM_CHNENBL43)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2B0	Channel Enable RAM (SDMAARM_CHNENBL44)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2B4	Channel Enable RAM (SDMAARM_CHNENBL45)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2B8	Channel Enable RAM (SDMAARM_CHNENBL46)	32	R/W	0000_0000h	43.8.28/ 2777
20E_C2BC	Channel Enable RAM (SDMAARM_CHNENBL47)	32	R/W	0000_0000h	43.8.28/ 2777

43.8.1 ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)

Address: 20E_C000h base + 0h offset = 20E_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MC0PTR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_MC0PTR field descriptions

Field	Description
MC0PTR	**Channel 0 Pointer** contains the 32-bit address, in ARM platform memory, of channel 0 control block (the boot channel). Appendix A fully describes the SDMA Application Programming Interface (API). The ARM platform has a read/write access and the SDMA has a read-only access.

43.8.2 Channel Interrupts (SDMAARM_INTR)

Address: 20E_C000h base + 4h offset = 20E_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HI[31:0]																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_INTR field descriptions

Field	Description
HI[31:0]	The ARM platform Interrupts register contains the 32 HI[i] bits. If any bit is set, it will cause an interrupt to the ARM platform. This register is a "write-ones" register to the ARM platform. When the ARM platform sets a bit in this register the corresponding HI[i] bit is cleared. The interrupt service routine should clear individual channel bits when their interrupts are serviced, failure to do so will cause continuous interrupts. The SDMA is responsible for setting the HI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

43.8.3 Channel Stop/Channel Status (SDMAARM_STOP_STAT)

Address: 20E_C000h base + 8h offset = 20E_C008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_STOP_STAT field descriptions

Field	Description
HE	This 32-bit register gives access to the ARM platform Enable bits. There is one bit for every channel. This register is a "write-ones" register to the ARM platform. When the ARM platform writes 1 in bit i of this register, it clears the HE[i] and HSTART[i] bits. Reading this register yields the current state of the HE[i] bits.

43.8.4 Channel Start (SDMAARM_HSTART)

Address: 20E_C000h base + Ch offset = 20E_C00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HSTART_HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_HSTART field descriptions

Field	Description
HSTART_HE	The HSTART_HE registers are 32 bits wide with one bit for every channel. When a bit is written to 1, it enables the corresponding channel. Two physical registers are accessed with that address (HSTART and HE), which enables the ARM platform to trigger a channel a second time before the first trigger is processed. • This register is a "write-ones" register to the ARM platform. Neither HSTART[i] bit can be set while the corresponding HE[i] bit is cleared. • When the ARM platform tries to set the HSTART[i] bit by writing a one (if the corresponding HE[i] bit is clear), the bit in the HSTART[i] register will remain cleared and the HE[i] bit will be set. • If the corresponding HE[i] bit was already set, the HSTART[i] bit will be set. The next time the SDMA channel *i* attempts to clear the HE[i] bit by means of a `done` instruction, the bit in the HSTART[i] register will be cleared and the HE[i] bit will take the old value of the HSTART[i] bit. • Reading this register yields the current state of the HSTART[i] bits. This mechanism enables the ARM platform to pipeline two HSTART commands per channel.

43.8.5 Channel Event Override (SDMAARM_EVTOVR)

Address: 20E_C000h base + 10h offset = 20E_C010h

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

SDMAARM_EVTOVR field descriptions

Field	Description
EO	The Channel Event Override register contains the 32 EO[i] bits. A bit set in this register causes the SDMA to ignore DMA requests when scheduling the corresponding channel.

43.8.6 Channel BP Override (SDMAARM_DSPOVR)

Address: 20E_C000h base + 14h offset = 20E_C014h

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

SDMAARM_DSPOVR field descriptions

Field	Description
DO	This register is reserved. All DO bits should be set to the reset value of 1. A setting of 0 will prevent SDMA channels from starting according to the condition described in Runnable Channels Evaluation. 0 - Reserved 1 - Reset value.

43.8.7 Channel ARM platform Override (SDMAARM_HOSTOVR)

Address: 20E_C000h base + 18h offset = 20E_C018h

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

SDMAARM_HOSTOVR field descriptions

Field	Description
HO	The Channel ARM platform Override register contains the 32 HO[i] bits. A bit set in this register causes the SDMA to ignore the ARM platform enable bit (HE) when scheduling the corresponding channel.

43.8.8 Channel Event Pending (SDMAARM_EVTPEND)

Address: 20E_C000h base + 1Ch offset = 20E_C01Ch

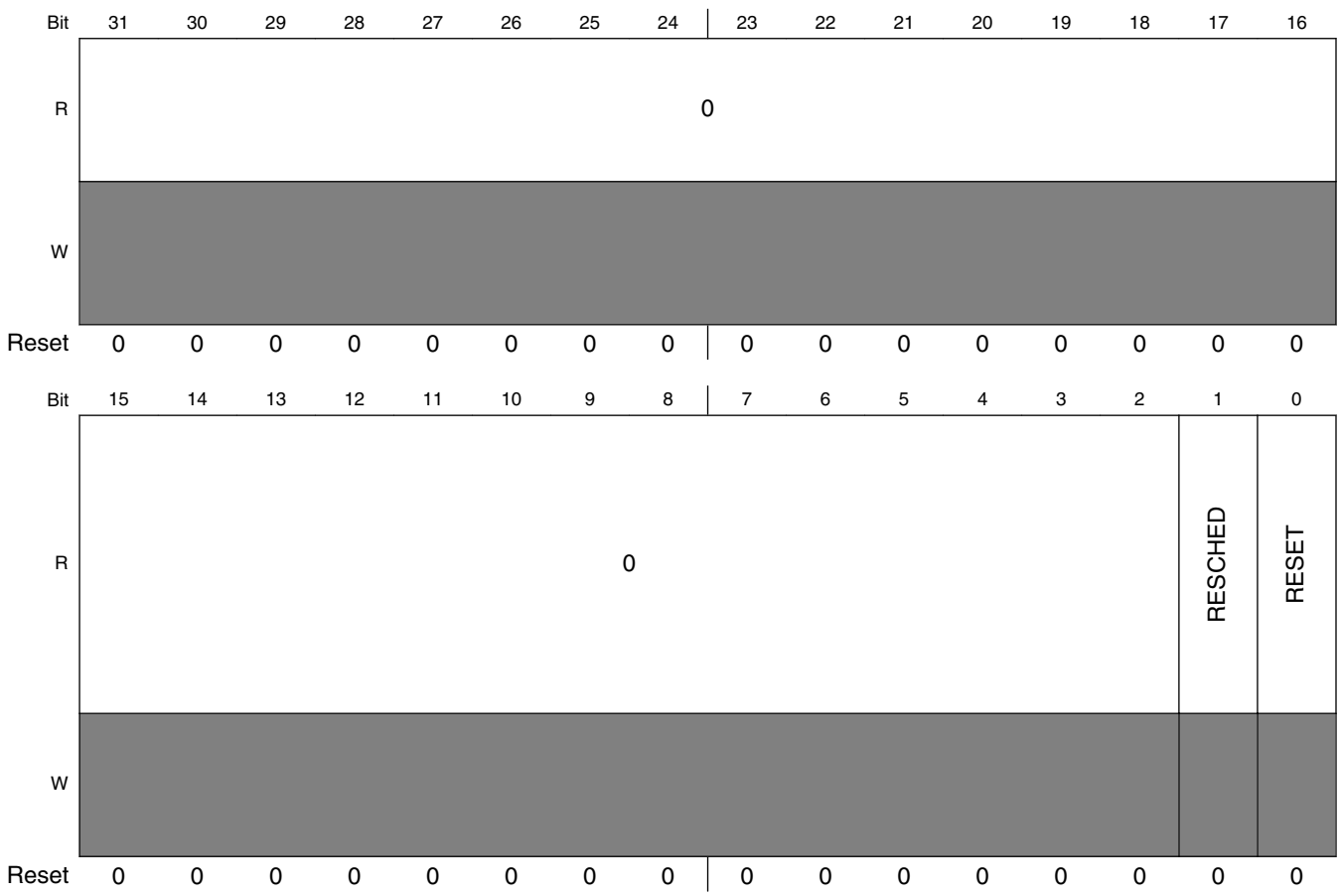
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EP																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTPEND field descriptions

Field	Description
EP	The Channel Event Pending register contains the 32 EP[i] bits. Reading this register enables the ARM platform to determine what channels are pending after the reception of a DMA request. • Setting a bit in this register causes the SDMA to reevaluate scheduling as if a DMA request mapped on this channel had occurred. This is useful for starting up channels, so that initialization is done before awaiting the first request. The scheduler can also set bits in the EVTPEND register according to the received DMA requests. • The EP[i] bit may be cleared by the `done` instruction when running the channel *i* script. This a "write-ones" mechanism: Writing a '0' does not clear the corresponding bit.

43.8.9 Reset Register (SDMAARM_RESET)

Address: 20E_C000h base + 24h offset = 20E_C024h



SDMAARM_RESET field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

SDMAARM_RESET field descriptions (continued)

Field	Description
1 RESCHED	When set, this bit forces the SDMA to reschedule as if a script had executed a `done` instruction. This enables the ARM platform to recover from a runaway script on a channel by clearing its HE[i] bit via the STOP register, and then forcing a reschedule via the RESCHED bit. The RESCHED bit is cleared when the context switch starts.
0 RESET	When set, this bit causes the SDMA to be held in a software reset. The internal reset signal is held low 16 cycles; the RESET bit is automatically cleared when the internal reset signal rises.

43.8.10 DMA Request Error Register (SDMAARM_EVTERR)

Address: 20E_C000h base + 28h offset = 20E_C028h

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

SDMAARM_EVTERR field descriptions

Field	Description
CHNERR	This register is used by the SDMA to warn the ARM platform when an incoming DMA request was detected and it triggers a channel that is already pending or being serviced. This probably means there is an overflow of data for that channel. • An interrupt is sent to the ARM platform if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the ARM platform or during SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set; the EVTERR[i] bit is unaffected if the ARM platform tries to set the EP[i] bit, whereas, that EP[i] bit is already set.

43.8.11 Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)

Address: 20E_C000h base + 2Ch offset = 20E_C02Ch

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

SDMAARM_INTRMASK field descriptions

Field	Description
HIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit HIMASK[i] is set, the HI[i] bit is set and an interrupt is sent to the ARM platform when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

43.8.12 Schedule Status (SDMAARM_PSW)

Address: 20E_C000h base + 30h offset = 20E_C030h

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

SDMAARM_PSW field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–13 NCP[2:0]	The Next Channel Priority gives the next pending channel priority. When the priority is 0, it means there is no pending channel and the NCR value has no meaning. 0 No running channel 1 Active channel priority
12–8 NCR[4:0]	The Next Channel Register indicates the number of the next scheduled pending channel with the highest priority.
7–4 CCP[2:0]	The Current Channel Priority indicates the priority of the current active channel. When the priority is 0, no channel is running: The SDMA is idle and the CCR value has no meaning. In the case that the SDMA has finished running the channel and has entered sleep state, CCP will indicate the priority of previous running channel. 0 No running channel 1 Active channel priority
CCR[4:0]	The Current Channel Register indicates the number of the channel that is being executed by the SDMA. SDMA. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel.

43.8.13 DMA Request Error Register (SDMAARM_EVTERRDBG)

Address: 20E_C000h base + 34h offset = 20E_C034h

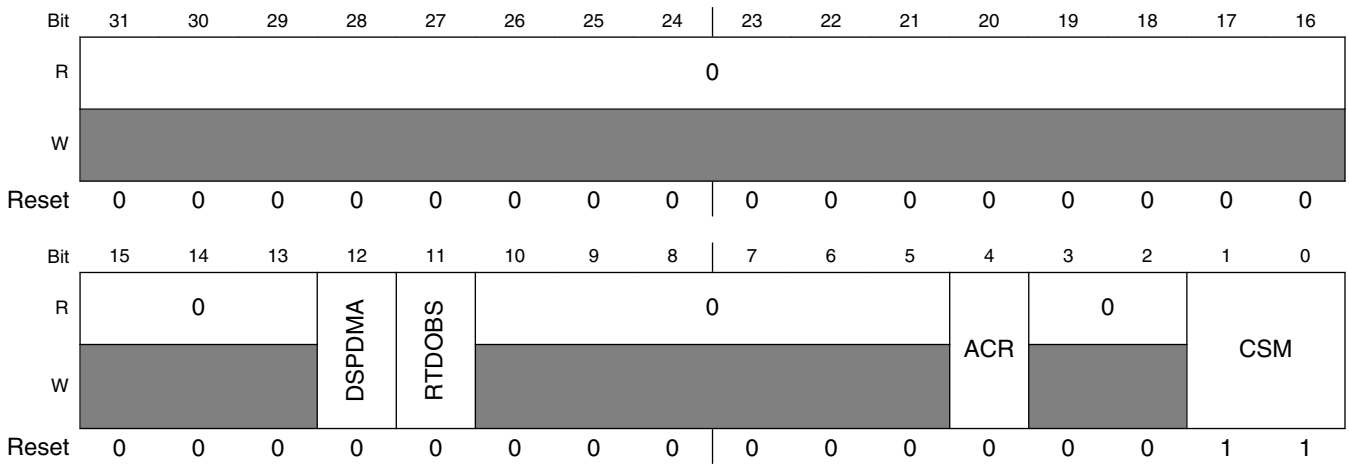
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTERRDBG field descriptions

Field	Description
CHNERR	This register is the same as EVTERR, except reading it does not clear its contents. This address is meant to be used in debug mode. The ARM platform OnCE may check this register value without modifying it.

43.8.14 Configuration Register (SDMAARM_CONFIG)

Address: 20E_C000h base + 38h offset = 20E_C038h



SDMAARM_CONFIG field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 DSPDMA	This bit's function is reserved and should be configured as zero. 0 - Reset Value 1 - Reserved
11 RTDOBS	Indicates if Real-Time Debug pins are used: They do not toggle by default in order to reduce power consumption. 0 RTD pins disabled 1 RTD pins enabled
10–5 Reserved	This read-only field is reserved and always has the value 0.
4 ACR	ARM platform DMA / SDMA Core Clock Ratio. Selects the clock ratio between ARM platform DMA interfaces (burst DMA and peripheral DMA) and the internal SDMA core clock. The frequency selection is determined separately by the chip clock controller. This bit has to match the configuration of the chip clock controller that generates the clocks used in the SDMA. 0 ARM platform DMA interface frequency equals twice core frequency 1 ARM platform DMA interface frequency equals core frequency
3–2 Reserved	This read-only field is reserved and always has the value 0.

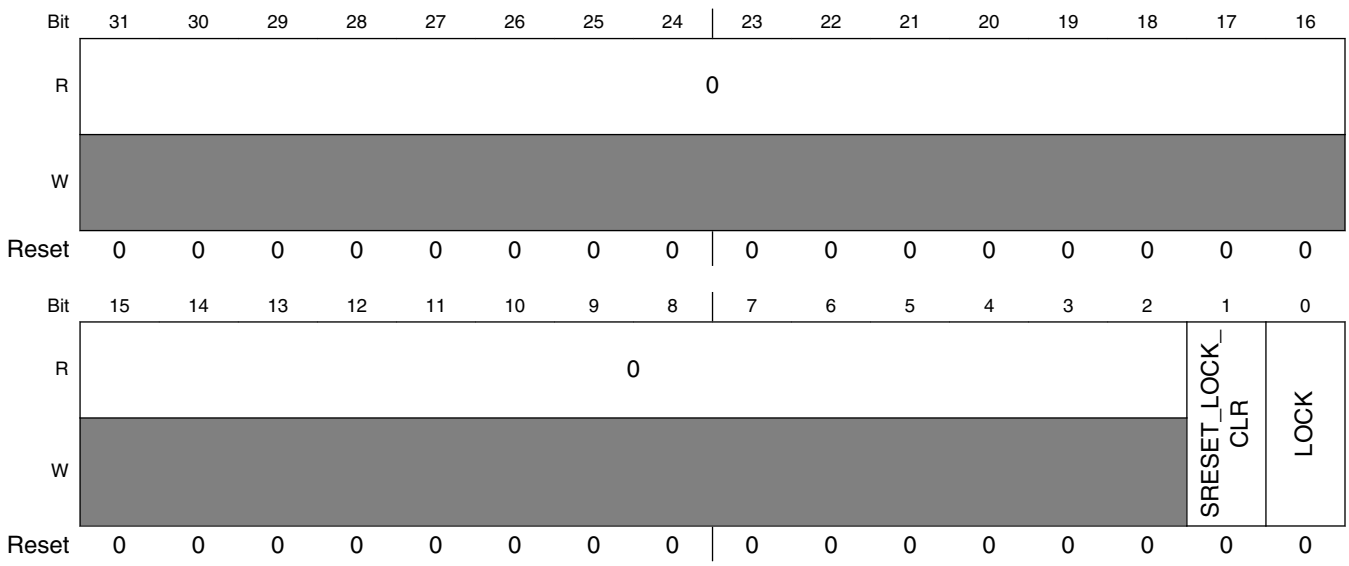
Table continues on the next page...

SDMAARM_CONFIG field descriptions (continued)

Field	Description
CSM	Selects the Context Switch Mode. The ARM platform has a read/write access. The SDMA cannot modify that register. The value at reset is 3, which selects the dynamic context switch by default. That register can be modified at anytime but the new context switch configuration will only be taken into account at the start of the next restore phase. NOTE: The first call to SDMA's channel 0 Bootload script after reset should use static context switch mode to ensure the context RAM for channel 0 is initialized in the channel SAVE Phase. After Channel 0 is run once, then any of the dynamic context modes can be used. 0 static 1 dynamic low power 2 dynamic with no loop 3 dynamic

43.8.15 SDMA LOCK (SDMAARM_SDMA_LOCK)

Address: 20E_C000h base + 3Ch offset = 20E_C03Ch



SDMAARM_SDMA_LOCK field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 SRESET_LOCK_CLR	The SRESET_LOCK_CLR bit determine if the LOCK bit is cleared on a software reset triggered by writing to the RESET register. This bit cannot be changed if LOCK=1. SREST_LOCK_CLR is cleared by conditions that clear the LOCK bit. 0 Software Reset does not clear the LOCK bit. 1 Software Reset clears the LOCK bit.

Table continues on the next page...

SDMAARM_SDMA_LOCK field descriptions (continued)

Field	Description
0 LOCK	The LOCK bit is used to restrict access to update SDMA script memory through ROM channel zero scripts and through the OnCE interface under ARM platform control. The LOCK bit is set: • The SDMA_LOCK, ONCE_ENB,CH0ADDR, and ILLINSTADDR registers cannot be written. These registers can be read, but writes are ignored. • SDMA software executing out of ROM or RAM may check the LOCK bit in the LOCK register Lock Status Register (SDMACORE_SDMA_LOCK) to determine if certain operations are allowed, such as up-loading new scripts. Once the LOCK bit is set to 1, only a reset can clear it. The LOCK bit is cleared by a hardware reset. LOCK is cleared by a software reset only if SRESET_LOCK_CLR is set. 0 LOCK disengaged. 1 LOCK enabled.

43.8.16 OnCE Enable (SDMAARM_ONCE_ENB)

Address: 20E_C000h base + 40h offset = 20E_C040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

SDMAARM_ONCE_ENB field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 ENB	The OnCE Enable register selects the OnCE control source: When cleared (0), the OnCE registers are accessed through the JTAG interface; when set (1), the OnCE registers may be accessed by the ARM platform through the addresses described, as follows. • After reset, the OnCE registers are accessed through the JTAG interface. • Writing a 1 to ENB enables the ARM platform to access the ONCE_* as any other SDMA control register. • When cleared (0), all the ONCE_xxx registers cannot be written. The value of ENB cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

43.8.17 OnCE Data Register (SDMAARM_ONCE_DATA)

Address: 20E_C000h base + 44h offset = 20E_C044h

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

SDMAARM_ONCE_DATA field descriptions

Field	Description
DATA	Data register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

43.8.18 OnCE Instruction Register (SDMAARM_ONCE_INSTR)

Address: 20E_C000h base + 48h offset = 20E_C048h

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

SDMAARM_ONCE_INSTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
INSTR	Instruction register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

43.8.19 OnCE Status Register (SDMAARM_ONCE_STAT)

Address: 20E_C000h base + 4Ch offset = 20E_C04Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0				ECDR		
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_STAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. Its states are as follows: • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: They are used when entering debug mode after reset. The reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 The JTAG interface controls the OnCE. 1 The ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

SDMAARM_ONCE_STAT field descriptions (continued)

Field	Description
ECDR	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one of the EDR bits is set (the meaning of the encoding is given below). The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addra_cond, addrb_cond, and data_cond conditions. The value of those fields is given by the EDR bits. 0 1 matched addra_cond 1 1 matched addrb_cond 2 1 matched data_cond

43.8.20 OnCE Command Register (SDMAARM_ONCE_CMD)

Address: 20E_C000h base + 50h offset = 20E_C050h

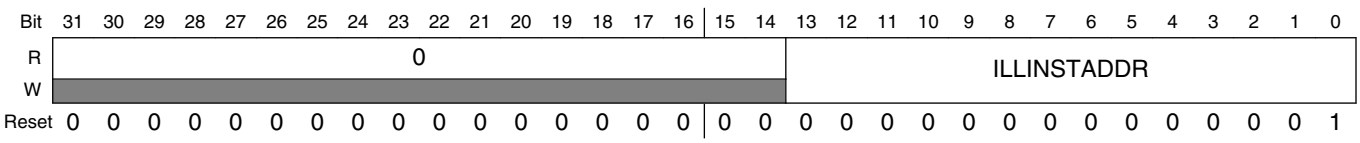
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																												CMD			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_CMD field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value 0.
CMD	Writing to this register will cause the OnCE to execute the command that is written. When needed, the ONCE_DATA and ONCE_INSTR registers should be loaded with the correct value before writing the command to that register. For a list of the OnCE commands and their usage, see OnCE and Real-Time Debug. **NOTE:** 7-15 *reserved* 0 rstatus 1 dmov 2 exec_once 3 run_core 4 exec_core 5 debug_rqst 6 rbuffer

43.8.21 Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)

Address: 20E_C000h base + 58h offset = 20E_C058h

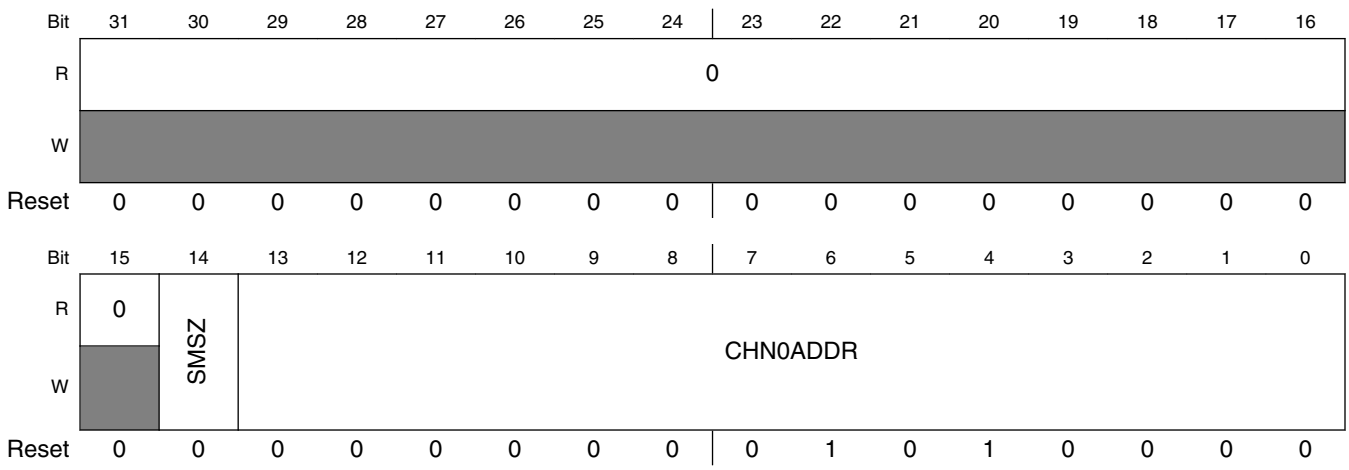


SDMAARM_ILLINSTADDR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
ILLINSTADDR	The Illegal Instruction Trap Address is the address where the SDMA jumps when an illegal instruction is executed. It is 0x0001 after reset. The value of ILLINSTADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

43.8.22 Channel 0 Boot Address (SDMAARM_CHN0ADDR)

Address: 20E_C000h base + 5Ch offset = 20E_C05Ch



SDMAARM_CHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value 0.
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context,

Table continues on the next page...

SDMAARM_CHN0ADDR field descriptions (continued)

Field	Description
	which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. The value of SMSZ cannot be changed if the LOCK bit in the SDMA_LOCK register is set. 0 24 words per context 1 32 words per context
CHN0ADDR	This 14-bit register is used by the boot code of the SDMA. After reset, it points to the standard boot routine in ROM (channel 0 routine). By changing this address, you can perform a boot sequence with your own routine. The very first instructions of the boot code fetch the contents of this register (it is also mapped in the SDMA memory space) and jump to the given address. The reset value is 0x0050 (decimal 80). The value of CHN0ADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

43.8.23 DMA Requests (SDMAARM_EVT_MIRROR)

Address: 20E_C000h base + 60h offset = 20E_C060h

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

SDMAARM_EVT_MIRROR field descriptions

Field	Description
EVENTS	This register reflects the DMA requests received by the SDMA for events 31-0. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of 32 DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR register is cleared following read access. 0 DMA request event not pending 1 DMA request event pending

43.8.24 DMA Requests 2 (SDMAARM_EVT_MIRROR2)

Address: 20E_C000h base + 64h offset = 20E_C064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EVENTS[47:32]															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVT_MIRROR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EVENTS[47:32]	This register reflects the DMA requests received by the SDMA for events 47-32. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR2 register is cleared following read access. 0 - DMA request event not pending 1- DMA request event pending

43.8.25 Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)

Address: 20E_C000h base + 70h offset = 20E_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CNF3	NUM3[5:0]						0	CNF2	NUM2[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CNF1	NUM1[5:0]						0	CNF0	NUM0[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_XTRIG_CONF1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30 CNF3	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by the reception of a DMA request or by the starting of a channel script execution. 0 channel 1 DMA request
29–24 NUM3[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value 0.

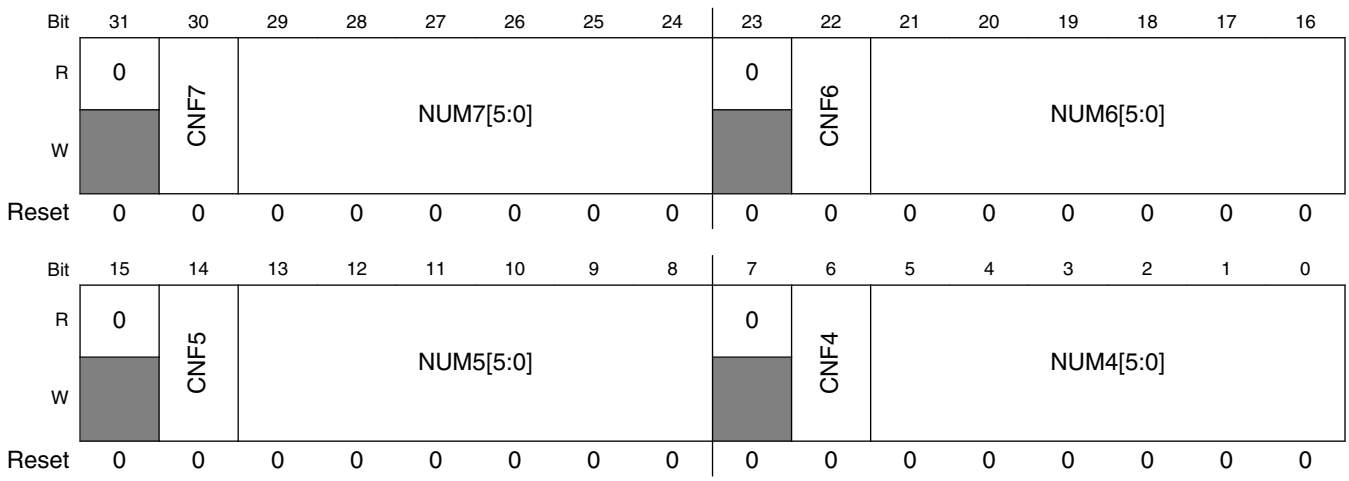
Table continues on the next page...

SDMAARM_XTRIG_CONF1 field descriptions (continued)

Field	Description
22 CNF2	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM2[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value 0.
14 CNF1	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
13–8 NUM1[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value 0.
6 CNF0	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
NUM0[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

43.8.26 Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)

Address: 20E_C000h base + 74h offset = 20E_C074h



SDMAARM_XTRIG_CONF2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30 CNF7	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
29–24 NUM7[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value 0.
22 CNF6	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM6[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value 0.
14 CNF5	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution

Table continues on the next page...

SDMAARM_XTRIG_CONF2 field descriptions (continued)

Field	Description
	0 channel 1 DMA request
13–8 NUM5[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value 0.
6 CNF4	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
NUM4[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

43.8.27 Channel Priority Registers (SDMAARM_SDMA_CHNPRI*n*)

Address: 20E_C000h base + 100h offset + (4d × i), where i=0d to 31d

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

SDMAARM_SDMA_CHNPRI*n* field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
CHNPRIn	This contains the priority of channel number *n*. Useful values are between 1 and 7; 0 is reserved by the SDMA hardware to determine when there is no pending channel. Reset value is 0, which prevents the channels from starting.

43.8.28 Channel Enable RAM (SDMAARM_CHNENBL*n*)

Address: 20E_C000h base + 200h offset + (4d × i), where i=0d to 47d

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

SDMAARM_CHNENBL*n* field descriptions

Field	Description
ENBLn	This 32-bit value selects the channels that are triggered by the DMA request number *n*. If ENBLn[i] is set to 1, bit EP[i] will be set when the DMA request *n* is received. These 48 32-bit registers are physically located in a RAM, with no known reset value. It is thus essential for the ARM platform to program them before any DMA request is triggered to the SDMA, otherwise an unpredictable combination of channels may be started.

43.9 BP Memory Map and Control Register Definitions

The following section describes SDMA control registers available to the BP.

NOTE

These registers are physically implemented in all platforms, but are not accessible when the SDMA BP control port is not connected. Reset values are calculated to allow the system to work when those registers cannot be accessed.

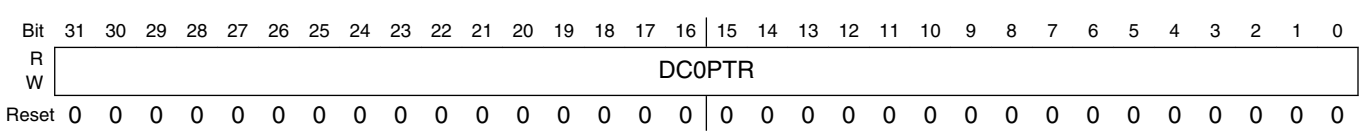
All registers are clocked with the SDMA clock (which means the SDMA clock must be running when the BP wants to access any register).

SDMABP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C000	Channel 0 Pointer (SDMABP_DC0PTR)	32	R/W	0000_0000h	43.9.1/2778
20E_C004	Channel Interrupts (SDMABP_INTR)	32	w1c	0000_0000h	43.9.2/2779
20E_C008	Channel Stop/Channel Status (SDMABP_STOP_STAT)	32	R/W	0000_0000h	43.9.3/2779
20E_C00C	Channel Start (SDMABP_DSTART)	32	R	0000_0000h	43.9.4/2780
20E_C028	DMA Request Error Register (SDMABP_EVTERR)	32	R	0000_0000h	43.9.5/2780
20E_C02C	Channel DSP Interrupt Mask (SDMABP_INTRMASK)	32	R/W	0000_0000h	43.9.6/2781
20E_C034	DMA Request Error Register (SDMABP_EVTERRDBG)	32	R	0000_0000h	43.9.7/2781

43.9.1 Channel 0 Pointer (SDMABP_DC0PTR)

Address: 20E_C000h base + 0h offset = 20E_C000h



SDMABP_DC0PTR field descriptions

Field	Description
DC0PTR	**Channel 0 Pointer** contains the 32-bit address, in BP memory, of the array of channel control blocks starting with the one for channel 0 (the control channel). This register should be initialized by the BP before it enables a channel (for example, channel 0). See the API document SDMA Scripts User Manual for the use of this register. The BP has a read/write access and the SDMA has a read-only access.

43.9.2 Channel Interrupts (SDMABP_INTR)

Address: 20E_C000h base + 4h offset = 20E_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DI																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_INTR field descriptions

Field	Description
DI	The BP Interrupts register contains the 32 DI[i] bits. If any bit is set, it will cause an interrupt to the BP. • This register is a "write-ones" register to the BP. When the BP sets a bit in this register, the corresponding DI[i] bit is cleared. • The interrupt service routine should clear individual channel bits when their interrupts are serviced; failure to do so will cause continuous interrupts. • The SDMA is responsible for setting the DI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

43.9.3 Channel Stop/Channel Status (SDMABP_STOP_STAT)

Address: 20E_C000h base + 8h offset = 20E_C008h

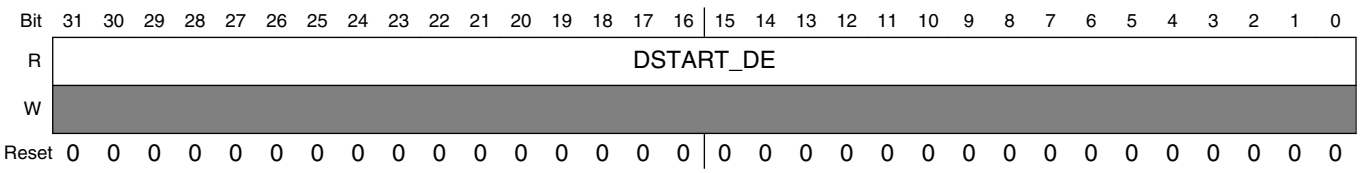
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_STOP_STAT field descriptions

Field	Description
DE	This 32-bit register gives access to the BP (DSP) Enable bits, DE. There is one bit for every channel. • This register is a "write-ones" register to the BP. • When the BP writes 1 in bit *i* of this register, it clears the DE[i] and DSTART[i] bits. • Reading this register yields the current state of the DE[i] bits.

43.9.4 Channel Start (SDMABP_DSTART)

Address: 20E_C000h base + Ch offset = 20E_C00Ch

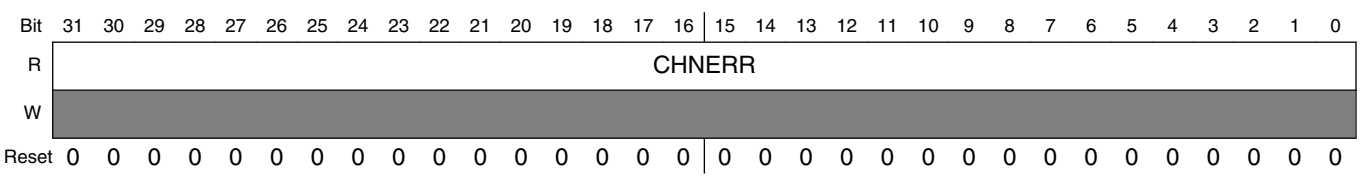


SDMABP_DSTART field descriptions

Field	Description
DSTART_DE	The DSTART_DE registers are 32 bits wide with one bit for every channel. • When a bit is written to 1, it enables the corresponding channel. • Two physical registers are accessed with that address (DSTART and DE), which enables the BP to trigger a channel a second time before the first trigger was processed. • This register is a "write-ones" register to the BP. Neither DSTART[i] bit can be set while the corresponding DE[i] bit is cleared. • When the BP tries to set the DSTART[i] bit by writing a one (if the corresponding DE[i] bit is clear), the bit in the DSTART[i] register will remain cleared and the DE[i] bit will be set. If the corresponding DE[i] bit was already set, the DSTART[i] bit will be set. • The next time the SDMA channel *i* attempts to clear the DE[i] bit by means of a `done` instruction, the bit in the DSTART[i] register will be cleared and the DE[i] bit will take the old value of the DSTART[i] bit. • Reading this register yields the current state of the DSTART[i] bits. This mechanism enables the BP to pipeline two DSTART commands per channel.

43.9.5 DMA Request Error Register (SDMABP_EVTERR)

Address: 20E_C000h base + 28h offset = 20E_C028h



SDMABP_EVTERR field descriptions

Field	Description
CHNERR	This register is used by the SDMA to warn the BP when an incoming DMA request was detected; it then triggers a channel that is already pending or being serviced, which may mean there is an overflow of data for that channel. An interrupt is sent to the BP if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the BP or during an SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set. The EVTERR[i] bit is unaffected if the BP tries to set the EP[i] bit when that EP[i] bit is already set.

43.9.6 Channel DSP Interrupt Mask (SDMABP_INTRMASK)

Address: 20E_C000h base + 2Ch offset = 20E_C02Ch

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

SDMABP_INTRMASK field descriptions

Field	Description
DIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit DIMASK[i] is set, the DI[i] bit is set and an interrupt is sent to the BP when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

43.9.7 DMA Request Error Register (SDMABP_EVTERRDBG)

Address: 20E_C000h base + 34h offset = 20E_C034h

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

SDMABP_EVTERRDBG field descriptions

Field	Description
CHNERR	This register is the same as EVTERR except reading it does not clear its contents. This address is meant to be used in debug mode. The BP OnCE may check this register value without modifying it.

43.10 SDMA Internal (Core) Memory Map and Internal Register Definitions

The actual SDMA memory mapped registers are summarized in the following sections; for peripherals' memory maps, refer to the respective chapters.

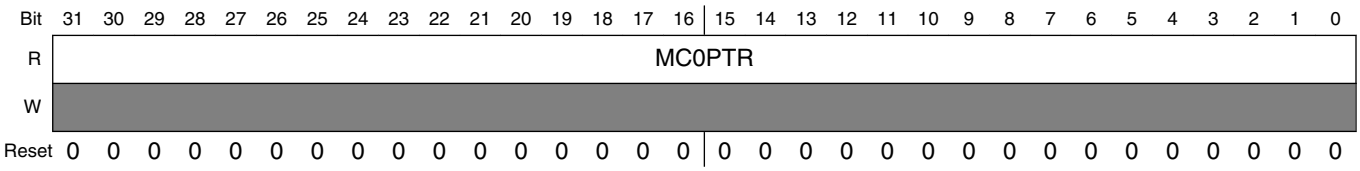
The following definitions serve as a key for the SDMA internal register summary.

SDMACORE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C000	ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)	32	R	0000_0000h	43.10.1/ 2783
20E_C002	Current Channel Pointer (SDMACORE_CCPTR)	32	R	0000_0000h	43.10.2/ 2783
20E_C003	Current Channel Register (SDMACORE_CCR)	32	R	0000_0000h	43.10.3/ 2783
20E_C004	Highest Pending Channel Register (SDMACORE_NCR)	32	R	0000_0000h	43.10.4/ 2784
20E_C005	External DMA Requests Mirror (SDMACORE_EVENTS)	32	R	0000_0000h	43.10.5/ 2785
20E_C006	Current Channel Priority (SDMACORE_CCPRI)	32	R	0000_0000h	43.10.6/ 2786
20E_C007	Next Channel Priority (SDMACORE_NCPRI)	32	R	0000_0000h	43.10.7/ 2786
20E_C009	OnCE Event Cell Counter (SDMACORE_ECOUNT)	32	R/W	0000_0000h	43.10.8/ 2787
20E_C00A	OnCE Event Cell Control Register (SDMACORE_ECTL)	32	R/W	0000_0000h	43.10.9/ 2787
20E_C00B	OnCE Event Address Register A (SDMACORE_EAA)	32	R/W	0000_0000h	43.10.10/ 2789
20E_C00C	OnCE Event Cell Address Register B (SDMACORE_EAB)	32	R/W	0000_0000h	43.10.11/ 2789
20E_C00D	OnCE Event Cell Address Mask (SDMACORE_EAM)	32	R/W	0000_0000h	43.10.12/ 2789
20E_C00E	OnCE Event Cell Data Register (SDMACORE_ED)	32	R/W	0000_0000h	43.10.13/ 2790
20E_C00F	OnCE Event Cell Data Mask (SDMACORE_EDM)	32	R/W	0000_0000h	43.10.14/ 2790
20E_C018	OnCE Real-Time Buffer (SDMACORE_RTB)	32	R/W	0000_0000h	43.10.15/ 2791
20E_C019	OnCE Trace Buffer (SDMACORE_TB)	32	R	0000_0000h	43.10.16/ 2791
20E_C01A	OnCE Status (SDMACORE_OSTAT)	32	R	0000_0000h	43.10.17/ 2792
20E_C01C	Channel 0 Boot Address (SDMACORE_MCHN0ADDR)	32	R	0000_0000h	43.10.18/ 2794
20E_C01D	ENDIAN Status Register (SDMACORE_ENDIANNESS)	32	R	0000_0001h	43.10.19/ 2795
20E_C01E	Lock Status Register (SDMACORE_SDMA_LOCK)	32	R	0000_0000h	43.10.20/ 2796
20E_C01F	External DMA Requests Mirror #2 (SDMACORE_EVENTS2)	32	R	0000_0000h	43.10.21/ 2797

43.10.1 ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)

Address: 20E_C000h base + 0h offset = 20E_C000h

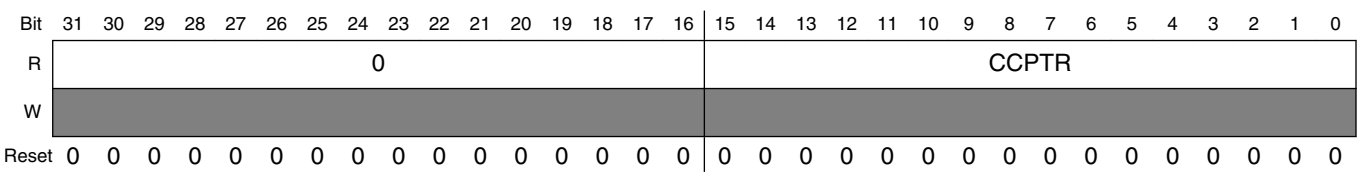


SDMACORE_MC0PTR field descriptions

Field	Description
MC0PTR	Contains the address-in the ARM platform memory space-of the initial SDMA context and scripts that are loaded by the SDMA boot script running on channel 0.

43.10.2 Current Channel Pointer (SDMACORE_CCPTR)

Address: 20E_C000h base + 2h offset = 20E_C002h

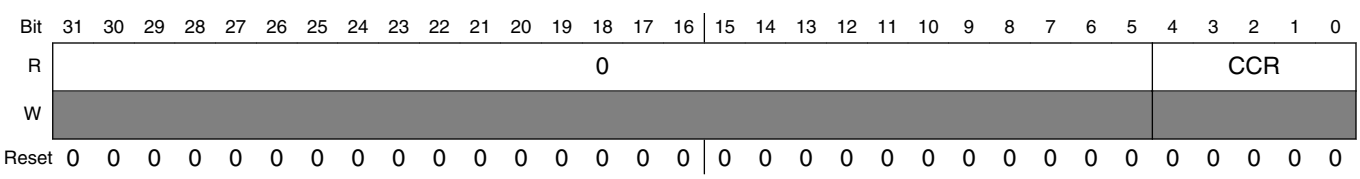


SDMACORE_CCPTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
CCPTR	Contains the start address of the context data for the current channel: Its value is *CONTEXT_BASE* + 24* *CCR* or *CONTEXT_BASE* + 32* *CCR* where CONTEXT_BASE = 0x0800. The value 24 or 32 is selected according to the programmed channel scratch RAM size in the register shown in Channel 0 Boot Address (SDMAARM_CHN0ADDR) .

43.10.3 Current Channel Register (SDMACORE_CCR)

Address: 20E_C000h base + 3h offset = 20E_C003h

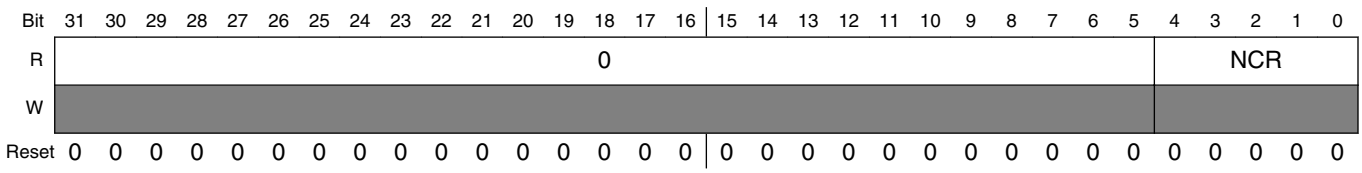


SDMACORE_CCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
CCR	Contains the number of the current running channel whose context is installed. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel. The PST bits in the OSTAT register indicate when the SDMA is in sleep state.

43.10.4 Highest Pending Channel Register (SDMACORE_NCR)

Address: 20E_C000h base + 4h offset = 20E_C004h



SDMACORE_NCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
NCR	Contains the number of the pending channel that the scheduler has selected to run next.

43.10.5 External DMA Requests Mirror (SDMACORE_EVENTS)

NOTE

This register is very useful in the case of DMA requests that are active when a peripheral FIFO level is above the programmed watermark. The activation of the DMA request (rising edge) is detected by the SDMA logic and it can enable one or several channels. One of the channels accesses the peripheral and reads or writes a number of data that matches the watermark level (for example, if the watermark is four words, the channel reads or writes four words).

If the channel is effectively executed long after the DMA request was received, reading or writing the watermark number of data may not be sufficient to reset the DMA request (for example, if the FIFO watermark is four and at the channel execution it already contains nine pieces of data). This means no new rising edge may be detected by the SDMA, although there still remains transfers to perform. Therefore, if the channel were terminated at that time, it would not be restarted, causing potential overrun or underrun of the peripheral.

The proposed mechanism is for the channel to check this register after it has performed the "watermark" number of accesses to the peripheral. If the bit for the DMA request that triggers this channel is set, it means there is still another watermark number of data to transfer. This goes on until the bit is cleared. The same script can be used for multiple channels that require this behavior. The script can determine its channel number from the CCR register and infer the corresponding DMA request bit to check. It needs a reference table that is coherent with the request-channel matrix that the ARM platform programmed.

Address: 20E_C000h base + 5h offset = 20E_C005h

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

SDMACORE_EVENTS field descriptions

Field	Description
EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 0-31. The EVENTS2 register displays events 32-47.

43.10.6 Current Channel Priority (SDMACORE_CCPRI)

Address: 20E_C000h base + 6h offset = 20E_C006h

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

SDMACORE_CCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
CCPRI	Contains the 3-bit priority of the channel whose context is installed. It is 0 when no channel is running. **NOTE:** 1-7 current channel priority 0 no running channel

43.10.7 Next Channel Priority (SDMACORE_NCPRI)

Address: 20E_C000h base + 7h offset = 20E_C007h

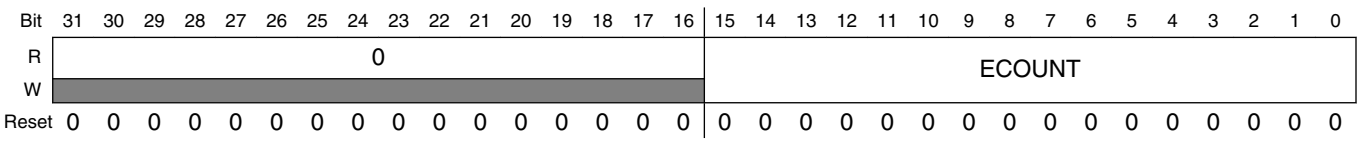
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													NCPRI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_NCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
NCPRI	Contains the 3-bit priority of the channel the scheduler has selected to run next. It is 0 when no other channel is pending.

43.10.8 OnCE Event Cell Counter (SDMACORE_ECOUNT)

Address: 20E_C000h base + 9h offset = 20E_C009h

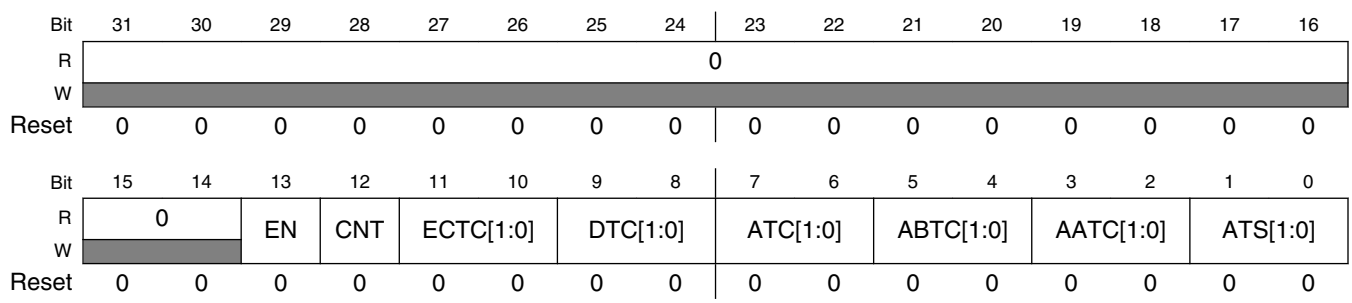


SDMACORE_ECOUNT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
ECOUNT	The event cell counter contains the number of times minus one that an event detection must occur before generating a debug request. • This register should be written before any attempt to use the event detection counter during an event detection process. • The counter is cleared on a JTAG reset.

43.10.9 OnCE Event Cell Control Register (SDMACORE_ECTL)

Address: 20E_C000h base + Ah offset = 20E_C00Ah



SDMACORE_ECTL field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13 EN	Event Cell Enable. If the EN bit is set, the event cell is allowed to generate debug requests (the cell is awakened). If it is cleared, the event detection unit is disabled and no hardware breakpoint is generated, but matching conditions are still reflected on the emulation pin. 0 Cell is disabled. 1 Cell is enabled.
12 CNT	Event Counter Enable. The event counter enable bit determines if the cell counter is used during the event detection. In order to use the event counter during an event detection process, the event cell counter register should be loaded with a value equal to the number of times minus one that an event occurs before

Table continues on the next page...

SDMACORE_ECTL field descriptions (continued)

Field	Description
	a debug request is sent. After every event detection, the counter is decreased. When the counter reaches the value 0, the event detection cell sends a debug request to the core. The event counter register should be written and the EN bit should be set before each new event detection process uses the event counter. 0 Counter is disabled. 1 Counter is enabled.
11–10 ECTC[1:0]	The event cell trigger condition bits select the combination of address and data matching conditions that generate the final address/data condition. During program execution, if this event cell trigger condition goes to 1, a debug request is sent to the SDMA. The EN bit must be set to enable the debug request generation. 00 address ONLY 01 data ONLY 10 address AND data 11 address OR data
9–8 DTC[1:0]	The data trigger condition bits define when data is considered matching after comparison with the data register of the event detection unit. The operations are performed on unsigned values. 00 equal 01 not equal 10 greater than 11 less than
7–6 ATC[1:0]	The address trigger condition bits select how the two address conditions (addressA and addressB) are combined to define the global address matching condition. The supported combinations are described, as follows. 00 addressA ONLY 01 addrA AND addrB 10 addrA OR addrB 11 reserved
5–4 ABTC[1:0]	The Address B Trigger Condition (ABTC) controls the operations performed by address comparator B. All operations are performed on unsigned values. This comparator B outputs the addressB condition. 00 equal 01 not equal 10 greater than 11 less than
3–2 AATC[1:0]	The Address A Trigger Condition (AATC) controls the operations performed by address comparator A. All operations are performed on unsigned values. This comparator A outputs the addressA condition. 00 equal 01 not equal 10 greater than 11 less than
ATS[1:0]	The access type select bits define the memory access type required on the SDMA memory bus. 00 read ONLY 01 write ONLY 10 read or write 11 -

43.10.10 OnCE Event Address Register A (SDMACORE_EAA)

Address: 20E_C000h base + Bh offset = 20E_C00Bh

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

SDMACORE_EAA field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAA	Event Cell Address Register A computes an address A condition. It is cleared on a JTAG reset.

43.10.11 OnCE Event Cell Address Register B (SDMACORE_EAB)

Address: 20E_C000h base + Ch offset = 20E_C00Ch

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

SDMACORE_EAB field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAB	Event Cell Address Register B computes an address B condition. It is cleared on a JTAG reset.

43.10.12 OnCE Event Cell Address Mask (SDMACORE_EAM)

Address: 20E_C000h base + Dh offset = 20E_C00Dh

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

SDMACORE_EAM field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAM	The Event Cell Address Mask contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before performing the address comparison. **NOTE:** There is a common address mask value for both address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The register is cleared on a JTAG reset.

43.10.13 OnCE Event Cell Data Register (SDMACORE_ED)

Address: 20E_C000h base + Eh offset = 20E_C00Eh

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

SDMACORE_ED field descriptions

Field	Description
ED	The event cell data register contains a user defined data value. This data value is an input for the data comparator which generates the data condition. It is cleared on a JTAG reset.

43.10.14 OnCE Event Cell Data Mask (SDMACORE_EDM)

Address: 20E_C000h base + Fh offset = 20E_C00Fh

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

SDMACORE_EDM field descriptions

Field	Description
EDM	The event cell data mask register contains the user-defined data mask value. • This mask is applied to the data value latched from the memory bus before performing the data comparison. • Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. • The data mask is cleared on a JTAG reset.

43.10.15 OnCE Real-Time Buffer (SDMACORE_RTB)

Address: 20E_C000h base + 18h offset = 20E_C018h

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

SDMACORE_RTB field descriptions

Field	Description
RTB	The Real Time Buffer register stores and retrieves run time information without putting the SDMA in debug mode. Writing to that register triggers a pulse on a specific real-time debug pin whose connection depends on the chip implementation. The RTB value can be accessed by the OnCE under ARM platform or JTAG control using the rbuffer command.

43.10.16 OnCE Trace Buffer (SDMACORE_TB)

Address: 20E_C000h base + 19h offset = 20E_C019h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			TBF	TADDR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

SDMACORE_TB field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 TBF	The Trace Buffer Flag is set when the buffer contains the addresses of a valid change of flow. The contents of the buffer should be ignored otherwise. 0 Invalid information 1 Valid information
27–14 TADDR	The target address is the address taken after the execution of the change of flow instruction.
CHFADDR	The change of flow address is the address where the change of flow is taken when executing a change of flow instruction.

43.10.17 OnCE Status (SDMACORE_OSTAT)

Address: 20E_C000h base + 1Ah offset = 20E_C01Ah

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0				ECDR[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_OSTAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel-switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel. They are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow Loop Sleep 12 Debug in Sleep 13 Functional Unit in Sleep

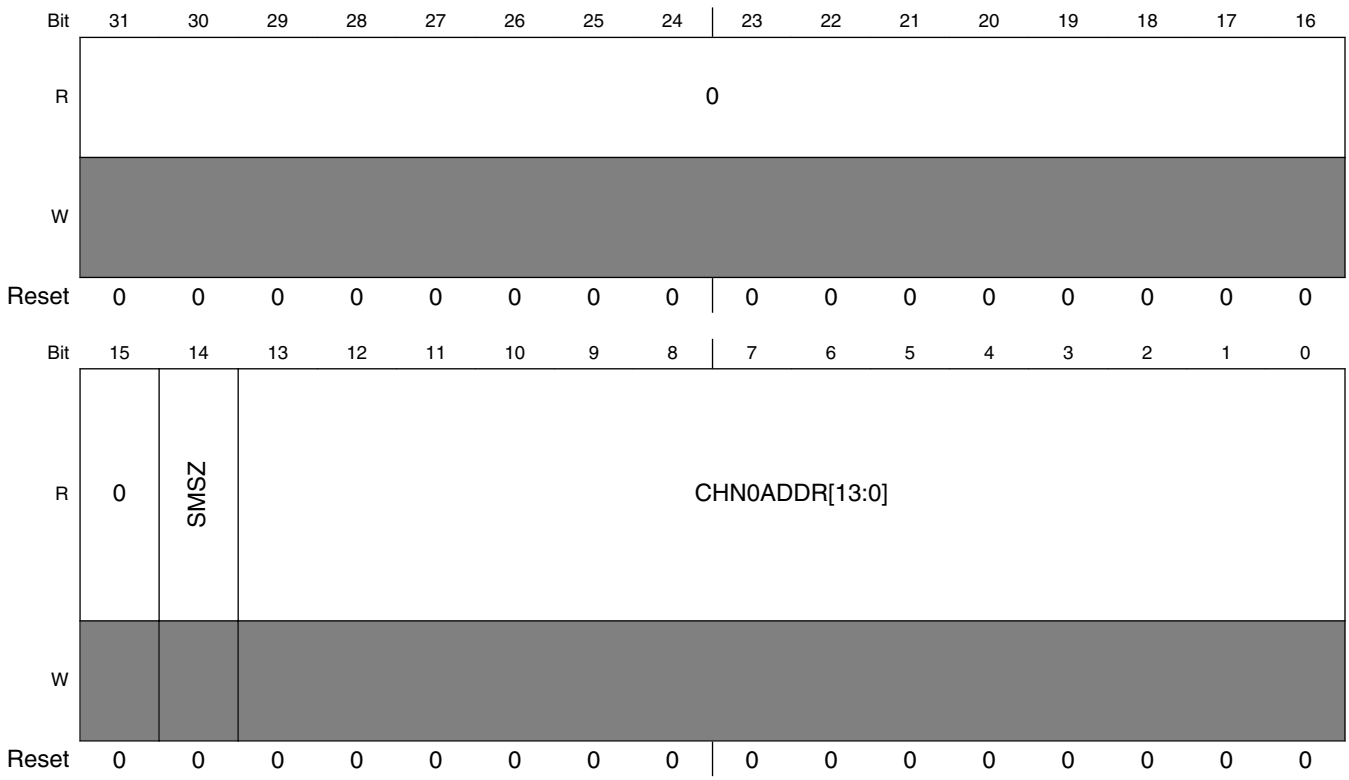
Table continues on the next page...

SDMACORE_OSTAT field descriptions (continued)

Field	Description
	14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 JTAG interface controls the OnCE. 1 ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value 0.
ECDR[2:0]	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addressA, addressB, and data conditions; the value of those fields is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one EDR bit is set; the meaning of the encoding is as follows: 0 1 matched addressA condition 1 1 matched addressB condition 2 1 matched data condition

43.10.18 Channel 0 Boot Address (SDMACORE_MCHN0ADDR)

Address: 20E_C000h base + 1Ch offset = 20E_C01Ch

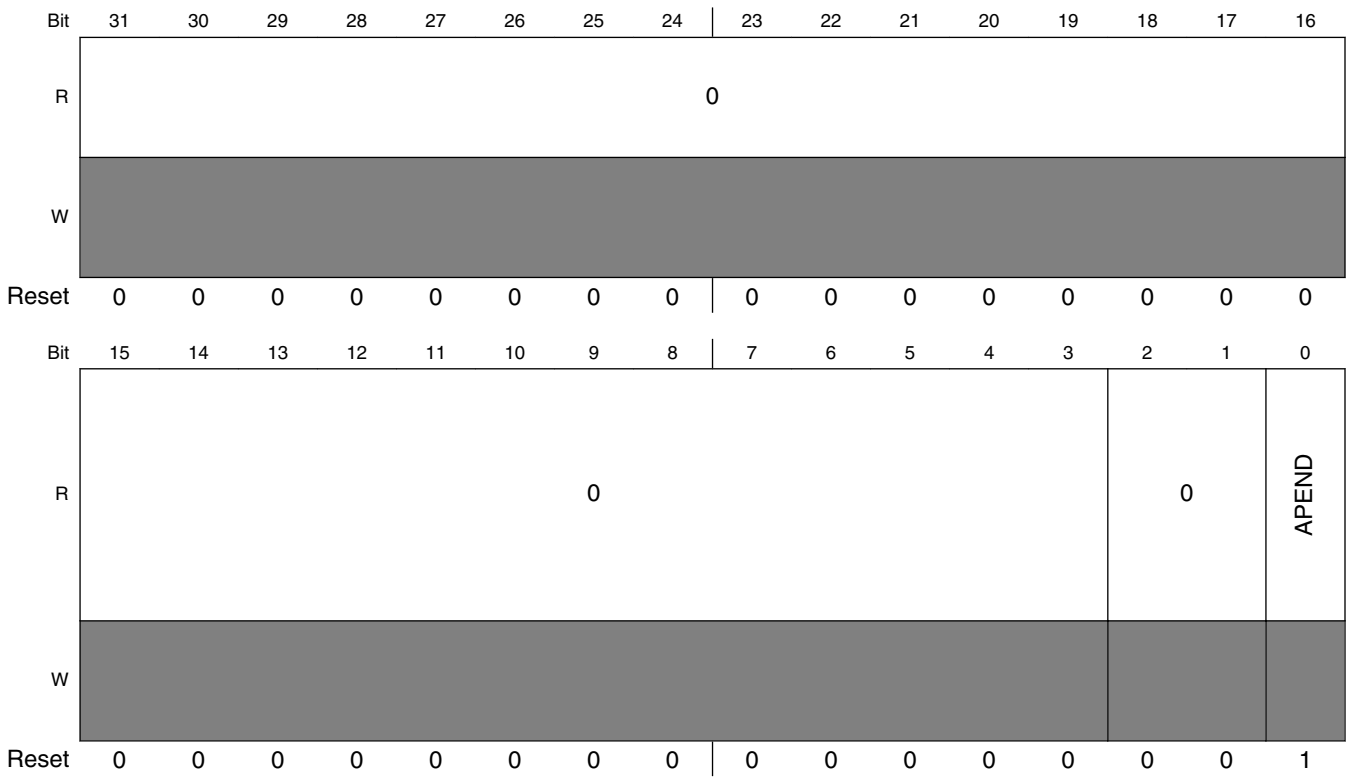


SDMACORE_MCHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value 0.
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context
CHN0ADDR[13:0]	Contains the address of the channel 0 routine programmed by the ARM platform; it is loaded into a general register at the very start of the boot and the SDMA jumps to the address it contains. By default, it points to the standard boot routine in ROM.

43.10.19 ENDIAN Status Register (SDMACORE_ENDIANNESS)

Address: 20E_C000h base + 1Dh offset = 20E_C01Dh

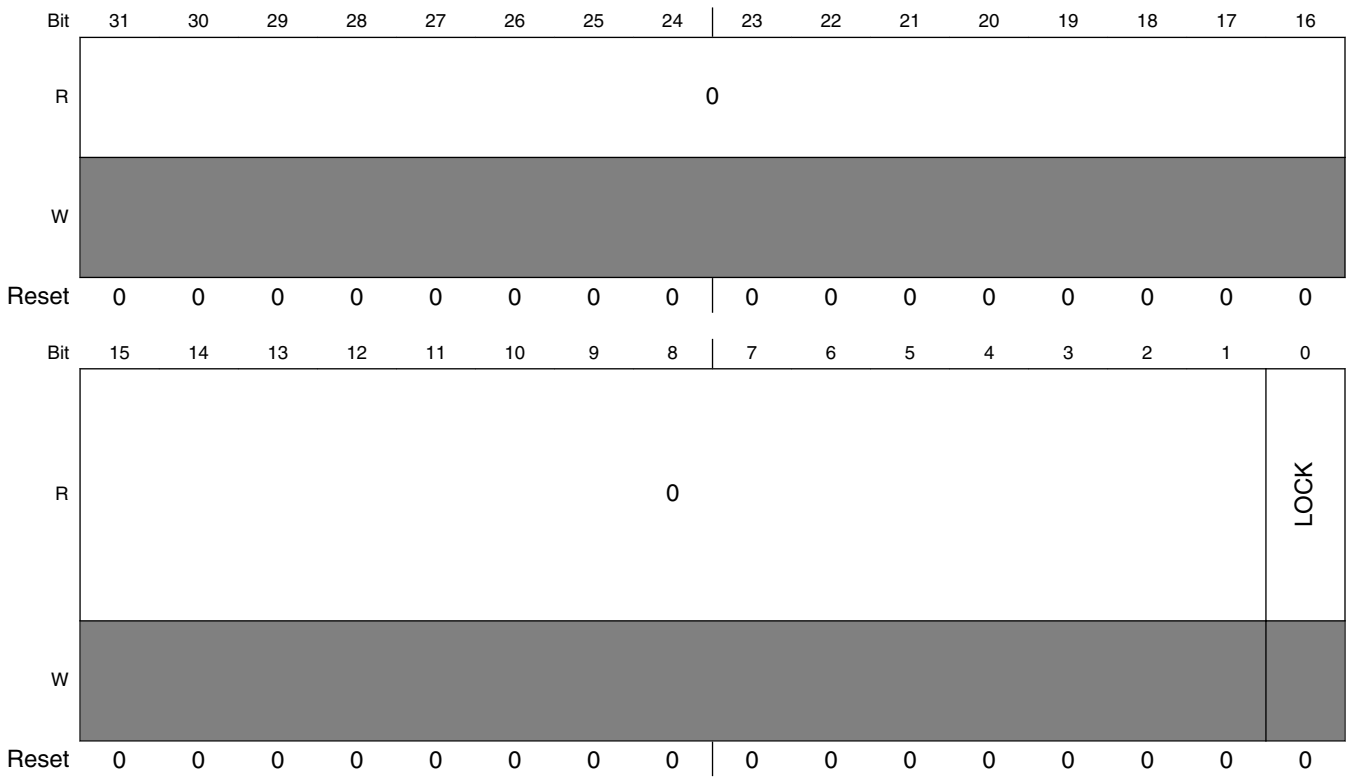


SDMACORE_ENDIANNESS field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
2–1 Reserved	This read-only field is reserved and always has the value 0.
0 APEND	APEND indicates the endian mode of the Peripheral and Burst DMA interfaces. This bit is tied to logic '1' indicating little-endian mode. 0 - ARM platform is in big-endian mode 1 - ARM platform is in little-endian mode

43.10.20 Lock Status Register (SDMACORE_SDMA_LOCK)

Address: 20E_C000h base + 1Eh offset = 20E_C01Eh



SDMACORE_SDMA_LOCK field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 LOCK	The LOCK bit reports the value of the LOCK bit in the SDMA_LOCK status register. SDMA software may use this value to determine if certain operations such as loading of new scripts is allowed. 0 - LOCK bit clear 1 - LOCK bit set

43.10.21 External DMA Requests Mirror #2 (SDMACORE_EVENTS2)

Address: 20E_C000h base + 1Fh offset = 20E_C01Fh

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

SDMACORE_EVENTS2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 32-47. The separate EVENTS register displays events 0-31.

43.11 SDMA Peripheral Registers

Refer to the respective peripherals' chapters for more information.

Chapter 44
Subscriber Identification Module (SIM)

44.1 SIM Overview

The Subscriber Identification Module (SIM) is designed to facilitate communication to SIM cards or Eurochip pre-paid phone cards, and compatible with ISO/IEC 7816-3 standards. See the SIM block diagram in the figure below. The SIM module has one port that can be used to interface with the various cards. The interface with the Micro Controller Unit (MCU) is a 32-bit connection as described in the reference document IP Bus Specification.

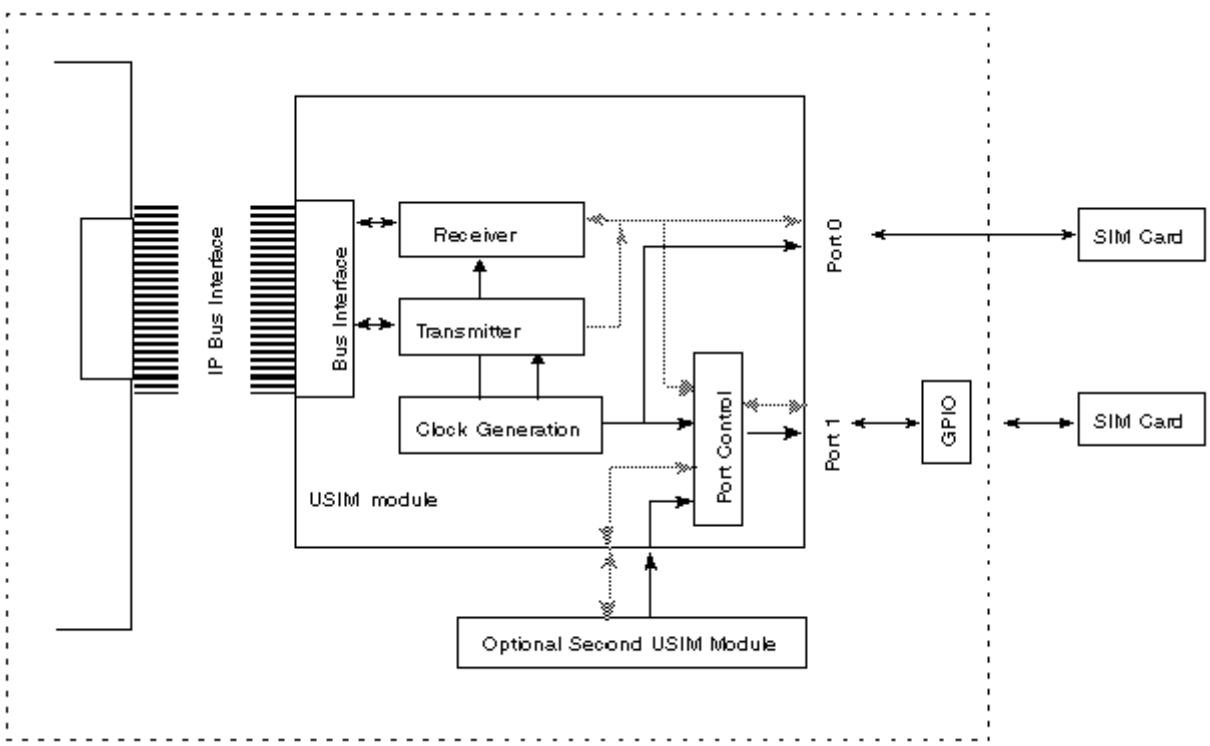


Figure 44-1. SIM Block Diagram

44.1.1 Features

The SIM design is summarized in the following sections:

- SIM Bus Interface Overview
- SIM Clock Generator Overview
- SIM Transmitter Overview
- SIM Receiver Overview
- SIM Port Control Overview
- SIM General Purpose Counter Overview
- SIM LRC Block Overview
- SIM CRC Block Overview

44.1.2 Modes of Operation

The SIM module I/O interface can be operated in one of three modes of operation summarized below.

- Two wire interface. In this mode both the IC pin RX and TX are used to interface to the SmartCard. This is activated by resetting the 3VOLTx bit in the port control register to a "0".
- External one wire interface. In this mode the IC pins RX and TX are tied together external to the IC and routed to the SmartCard. The 3VOLT bit in the port control register is reset to a "0" and the OD_Px bit in the OD_CONFIG register is set to a "1". For this interface to work properly the IC pin (RX-TX) must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.
- Internal one wire interface. In this mode the IC pin TX is routed to the SmartCard. The receive pin RX is connected to the TX pin internal to the IC. The 3VOLTx bit in the port control register is reset to a "1" and the OD_Px bit in the OD_CONFIG register is set to a "1". For this interface to work properly the IC pin TX must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.

NOTE

For the internal one wire interface to work, the bond pad must be capable of simultaneous input and output. (The input path should not be disabled when the output is enabled.)

44.1.3 SIM Bus Interface Overview

The SIM module is designed with a separate block that encompasses the interface to the IP bus. The bus interface is built in such a way as to be easily modified to other bus definitions. The SIM module is to be considered a 32-bit peripheral. To avoid endian issues with register access, all read/writes should be done as word access (32-bit). The bus interface has the following features:

- Peripheral Address/chip select decoding
- Illegal Address generation
- Dynamic wait state generation (tied inactive)
- Register organization
- Register bit implementations

See the table below for a summary of SIM top level interrupts.

Table 44-1. SIM Top Level Interrupt Summary

Flag	Interrupt Sources	Interrupt Masks	Description
SIMIRQ_N	XTE, OEF, SDI0, SDI1, TFO, GPCNT, BWT, BGT, CWT, RTE	XTM, OIM, SDIM0, SDIM1, TFOM, GPCNTM, BWTM, BGTM, CWTM, RTM	SIM General Interrupt
SIMDAT_N	TC, ETC, TFE, RDRF, TDTF, RFE	TCIM, ETCIM, TFEIM, RIM, TDTFM,RFEM	SIM Data Interrupt

44.1.4 SIM Clock Generator Overview

The SIM module contains a block designed specifically for generating the clocks used internal to the SIM module, and the clocks provided to the SIM cards. The clock generator has the following features:

- Programmable Divisor (CLK_PRESCALER[7:0]) for SIM card clock generation (45-55% duty cycle except for divider by 3, 5, 7, 9)
- Receive clock generation (/1, /2, /4, /8, /16, /32, /31) from first stage
- Transmit clock generation (/12, /8) from second stage
- Programmable divisor for receive clock generation
- There are no interrupt sources generated by the SIM clock generator block

44.1.5 SIM Transmitter Overview

The SIM transmitter block contains the transmit state machine, transmit shift register, and transmit FIFO. The transmitter has the following features.

- 16 bytes deep transmit FIFO
- Automatic NACK generation on parity and overrun errors
- Hardware data format support (inverse convention or direct convention)
- Retransmission of data upon SIM card NACK request with programmable maximum threshold of retransmissions
- Programmable guard time between transmitted bytes
- Six interrupt sources (see the table below)

Table 44-2. SIM Transmitter Interrupt Summary

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit Complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early Transmit Complete
TFE	XMT_STATUS	TFEIM	INT_MASK	Transmit FIFO Empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit Threshold Error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill Error
TDTF	XMT_STATUS	TDTFM	INT_MASK	Transmit Data Threshold Flag

44.1.6 SIM Receiver Overview

The SIM receiver block contains the receive state machine, receive FIFO, and control logic. The receiver has the following features.

- 288 bytes deep receive FIFO
- Decoding of initial character mode for setting data format
- Hardware data format support (inverse convention or direct convention)
- NACK detection
- Eleven ETU character support
- Character Wait Time Counter
- Six interrupt sources (see the table below)

Table 44-3. SIM Receiver Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive Nack threshold
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time flag
OEF	RCV_STATUS	OIM	INT_MASK	Overrun Error Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive Data Register Full
RFE	RCV_STATUS	RFEM	INT_MASK	Receive Fifo Empty

44.1.7 SIM Port Control Overview

The SIM module supports numerous port control functions necessary for SIM card interaction. Each of the two SIM card ports has the following I/O: CLK, $\overline{\text{RST}}$, DATA_RCV, DATA_XMT, SIMPD (SIM presence detect), and SVEN (SIM V_{CC} enable). The following list gives a number of the important features of the port logic.

- Open-drain or push pull DATA_XMT pin
- Port 1, port 0, ore alternate port selection
- SIM card presence detect with interrupt capability
- Sleep mode wake-up via SIM card presence detect interrupt
- Manual control of all SIM card interface signals
- Automatic power down of port logic on SIM card presence detect
- Two interrupt sources. See Table 44-4.

Table 44-4. SIM Card Detect Interrupts

Flag	Flag register	Mask	Mask Register	Description
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM Detect Interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT	SIM Detect Interrupt for port 0

44.1.8 SIM General Purpose Counter Overview

The SIM module provides a 16-bit counter that can be configured to count using clock sources from the card clock, receive clock, or transmitter clock (ETU Clock). A programmable 16-bit register is provided to compare with the counter for interrupt generation:

- Selectable clock source
- 16-bit counter and comparator
- One Interrupt source (see the table below)

Table 44-5. SIM General Purpose Counter Interrupts

Flag	Flag Register	Mask	Mask Register	Description
GPCNT	XMT_STATUS	GPCNTM	INT_MASK	GPCNT Comparator Interrupt

44.1.9 SIM LRC Block Overview

The SIM module contains a block designed to generate Linear Redundancy Checking (LRC) information for both received and transmitted characters. The LRC block has the following features:

- 8-bit LRC value
- Valid LRC Detector
- There are no interrupt sources generated by the SIM LRC block

44.1.10 SIM CRC Block Overview

The SIM module contains a block designed to generate Cyclic Redundancy Checking (CRC) information for both received and transmitted characters. The CRC block has the following features:

- 16-bit CRC value
- 16-bit CRC Polynomial Generator
- Valid CRC Detector
- There are no interrupt sources generated by the >SIM CRC block

44.2 External Signal Description

44.2.1 External Signals Overview

See the table below for summary of the SIM module signals.

Table 44-6. Signal Properties

Name	Port	Function	Reset State	Pull-up
External Signals				
SIM_CLK	O	Clock for the smartcard on port0	0	Active
SIM_RST_B	O	Reset signal for port0	0	Active
SIM_SVEN	O	Vcc enable signal for port0	0	Active
SIM_TRXD	O	Transmit/receive data signal for port0	0	Active/Passive
SIM_PD	I	Card insertion detect signal for port0	-	-
sim_pin_sclk1	O	Clock for the smartcard on port1	0	Active
sim_pin_srst1	O	Reset signal for port1	0	Active
sim_pin_sven1	O	Vcc enable signal for port1	0	Active

Table continues on the next page...

Table 44-6. Signal Properties (continued)

Name	Port	Function	Reset State	Pull-up
sim_pin_data1_tx_out	O	Transmit/receive data signal for port1	0	Active/Passive
pin_sim_simpd1	I	Card insertion detect signal for port1	-	-
sim_rcvd1_io	I	I/O Receive data signal for port1	-	-
Module Interrupts				
ipi_int_sim_ipb_int	O	Active high SIM Interrupt. Composed of OEF, XTE, SDI1, and SDI0.	0	-
ipi_int_sim_data_irq	O	Active high SIM Data Interrupt. Composed of TC, ETC, TFE, and RDRF.	0	-
Module DMA Requests				
ipd_sim_tx_dmareq_b	O	Active low. Transmitter DMA request.	1	-
ipd_sim_rx_dmareq_b	O	Active low. Receiver DMA request.	1	-

44.2.2 Detailed Signal Descriptions

44.2.2.1 SIM_CLK

The SIM_CLK is an output from the chip to the SmartCard. This signal is the clock that the SIM module provides for the SmartCard. Typical frequencies are 1 MHz to 5 MHz. This clock is 372 times the data rate that is on pin SIM_TRXD. There is no required timing relationship between this clock signal and any of the other data signals. This is because of the asynchronous nature of the protocol.

44.2.2.2 SIM_RST_B

The SIM_RST_B is the reset signal from the SIM to the SmartCard.

44.2.2.3 SIM_SVEN

The SIM_SVEN is the SmartCard power supply enable control signal.

44.2.2.4 SIM_TRXD

The SIM_TRXD is the data transmitted/received from the SIM module to the SmartCard. In a 1-wire mode of operating, this output port must be bidirectional with a pull-up resister off chip.

44.2.2.5 SIM_PD

The SIM_PD is the SmartCard insertion detect.

44.3 SIM Functional Description

To best describe the organization of the SIM module from a user's point of view, it is instructive to view the SIM at a number of different levels of hierarchy. See Figure 44-1 for illustration of the organization of SIM and connection of the signals to the available serial port.

The SIM module is essentially a standard UART with some special provisions made for SIM card communication. The SIM consists of the following main parts:

- IP bus interface
- Bus interface
- Clock generator
- Transmitter
- Receiver
- General purpose counter
- LRC blocks
- CRC blocks

See the figure below for an illustration of the block diagram in detail, specifically the main parts.

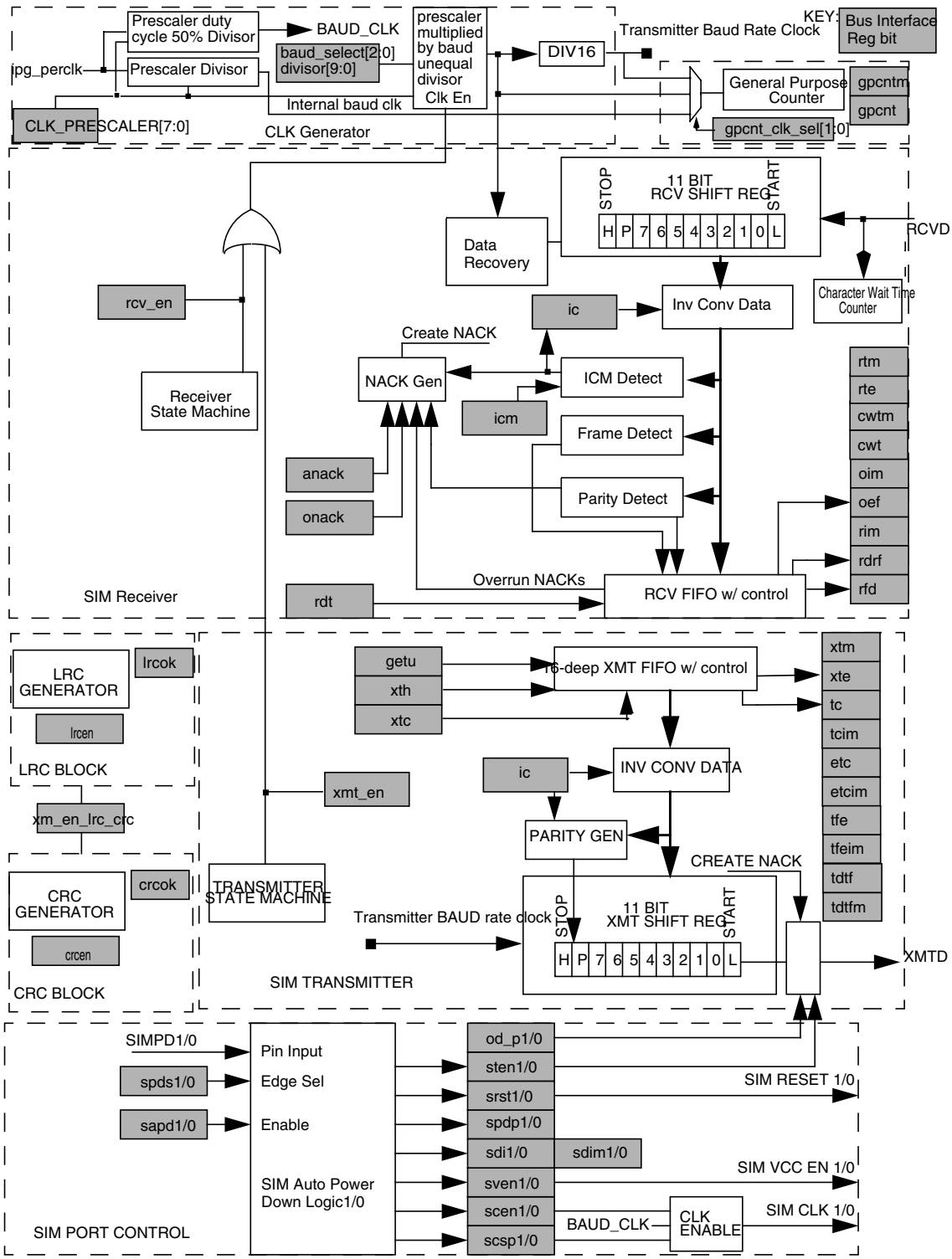


Figure 44-2. SIM Detailed Block Diagram

44.3.1 SIM Bus Interface

The IM Bus interface block has been designed to enable the SIM module to be easily ported to other MCU cores. The bus interface block contains all of the logic that uses the bus interface signals. See the figure below.

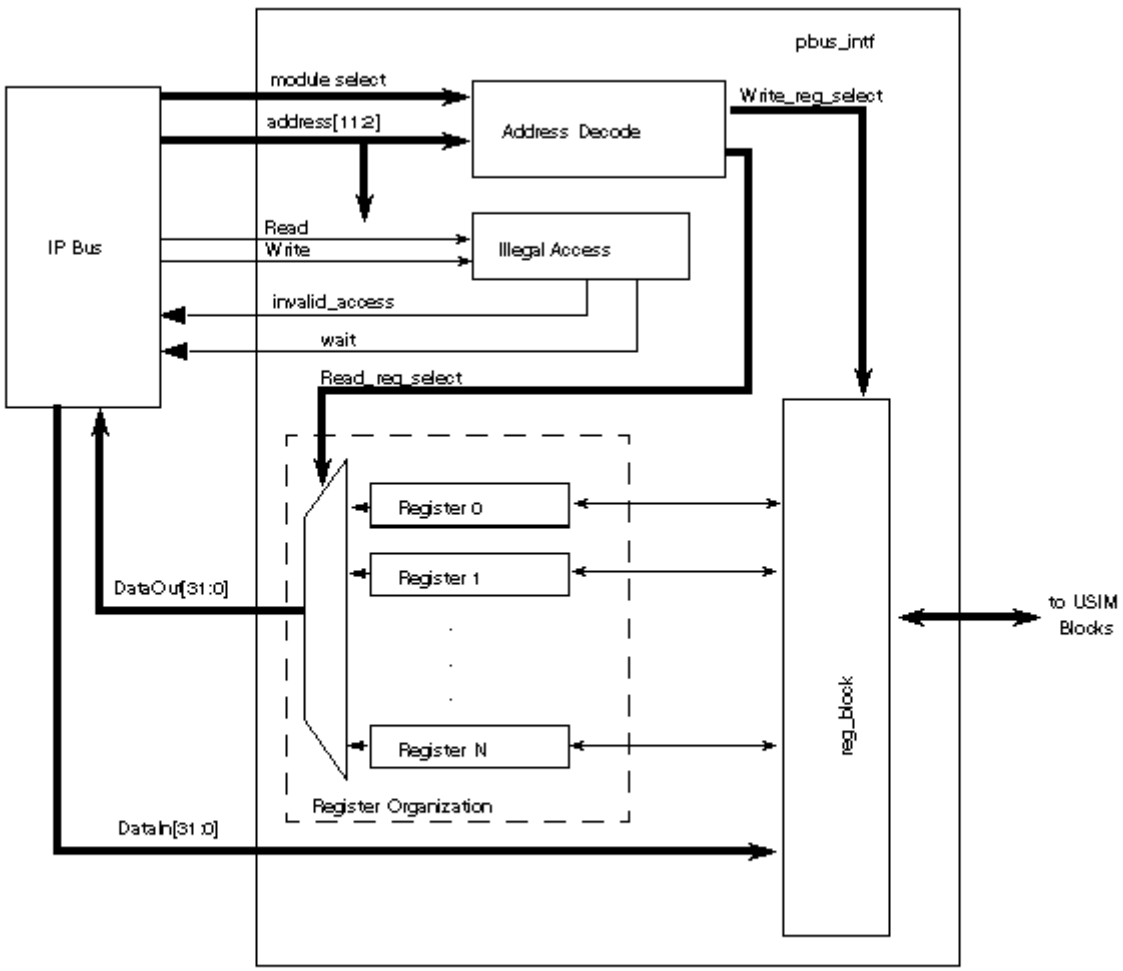


Figure 44-3. SIM Bus Interface

The bus interface block is responsible for peripheral bus address decoding, illegal access detection, dynamic wait-state requests, register organization, and register bit implementations. The address decoding uses the module address bus input along with the module select output of the IP bus to decode if the SIM module is being addressed. This decoding of these signals is done asynchronously. The output of the address decode is used with the read/write signals from the IP bus to determine the action requested by the bus master. If the read signal is active, the data_out bus will be driven with the register

selected by the address decoding. If the write signal is active, the data_in bus will be latched into the register selected by the address decoder at the rising edge of the data_strobe signal.

The illegal access detection is implemented similarly to the address decoding. If the module_select signal becomes active while the address inputs are pointing to an unimplemented address, the invalid signal is asserted asynchronously to the bus master. The invalid signal can also be generated under certain register access conditions such as writing to a full transmit FIFO, or writing to a read-only register. See the figure below.

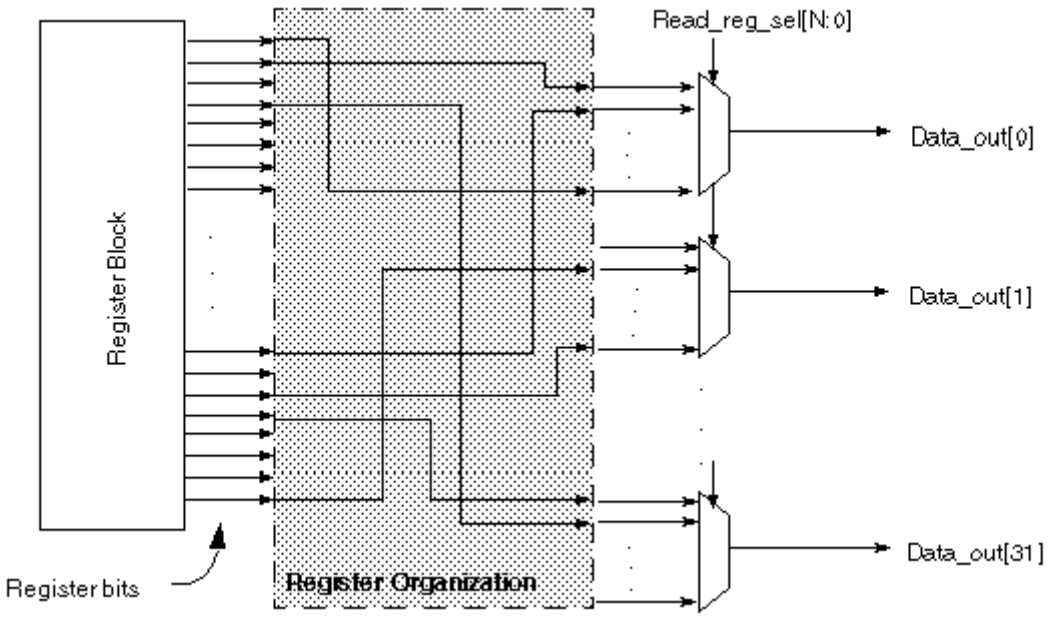


Figure 44-4. SIM Read Registers

The register bit implementations are done inside the reg_block sub-module. This block accepts the decoded read and write signals for the register bits generated in the address decoder. The data_strobe output is used as the clock input to the register bits in order to clock in the new data during a write access. The write_reg_sel signals are used to gate the write action with the address decoding. The clearing mechanisms for status bits are implemented as write-one-to-clear. See the figure below.

.

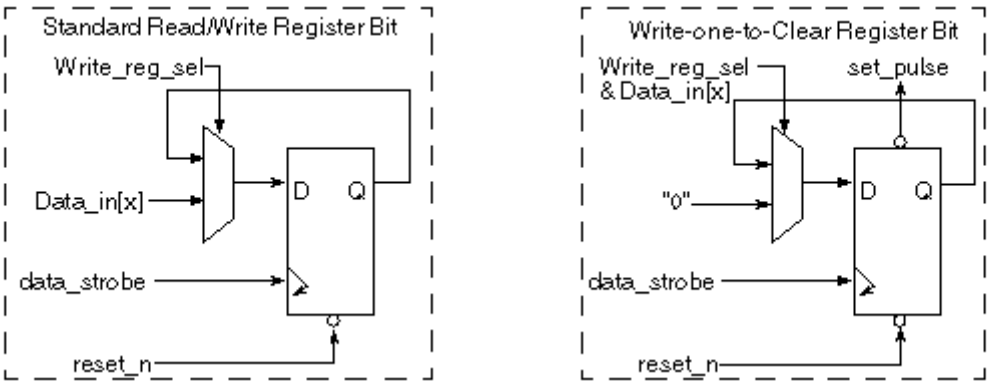


Figure 44-5. Register Bit Diagram

In the "write-one-to-clear" register bit example, "set_pulse" represents a pulsed signal generated in the data_strobe clock domain to set the register output.

44.3.2 SIM Clocking

The SIM has three clock inputs:

- ipg_clk-read/write clock; the frequency is half of platform_clk
- ipg_per_clk-reference frequency input clock to generate the SIM card clock and other internal clocks; the frequency is platform_clk divided by 8
- ckil-low frequency clock (32 KHz) used by the auto power down hardware circuit in the SIM. It is generated by dividing the platform_clk by a division factor which is programmed in the SIMCLKCR register in the global utilities. The default value of the divider is 0x3D09, which gives a 32 KHz clock from a platform_clk of 500 MHz.

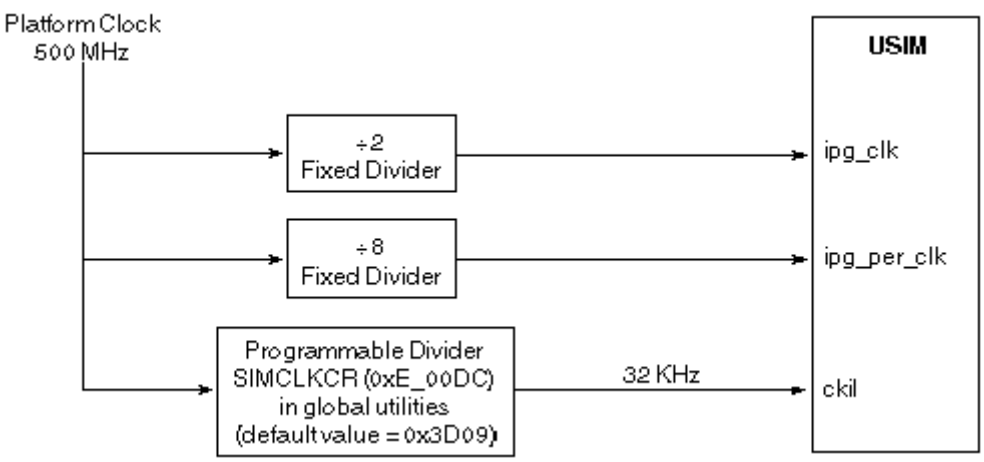


Figure 44-6. SIM Clock Generator

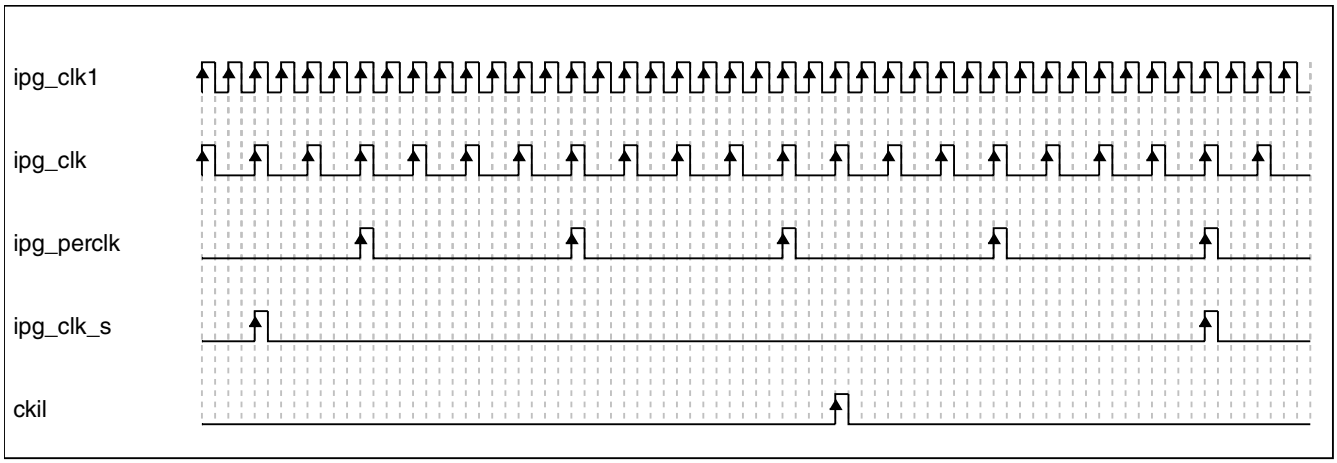


Figure 44-7. SIM Clocking Diagram

44.3.3 SIM Clock Generator

The clock generator is responsible for implementing clock tree synthesis constructs, scan clock muxing, baud rate clock (BAUD_CLK) generation, and providing clocks to the transmitter, receiver, and port controller sections of the SIM module. See the figure below for the schematic of the SIM clock generator. The dividers outlined in bold, generate a pulsed (gated) clock of the desired frequency. The pulse is equal in duration to one half the ipg_per_clk period. This clock is used internal to the SIM module to drive the transmit and receive sections of the module.

The dividers outline in normal, generate a clock of 45-55% duty cycle to drive the clock pin of the SmartCard. This clock is not used internal to the SIM module.

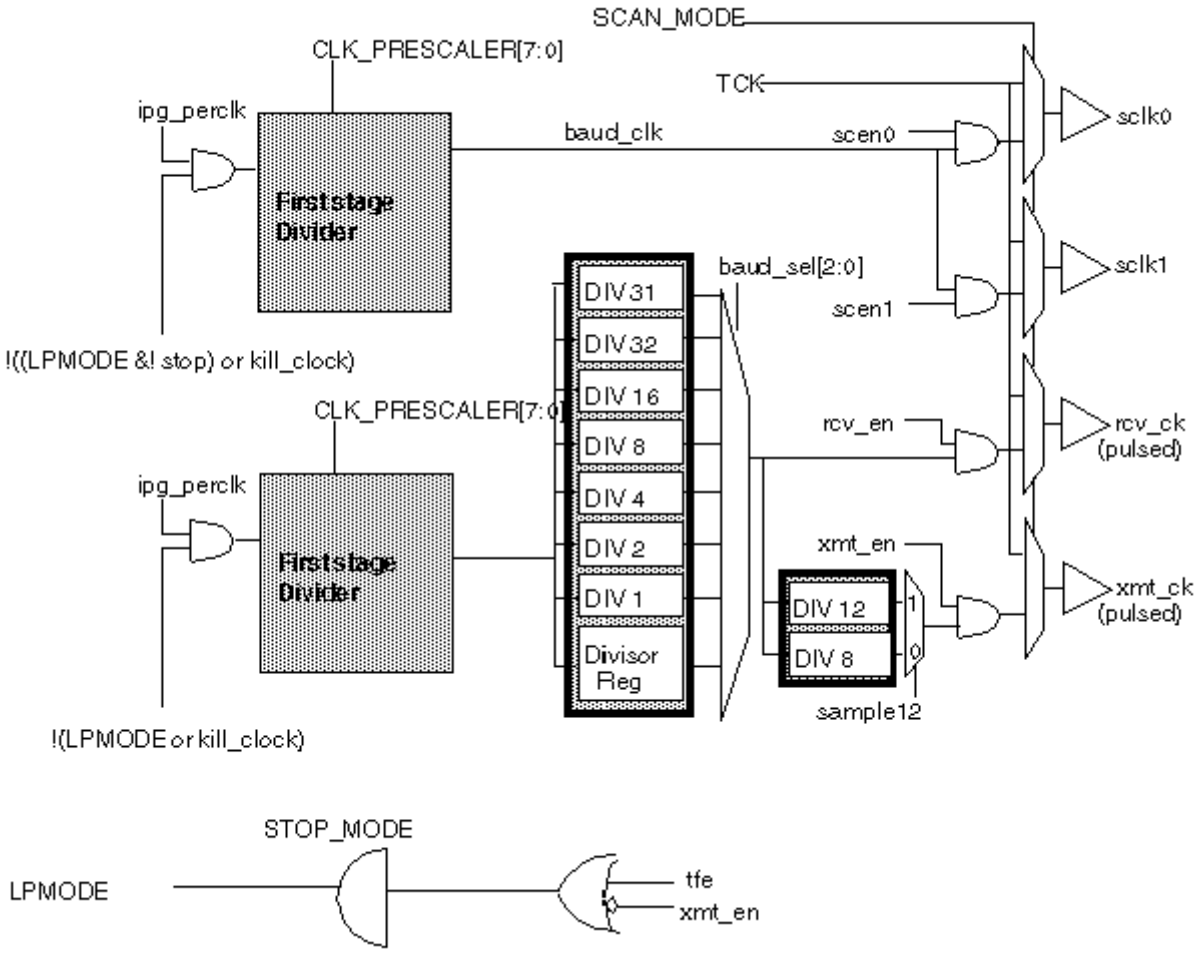


Figure 44-8. SIM Clock Generator

44.3.3.1 Scan Test

The scan clock muxing is implemented according to the design for testability rules.

44.3.3.2 Baud Clock Generation

The baud rate clock generation performed by the clock generator results in different frequencies. The default frequency is a divide by two of the ipg_perclk input. The baud rate can be programmed to be ipg_perclk divided by the even value using the CLK_PRESCALER[7:0] bits as shown in the CLK_PRESCALER register. The baud rate clock is generated in two forms in the design. The BAUD_CLK that is used by the SIM cards (SCLK0) is generated so that it must be 45-55% duty at the divide values. This is necessary to meet the requirements of the ISO 7816 specification. The BAUD_CLK that is used internal to the SIM module is generated as a gated version of the ipg_perclk

clock. The resultant clock will be a pulse of one-half ipg_perclk period in width with the expected frequency. The gated baud clock complies with the clock methodology initiative. The 50% duty cycle clock does not comply with the clock methodology initiative but this clock is not used as a clock internal to the SIM module. It is only used as a data path.

44.3.3.3 Transmitter Clock Generation

The transmitter clock (xmt_clk) is generated by the clock generator based on the values passed to it for the baud rate select (baud_sel[2:0]) and sample12. The transmit clock is gated by the transmit enable (XMT_EN) register bit. When the transmitter is enabled, the clock generator counts the appropriate number of receive clock (rcv_clk) positive edges to determine when to toggle the transmitter clock output. The transmitter clock is always based upon the receive clock.

44.3.3.4 Receiver Clock Generation

The receiver clock (rcv_clk) is generated by the clock generator based on the value passed to it for the baud rate select (baud_sel[2:0]). The receiver clock is gated by the receiver enable (RCV_EN) register bit. When the receiver is enabled, the clock generator counts the appropriate number of BAUD_CLK positive edges to determine when to toggle the receiver clock output. The number of BAUD_CLK positive edges is determined by the value of the baud rate select input (baud_sel[2:0]) and is programmable to these divisors: 31 (slowest because used with the 372/1 Fi/Di rate), 32, 16, 8, 4, 2, 1, and divisor[7:0]. The programmable divisor (divisor[7:0]) is programmable via the DIVISOR_REG register.

44.3.3.5 Port Control Clock Generation

The port controller clocks are provided by the clock generator for use on the SIM card ports. These clocks are equivalent in frequency to the BAUD_CLK and are gated by the SIM clock enable (scen0, scen1) signals. The level at which the card clocks (SCLK0, SCLK1) are stopped when disabled is determined by the SIM clock select polarity (SCSP0, SCSP1) inputs to the clock generator. Synchronizers are implemented to ensure glitch free operation of the card clocks when enabling/disabling, or changing the clock stopped polarity.

44.3.3.6 Low Power Mode Clock Control

The clock generator block is responsible for gating the clocks to the SIM module appropriately whenever a low power mode instruction is decoded. The IP interface provides two signals that encode the two available low-power states of the processor. These states are: STOP and DOZE. The response to the DOZE instruction is controlled through the use of the DOZE bit in the RESET_CNTL register. See the RESET_CNTL register description. The response to the STOP instruction is controlled through the use of the STOP bit in the RESET_CNTL register. See the RESET_CNTL description

44.3.4 SIM Transmitter

The SIM Transmitter block has been designed to localize all transmit related circuitry into one section of hierarchy. The Transmitter block comprises the following sections of logic: transmit state machine, transmit shift register, transmit FIFO, guard time generator, transmit NACK control, and transmit data convention.

44.3.4.1 Transmit State Machine

The Transmit state machine is the heart of the transmitter block. The state machine is responsible for sequencing through a transmit operation while reacting to inputs from the receiver, the transmit FIFO, and the guard time circuit. See the figure below for flow diagram of the transmit state machine.

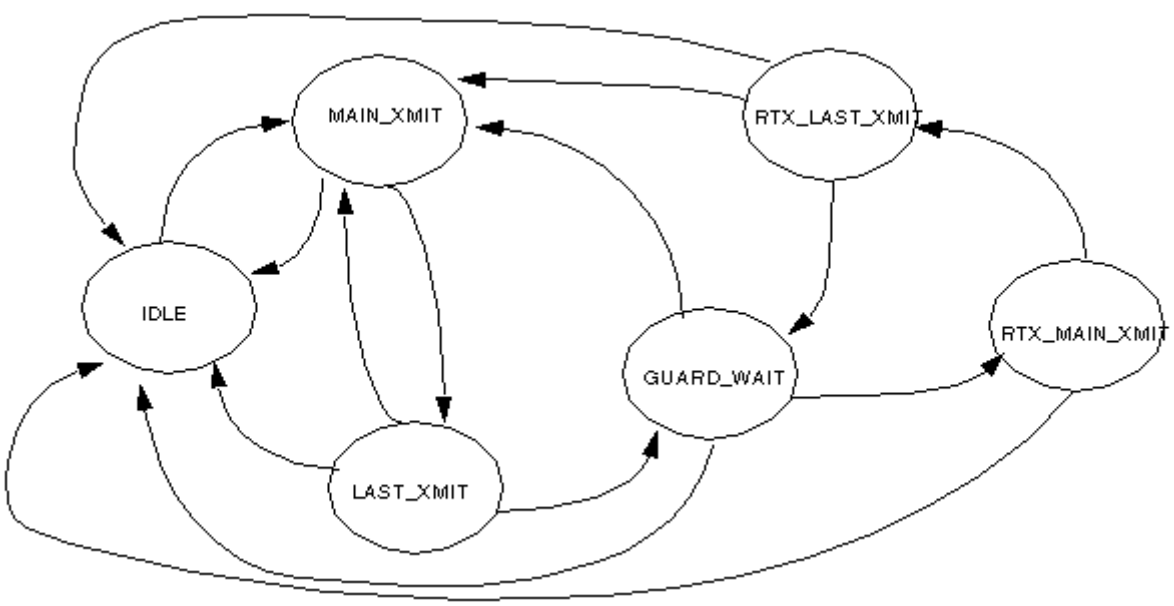


Figure 44-9. Transmit State Machine

The functions performed by each state are:

- IDLE
  - This is the initial state. The state machine waits here until it detects XMT_EN is set and a write to the transmit FIFO has occurred. The data pointed to by the transmit read pointer is loaded into the shift register, and the state machine transitions to the MAIN_XMIT state. Any time XMT_EN is cleared, the state machine returns to this state.
- MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the LAST_XMIT state.
- LAST_XMIT
  - This state transmits the last bit of the current transmission and determines the next operation. One of the following will occur.
    - If GETU[7:0] is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in GETU[7:0], jump to MAIN_XMIT state to retransmit the current byte.
    - If no transmit NACK, and GETU[7:0] is zero, load the shift register, jump to MAIN_XMIT to transmit the next byte
    - If no transmit NACK, GETU[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.
- GUARD_WAIT
  - The state machine remains in this state until the Guard time counter has expired.
    - If a transmit NACK error occurred on last transmission, jump to RTX_MAIN_XMIT and re-transmit
    - If no transmit NACK and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte.
    - If no transmit NACK and the FIFO is empty, return to the IDLE state.
    - If transmit NACK threshold is detected, stop transmitter, set XTE flag, and jump to the IDLE state.
- RTX_MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the RTX_LAST_XMIT state. This state is identical to the MAIN_XMIT state except that it retransmits the previously NACKed byte.
- RTX_LAST_XMIT
  - This state transmits the last bit of the current re-transmission and determines the next operation. One of the following will occur.
    - If GETU[7:0] is non-zero, jump to the GUARD_WAIT state.

- If a transmit NACK error occurred, with a zero in GETU[7:0], jump to GUARD_WAIT state to check Transmit NACK threshold.
- If no transmit NACK, GETU[7:0] is zero, and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte
- If no transmit NACK, GETU[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.

44.3.4.2 Transmit Shift Register

The transmit shift register is 11 bits wide and controlled by the transmit state machine described previously. The shift register shifts out data at the transmit clock frequency (xmt_ck).

44.3.4.3 Transmit FIFO

The transmit FIFO is implemented inside the transmitter block. The FIFO depth is 16 bytes. The FIFO block is shared by both SIM module ports. The transmit FIFO cannot be accessed by the alternate SIM module through the alternate port. Each write to the transmit FIFO increases the transmit FIFO write pointer. Each time the transmit shift register is loaded from the transmit FIFO, the transmit FIFO read pointer is increased. When the read and write pointers are equal, the transmit FIFO empty flag (TFE) is set. Software has no visibility of the transmit FIFO pointers, but a transmit FIFO threshold value can be set to alert the software when the number of bytes in the FIFO has reached a specified level. A read of the transmit FIFO register (XMT_BUF) will return the last byte written to the FIFO. Writes to the transmit FIFO can occur at any time.

The transmit FIFO can be flushed by setting the XMT_FLUSH bit in the RESET_CNTL register. A transmit NACK threshold error (XTE) will halt the transmitter, and flush the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The Transmit data threshold flag (TDTF) does not get cleared by the XMT_FLUSH operation.

44.3.4.4 Transmit Guard Time Generator

The guard time generator is simply a counter that is clocked by the transmit clock in order to delay the beginning of the next transmission and the setting of the transmit complete interrupt flag (TC) by a programmable amount of transmit bit widths (ETU's).

The duration of the count is controlled by the GUARD_CNTL register (GETU[7:0]). See the figure below for depiction of three transmit operations in order to show the effect of the guard time generator logic

.

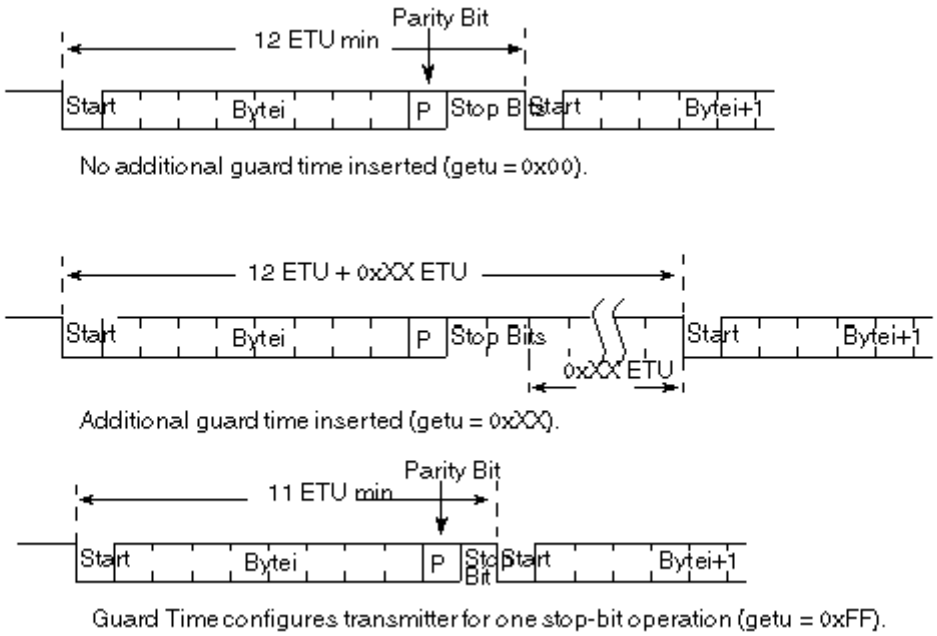


Figure 44-10. Transmit Guard Time

44.3.4.5 Transmit NACK Generator

The transmit NACK generator is responsible for driving the transmitter output low during the STOP bit time to signify an error was detected in the received data from the SIM card. This logic responds to a NACK request generated by the receiver block. The figure below shows a typical SIM transaction with the NACK pulse inserted.

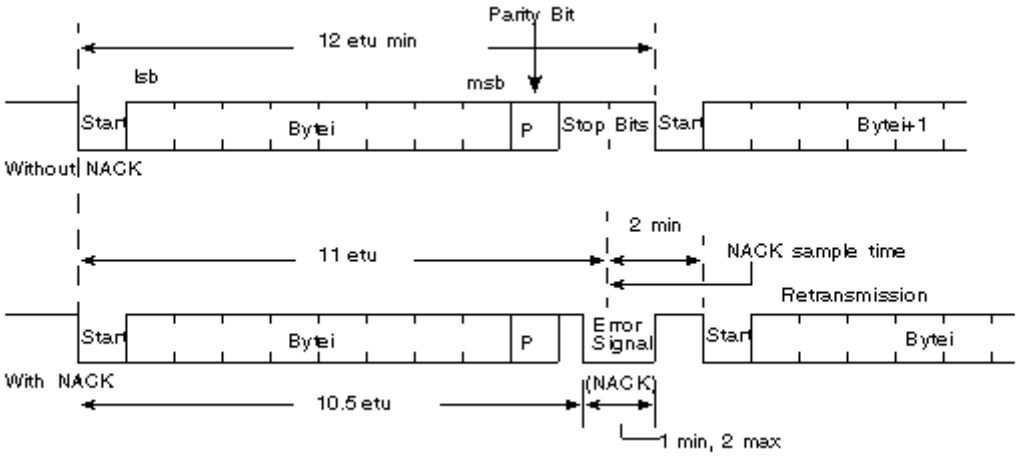
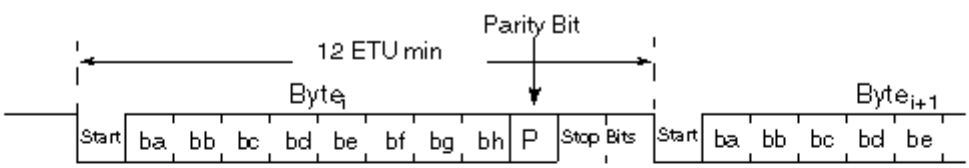


Figure 44-11. Transmit NACK Generator

The transmit NACK generator is also responsible for keeping track of the number of NACKs received during a transmit operation. The SIM receive state machine detects NACKs generated by the SIM card, and reports them to the transmit NACK logic. Once the number of detected NACKs has reached the programmed threshold (XTH[3:0]), an interrupt flag is generated, the transmit FIFO is flushed, and the transmitter is disabled.

44.3.4.6 Transmit Data Convention Logic

The transmit data convention logic provides the support for the two different data conventions available in SIM cards. See the figure below for an illustration of SIM data conventions.



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by USIM before being transmitted.

Direct Convention: *ba* is lsb of data byte to be sent. *bh* is msb.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is msb of data byte to be sent. *bh* is lsb.
Both the data bits and parity bit are logically inverted by hardware.

Figure 44-12. SIM Data Conventions

The direct data convention is the default. If the inverse convention bit (IC) in the DATA_FORMAT register is set, then the transmit data convention logic will convert the output of the transmit FIFO to the inverse convention before sending it to the transmit shift register.

44.3.5 SIM Receiver

The SIM Receiver block has been designed to localize all receive related circuitry into one section of hierarchy. The receiver block is comprised of the following functions: receive state machine and the receive FIFO.

44.3.5.1 Receive State Machine

The receive state machine is responsible for sampling the receive data pin and capturing the bit value into the receive shift register. Additionally, the receive state machine can detect the START bit, parity errors, framing errors, and initial character when operating in initial character mode.

Once enabled by the RCV_EN bit in the ENABLE register, the receive state machine sequences through the states as shown in the figure below.

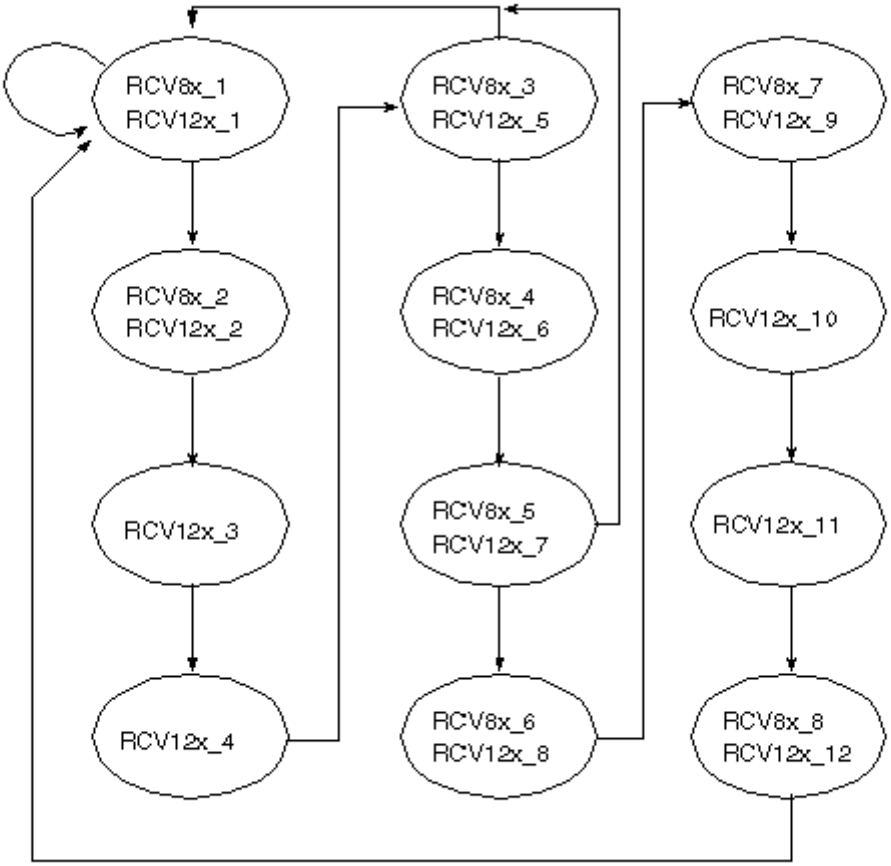


Figure 44-13. Receive State Machine

The states identified with a "8x" are used when operating in a 8x oversampling mode. The states identified with a "12x" are used when operating in a 12x oversampling mode. This mode is only used when sample12 is set to "1". The number following the oversampling mode identifier represents the state number in the current mode. There are 12 states in "12x" mode, and 8 states in "8x" mode. Some states simply implement a one RCV_CK delay. States that perform additional functions are:

- RCV8x_1, RCV12x_1
  - This is the initial state of the receive state machine. If the first bit has not been received, the state machine remains in this state until a valid START bit is detected. For every subsequent bit, this state is simply a one RCV_CK cycle delay.
- RCV8x_2, RCV12x_2
  - This state captures the first sample of the current receive data input.
- RCV12x_3
  - This state captures the second sample of the current receive data input.
- RCV12x_4
  - This state captures the third sample of the current receive data input.

- RCV8x_3, RCV12x_5
  - This state checks if we are receiving a correct START Bit. If not, Then we return to state 1.
- RCV8x_4, RCV12x_6
  - If we are in the 11th Bit, We check for Parity or Initial Character errors and send NACK if needed.
- RCV8x_5, RCV12x_7
  - Store current Bit value in Shift Register. If this is the first bit and the value is not 0, restart the State Machine.
- RCV8x_6, RCV12x_8
  - If the current bit is the last bit of the transfer, Set flag to transfer Shift Register to Receive Buffer.
- RCV8x_7, RCV12x_9
  - Clear flag for transferring Shift Register to Receive Buffer.
- RCV12x_10
  - If the current bit is the last bit of the transfer (first stop bit), this state takes the sample of the NACK window.
- RCV12x_11
  - If the current bit is the last bit of the transfer (first stop bit), this state takes the sample of the NACK window.
- RCV8x_8, RCV12x_12
  - This state represents the end of the current receive input bit. Several operations occur during this state, including:
    - Increment bit counter
    - Perform a majority vote on the NACK samples and notify the transmitter if a NACK pulse was detected.

44.3.5.2 Data Sampling / Voting

The receive state machine runs at the receive clock rate (RCV_CK). This clock is used to oversample the received data at either a 8X or 12X sample rate. For each input bit, the receive state machine captures 3 samples. A majority voting algorithm is then applied to determine the value of the bit received. The value common to 2 or more of the samples is determined to be the bit value in the receive shift register.

44.3.5.3 Start Bit Detection

The SIM receive input is defined to be high when not active. The data transmission is defined to begin with a low pulse for a bit duration. This is called the START bit. The receive state machine is responsible for detecting and validating the START bit. The

receive state machine samples the START bit three times using a majority voting scheme to determine if the START bit is valid. This will effectively filter out any low receive inputs shorter than one RCV_CK period. See the figure below for an illustration of a typical SIM data transaction with the START bit identified.

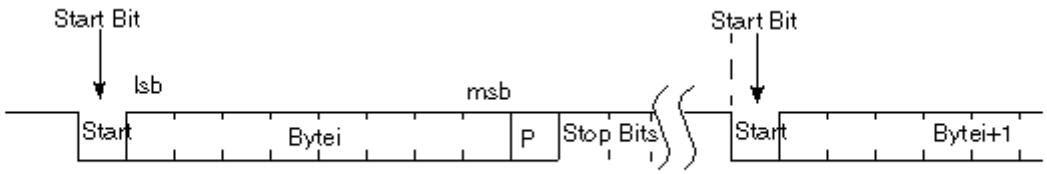


Figure 44-14. Start Bit Diagram

44.3.5.4 Parity Error Detection

The receive state machine is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except when in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented making the data appear to be odd parity. The parity bit is defined as the 10th bit of the received data. The parity of the 2nd through 10th received bits is calculated by the receiver parity logic. This logic determines if the parity of the 9 received bits is correct. See the figure below for an illustration of a typical SIM data transaction with the parity bit identified.

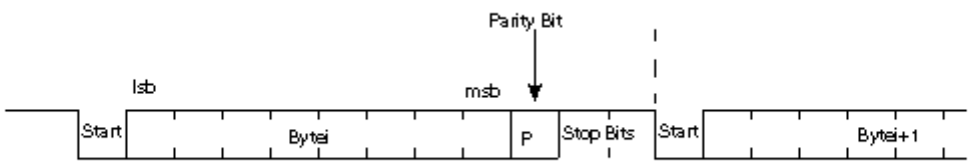


Figure 44-15. Parity Bit Diagram

When a parity error is detected on a given byte, the RCV_PF bit for that byte is set in the receive FIFO if NACK generation on errors is disabled (ANACK = 0 in the CNTL register). A parity error cannot cause an interrupt, but when NACK generation is enabled (ANACK = 1), it can signal the SIM transmitter to create a NACK pulse to the SIM card asking for a retransmission of the corrupted data.

44.3.5.5 Framing Error Detection

The receive state machine is responsible for detecting framing errors in the received data. A SIM data transaction is defined as 11 or 12 bits long consisting of the START bit, 8 data bits, 1 parity bit and 1 or 2 STOP bits. A framing error occurs when the STOP bit is not detected during the 11th bit time of a data transaction. The STOP bit is generally defined as two bit times (ETU's) of a high pulse following the parity bit. When the GUARD_CNTL register is programmed to 0xFF, the STOP bit is defined as one bit time. A framing error can only occur when the parity bit of the current byte is low, and the STOP bit arrives late. See the figure below for an illustration of a typical SIM data transaction with the STOP bits identified. Also, shown is a SIM data transaction with a late arriving STOP bit indicating a framing error.

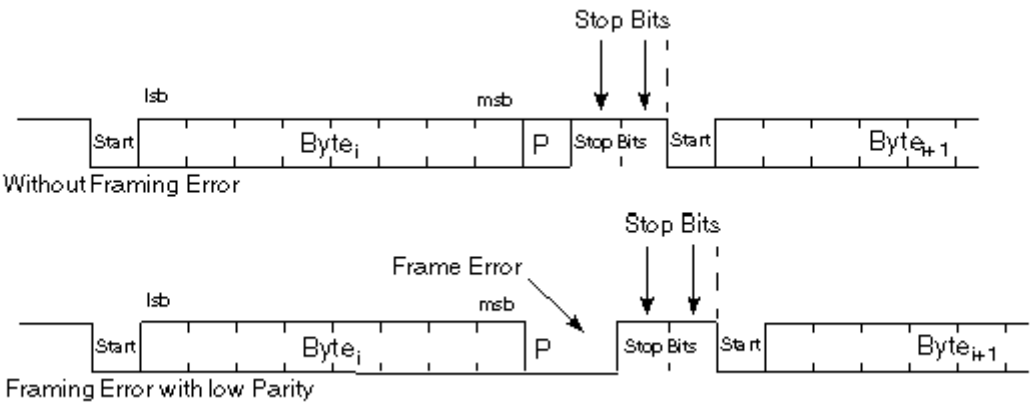


Figure 44-16. Framing Error Diagram

When a framing error is detected on a given byte, the RCV_FE bit for that byte is set in the receive FIFO. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the SIM card asking for a retransmission of the corrupted data.

44.3.5.6 NACK Detection

The existence of the NACK pulse is sampled by the receive state machine at 11 Elementary Time Units (ETUs) after the falling edge of the START bit. An ETU is equivalent in time to 1 transmit clock period. Once the receiver detects a NACK, it signals the transmitter that an error occurred. The transmitter will not initiate retransmission for at least another two ETU times as required by the ISO 7816 specification.

44.3.5.7 Initial Character Detection

The SIM receive state machine supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential characters that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. The following figures illustrate the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the data is flipped msb for lsb and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

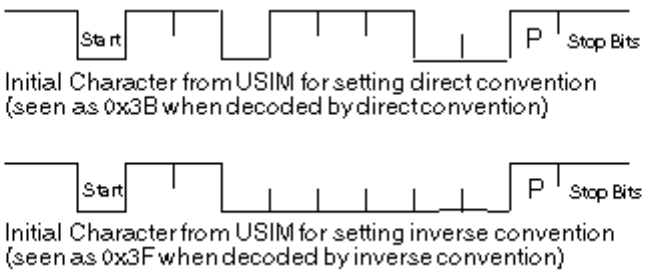
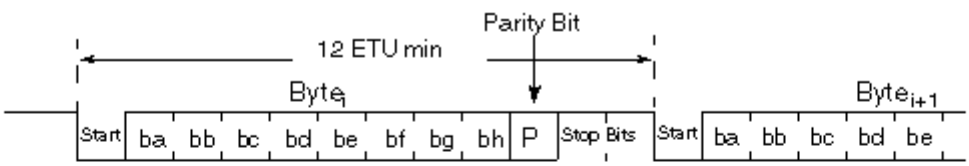


Figure 44-17. Valid Initial Characters



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM card before being transmitted.

Direct Convention: *ba* is lsb of data byte to be sent. *bh* is msb.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is msb of data byte to be sent. *bh* is lsb.
Both the data bits and parity bit are logically inverted by SIM card.

Figure 44-18. Inverse Convention versus Direct Convention

44.3.5.8 Receive FIFO

The receive FIFO is implemented inside a sub-block of the receiver block. The FIFO depth is 288 bytes. The FIFO block is shared by both SIM module ports. The receive FIFO cannot be accessed by another SIM module through the alternate port. Only the port1 IO pins can be accessed through the alternate port. The secondary SIM would then

uses its own internal FIFO while using the primary SIM IO ports. This is only done if it is required to control two SIM cards at the same time. It the two SIM card access only occurs one at a time, then a single SIM module can control both SIM cards (one on port0 and one on port1). The receive FIFO is accessed through the RCV_BUF register.

The receive FIFO is loaded from the receive shift register after the final bit of the current SIM card transmission has been received. The FIFO contains 10 bits per transmission. The lower eight bits contain the received data byte. Bits 8 and 9 contain the parity and framing status for the received byte.

Each read from the receive FIFO increases the receive FIFO read pointer. Each time the receive shift register is transferred to the receive FIFO, the receive FIFO write pointer is increased. When the difference between the read and write pointers equals the programmed threshold value (RDT), the receive data register full flag (RDRF) will be set. An interrupt can be generated by the RDRF flag if the RIM bit in the INT_MASK register is cleared. Software has no visibility of the receive FIFO pointers. A write to the receive FIFO register (RCV_BUF) will generate an invalid access exception to the processor.

The receive FIFO can be flushed by setting the RCV_FLUSH bit in the RESET_CNTL register. The flush operation resets the receive read and write pointers to equal values. All logic associated with the receiver will be reset by the flush operation except for receiver control registers.

44.3.5.9 Overrun Detection

The receive FIFO logic is responsible for detecting an overrun condition. When a received byte is transferred from the receive shift register to a receive FIFO that contains 288 unread bytes, the SIM receiver will flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 288 unread bytes in the FIFO unaltered. The SIM module will generate a NACK to the SIM card on an overrun condition if the ONACK bit in the CNTL register is set. The SIM module will continually NACK SIM card transmissions until a read of the receive FIFO occurs.

44.3.5.10 Character Wait Time Counter

The SIM receiver block includes a 16-bit counter that counts the number of bit times (ETUs) between received characters. When enabled, the Character Wait Time Counter (CWT) will not start counting until after the START bit(s) of a valid character has been received. The counter is synchronized to the receive character bit positions so that an

accurate count of the number of ETUs between characters can be made. The Character Wait Time Counter has a 16-bit programmable comparator. Software can write the expected number of ETUs between characters to the comparator. If the time between characters exceeds this value, an interrupt flag will be set and an interrupt generated if the mask is clear.

44.3.6 SIM Port Control

The SIM Port Control block has been designed to localize all port related circuitry into one section of hierarchy. The port control block comprises the following functions: SIM card interface, SIM Card presence detect, and SIM card auto-power down.

44.3.6.1 SIM Card Interface

The SIM module allows for direct control of two separate SIM cards. The SIM module does not support simultaneous communication with two SIM cards. To achieve simultaneous communication with two SIM cards, a second SIM module must be used. The SIM module can relinquish control of the inactive port to a secondary SIM module by setting the AMODE bit in the SETUP register. When alternate SIM Card Mode enable, output signals in port1 is directly from the secondary SIM module and input signals in port1 will pass through to the secondary SIM module.

The SIM card clock is generated in the SIM clock generator. The SIM card reset is controlled by software through the PORT0_CNTL register. Since the SIM module does not support a separate transmit data and receive data port, the 3VOLT0 bit should be set to 1.

44.3.6.2 SIM Card Presence Detect

The SIMPDx input allows for detection of the insertion or removal of a SIM card. The SPDSx control bit in the PORTx_DETECT register allows the software to configure which edge of the SIMPDx pin will cause a presence detect event. A maskable interrupt can be generated when a SIMPDx event occurs.

An internal pull-down device is present on the SIMPDx pins. This will provide for a high to low transition on the SIMPDx pin when a SIM card is removed.

44.3.6.3 SIM Card Automatic Power Down

When interfacing to the SIM cards, it is necessary to follow a particular sequence when powering them up and down. The SIM port control block contains hardware that provides the correct sequence to power down a SIM card (see the figure below). The power up sequence must be done manually by the software using the pin control bits supplied in the PORTx_CNTL registers.

The power down sequence is specified in ISO 7816 as:

1. RST transitions from high to low
2. CLK is turned off to a low
3. I/O transitions from high impedance to low
4. SIM V_{CC} is turned off

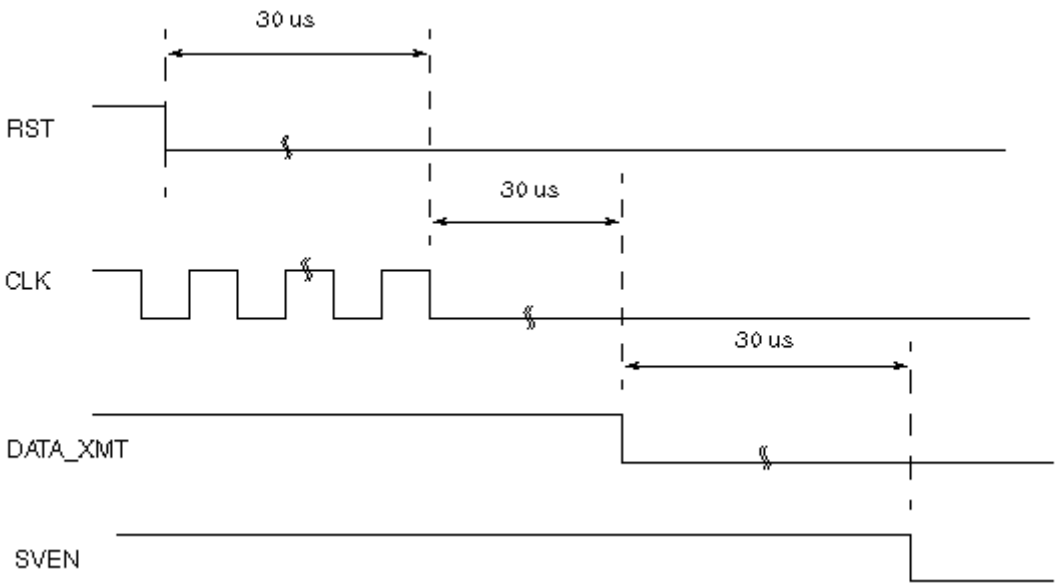


Figure 44-19. Auto Power Down Sequence

44.3.7 SIM General Purpose Counter

The SIM module provides a 16-bit counter for use when timing events during SIM card communication. The clock source for the counter is selectable between three sources: BAUD_CLK, RCV_CLK, or XMT_CK (ETU clock). The GPCNT_CLKSEL[1:0] bits in the CNTL register are used to select the clock input. The counter is enabled as soon as the input clock is selected. The starting of the counter is immediate once the input clock is running. Software can control the three input clock sources by using the KILL_CLOCK, RCV_EN and XMT_EN bits provided in the RESET_CNTL and ENABLE registers.

To run the counter from the card clock source the following conditions must be met:

1. 'KILL_CLOCK = 0' in the RESET_CNTL register.
2. 'GPCNT_CLKSET[1:0] = 01' in the CNTL register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counter from the receive clock source the following conditions must be met:

1. 'KILL_CLOCK = 0' in the RESET_CNTL register.
2. 'RCV_EN = 1' or XMT_EN = 1 in the ENABLE register
3. 'GPCNT_CLKSET[1:0] = 10' in the CNTL register

The counter will begin to count at the receive clock rate as soon as these conditions are met.

To run the counter from the ETU (transmit) clock source the following conditions must be met:

1. 'KILL_CLOCK = 0' in the RESET_CNTL register.
2. Either one of the following:
3. "RCV_EN = 1' in the ENABLE register and 'CWT_EN = 1' in the CNTL register

   'XMT_EN = 1' in the ENABLE register
4. 'GPCNT_CLKSET[1:0] = 11' in the CNTL register

The counter will begin to count at the ETU clock rate as soon as these conditions are met.

The counter can be reset by setting GPCNT_CLKSEL[1:0] to 00. A 16-bit comparator value is provided that allows the software to select a count value at which an interrupt flag can be set and an interrupt generated if the mask is clear.

44.3.8 SIM LRC Block

The SIM module provides an 8-bit Linear Redundancy Check (LRC) generator / checker. The block is provided for use with T=1 SIM cards that support LRC. This block can be enabled through the LRCEN bit in the CNTL register. This block performs an 8-bit exclusive-OR on all received or transmitted characters. At the end of the reception of a block of characters, the result is expected to be 00. If so, the LRCOK bit is set in the RCV_STAT register. During transmission, the LRC block Exclusive-ORs each character that is transmitted with the current value of the LRC. If the XMT_CRC_LRC bit in the CNTL register is set, the LRC value will automatically be sent by the SIM transmitter as the final character when the transmit FIFO empties.

The LRC value can be reset in multiple ways. Clearing the LCREN bit in the CNTL register will reset the LRC value. At the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the LRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the LRC value.

44.3.9 SIM CRC Block

The SIM module provides an 16-bit Cyclic Redundancy Check (CRC) generator / checker. The block is provided for use with T=1 SIM cards that support CRC. This block can be enabled through the CRCEN bit in the CNTL register. This block performs a polynomial based check on all received or transmitted characters. The polynomial description is shown in Figure 4-17. The polynomial used is the standard CRC-CCITT where g(x) = x^16 + x^12 + x^5 + 1. The CRC register is initialized to all "1" before the data is shifted in. Before transmission the resulting CRC is inverted. For example, transmitting a 0xFA results in the following:

- Data = 0x5F (bit reversal of 0xFA)
- crc = 0x4AEA
- invert the crc = 0xB515 (bit reversal of 0xADA8
- data transmitted = 0xFA, 0xAD, 0xA8

See the figure below for an illustration of the SIM CRC block.

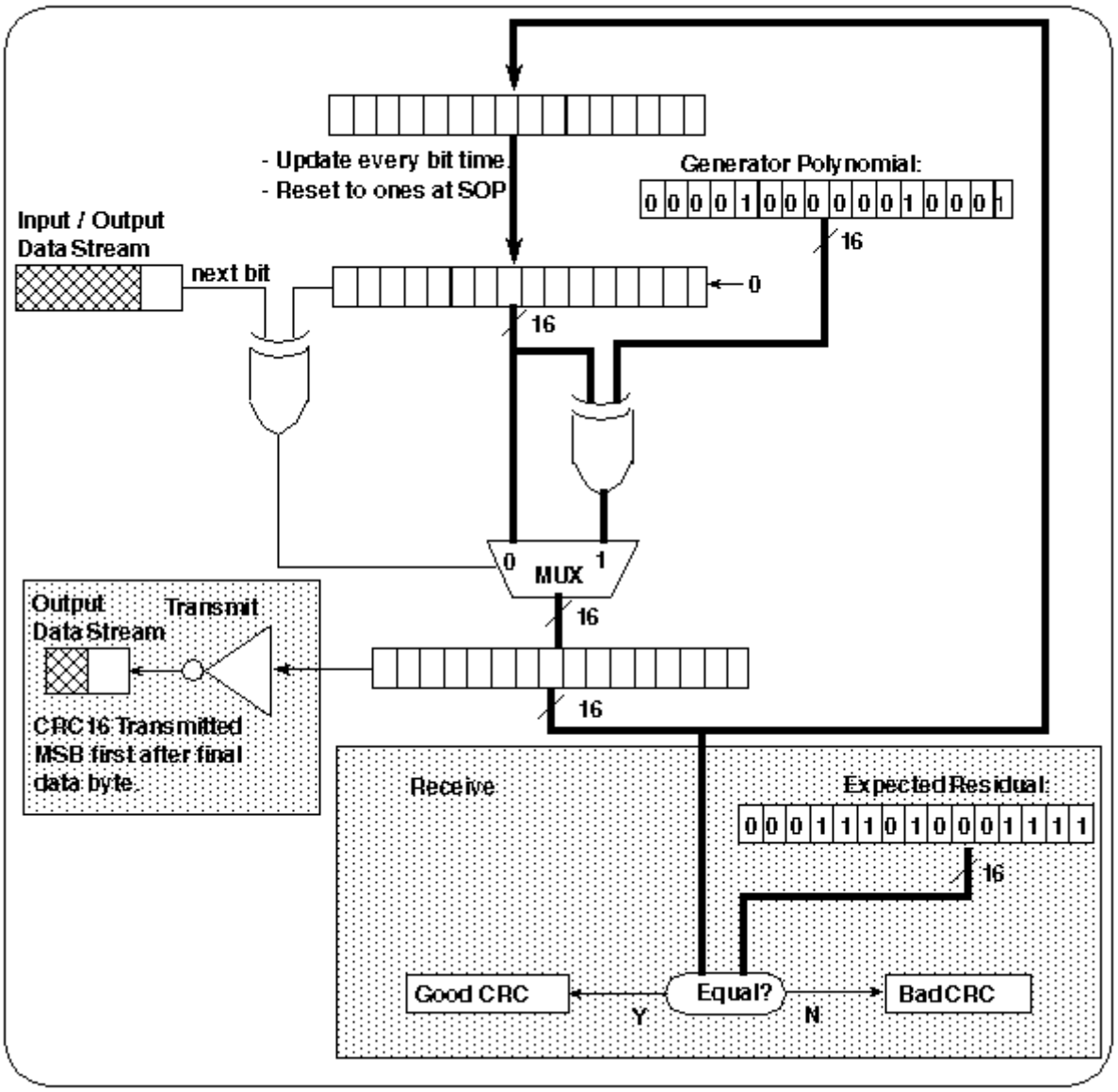


Figure 44-20. CRC Block Diagram

At the end of the reception of a block of characters, the residual from the CRC calculation is compared to 0x1D0F. If so, the CRCOK bit is set in the RCV_STAT register. During transmission, the CRC block updates the current value of the CRC residual using each character. If the XMT_CRC_LRC bit in the CNTL register is set, the CRC value will automatically be inverted and sent by the SIM transmitter as the final two characters when the transmit FIFO empties.

The CRC value can be reset in multiple ways. Clearing the CRCEN bit in the CNTL register will reset the CRC value. At the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the CRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the CRC value.

44.3.10 Module Interrupts

See the table below for the list of all possible interrupt sources and their corresponding mask bits. All SIM interrupts are logically ORed to create the active low irq_n and data_irq_n signals that go to the ITC module. All mask bits are such that a logic 1 implies that the corresponding interrupt is disabled (masked), whereas a 0 implies that the interrupt enabled (unmasked). All of these mask bits are logic 1 out of reset (all interrupts are masked).

Table 44-7. SIM Module Interrupts

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit Complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early Transmit Complete
TFE	XMT_STATUS)	TFEIM	INT_MASK	Transmit FIFO Empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit Threshold Error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill error
TDTF	XMT_STATUS	TDTFM	INT_MASK	Transmit Data Threshold Flag
GPCNT	XMT_STATUS	GPCNTM	INT_MASK	General Purpose Counter Comparator Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive data register full (FIFO threshold level reached)
OEF	RCV_STATUS	OIM	INT_MASK	Overrun Error Flag
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time Counter Comparator Flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time Counter Comparator Flag
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time Counter Comparator Flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive NACK Threshold Error
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM Detect Interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT)	SIM Detect Interrupt for port 0

44.4 Initialization/Application Information

This section describes the intended programming model for using the SIM module. The section describes how to begin a typical mode of operation using the registers.

44.4.1 Configuring SIM for Operation

The following is a list of items that need to be performed in order to configure the SIM module for operation:

1. Port selection in the SETUP register.
   a. Select which SIM module port is active by writing the SPS bit.
   b. Select whether the second SIM module port is active at the same time under the control of an alternate SIM module by writing the AMODE bit.
2. Enable the selected port (for port 1, see PORT1_CNTL register; for port 0, see PORT0_CNTL) following the SIM power up procedure specified in ISO 7816.
   a. Program 3VOLT0 pin of PORT0_CNTRL register to 1.
   b. Enable power to the SIM card using the SVEN1 or SVEN0 bit.
   c. Enable SIM module transmit data output using the STEN1 or STEN0 bit. This is required to allow the SIM receiver to create NACK pulses.
   d. Enable SIM card clock using the SCEN1 or SCEN0 bit.
   e. Release the SIM card reset using the SRST1 or SRST0 bit (assert SRSTx high).
3. Select XMT port driver type (open-drain or push-pull) in the OD_CONFIG register.
   a. Configure the selected port to be open drain or push-pull by using the OD_P1 or OD_P0 bit.
4. Choose the port baud rate in the CNTL register.
   a. Select the SIM card clock rate by using the CLK_PRESCALER[7:0] register.
   b. Select the SIM card baud rate by using the BAUD_SEL[2:0] bits and SAMPLE12.

NOTE

Follow the ISO 7816 spec with regards to card clock frequencies to ensure the maximum frequency specification is not violated.

5. Select Data format type, or place SIM receiver in initial character mode.
   a. Select inverse convention or direct convention by using the IC bit in the DATA_FORMAT register, or
   b. Enable initial character mode by using the ICM bit in the CNTL register.

44.4.1.1 Configuring SIM Receiver

The following is a list of items that need to be performed in order to configure the SIM receiver for operation:

1. Enable NACK capability in CNTL register.

a. Select NACK generation on parity errors, or invalid initial character by using the ANACK bit.
b. Select NACK generation on overrun conditions by using the ONACK bit.

2. Select the desired Receive NACK threshold and receive FIFO threshold in RCV_THRESHOLD register.
   a. Program the threshold at which the RDRF flag will be set by using the RDT bits.
   b. Program the threshold at which the RTE flag will be set by writing to the RTH[3:0] bits. If an auto power down after RTE flag is set, is desired, then enable the SIM card auto power down by setting the SAPD0,1 bits in the port0,1_cntl register.
3. Configure the Character Wait Time Counter, the Block Wait Timer Counter and the Block Guard Time Counter.
4. Enable interrupts in the INT_MASK register.
   a. Enable the receive data register full interrupt by using the RIM bit
   b. Enable the receive threshold error interrupt by using the RTM bit.
   c. Enable the overrun condition interrupt by using the OIM bit.
   d. Enable the Character Wait Time interrupt by using the CWTM bit
5. Enable enforce reception early stop function in ENABLE register.
   a. Enable the enforce reception early stop enable by using the ESTOP_EN bit.
   b. Enable the enforce reception early stop execution by using the ESTOP_EXE bit.

44.4.1.2 Configuring SIM Transmitter

The following is a list of items that need to be performed to configure the SIM transmitter for operation:

1. Select desired re-transmission threshold for NACKed characters in XMT_THRESHOLD register.
   a. Program the threshold at which the XTE flag will be set by using the XTH[3:0] bits.
2. Select the guard time between transmissions in the GUARD_CNTL register.
   a. Program the desired guard time between characters transmitted by the SIM module by using the GETU[7:0] bits.
3. Select the desired Transmit FIFO threshold level in the XMT_THRESHOLD register.
   a. Program the desired threshold using the TDT[3:0] bits.
4. Enable interrupts in the INT_MASK register.
   a. Enable the transmit complete interrupt by using the TCIM BIT.
   b. Enable the early transmit complete interrupt by using the ETCIM bit.
   c. Enable the transmit FIFO empty interrupt by using the TFEIM bit.
   d. Enable the transmit threshold error interrupt by using the XTM bit.

e. Enable the transmit FIFO threshold interrupt by using the TDTFM bit.
f. Enable the transmit FIFO overfill interrupt by using the TFOM bit.

5. Enable NACK delay detection in ENABLE register.
   a. Enable the NACK delay detection enable by using the NACK_DD_EN bit.

44.4.1.3 Configuring SIM General Purpose Counter

The following is a list of items that need to be performed in order to configure the SIM General Purpose Counter for operation:

1. Select desired clock source for the General Purpose Counter using the CNTL register.
   a. Use the GPCNT_CLKSEL[1:0] bits to select the desired clock source for the counter
2. Program counter comparator using the GPCNT register.
   a. Use the GPCNT[15:0] bits to select the desired count value at which the GPCNT interrupt flag will be set.
3. Enable the selected clock source for the General Purpose Counter using either the RESET_CNTL or ENABLE registers.
   a. If the GP Counter is configured for the card clock, enable the clock by clearing the KILL_CLOCK bit in the RESET_CNTL register.
   b. If the GP Counter is configured for the receive oversample clock, enable this clock by setting the RCV_EN bit or XMT_EN bit in the ENABLE register.
   c. If the GP Counter is configured for the transmit oversample clock, enable this clock by setting the XMT_EN bit in the ENABLE register.
4. Enable interrupts in the INT_MASK register.
   a. Enable the general purpose counter interrupt by using the GPCNTM bit.

44.4.1.4 Configuring SIM to Measure WWT (Work Wait Time) for Type=0 Smartcards

The following is a list of items that need to be performed in order to configure the SIM to measure work wait time. This is intended for type=0 SmartCards. Work wait time is a combination of BWT and CWT. The CWT timer is to measure the time between the start bits of two consecutive characters in the block received from the SmartCard. If this time is larger that the value programmed in the cwt[15:0] register, then a flag will be set and an interrupt generated. This timer can be used for both type=0 and type=1 SmartCards. The BWT and BGT timer are used to measure the delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block sent by the card. If you want the SIM to enforce a WWT of 100 bits.

Then, you must activate both the CWT and BWT, and program both of them for a value of 100 bits. When measuring WWT, the BGT timer will not be used so it will be masked (inactive) by the BGTM bit. Below is the sequence to program the SIM to send and receive data while checking WWT.

1. Program both the CWT and the BWT (low and high) registers to the WWT (work wait time) value that needs to be enforced. Set BGT register to 0 (this is the default value).
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT and BWT interrupts by clearing the CWTM and BWTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a CWT or a BWT interrupt occurs, then the software should consider that a WWT violation. The software should clear the CWT or BWT interrupt by writing a "1" to the BWT or CWT bit in the RCV_STATUS register.

44.4.1.5 Configuring SIM to measure CWT, BWT, BGT for type=1 SmartCards

The following is a list of items that need to be performed in order to configure the SIM to measure CWT, BWT, BGT. This is intended for type=1 SmartCards. The CWT timer is to measure the time between start bits in the consecutive characters received from the SmartCard. If this time is larger that the value programmed in the CWT[15:0] register, then a flag will be set and an interrupt generated. This timer can be used for both type=0 and type=1 SmartCards. The BWT, BGT timer is used to measure the delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block sent by the card. If this time is larger than the value

in the both BWT registers (BWT and BWT_H), then the BWT flag will be set and an interrupt generated. If the time is less than the value in the BGT register, then the BGT flag will be set and an interrupt generated.

1. Program the CWT, BWT, BWT_H and BGT registers to the value that needs to be enforced.
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT, BWT, BGT interrupts by clearing the CWTM, BWTM, BGTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a CWT or a BWT interrupt occurs, then the software should read the RCV_STATUS register to determine if the error was caused by a CWT, BWT, or a BGT violation. The software should clear the CWT, BWT, BGT interrupt by writing a "1" to the CWT, BWT, BGT bit in the RCV_STATUS register.

44.4.1.6 Configuring SIM Linear Redundancy Check (LRC) Block

The following is a list of items that need to be performed in order to configure the SIM Linear Redundancy Check Block for operation:

1. Enable the LRC block by using the CNTL register.
   a. Use the LRCEN bit to enable the LRC block.
   b. Use the XMT_CRC_LRC bit to enable the transmission of the LRC Character after the last character in the Transmit FIFO is sent. Refer to the T=1 programming model for more details.

44.4.1.7 Configuring SIM Cyclic Redundancy Check (CRC) Block

The following is a list of items that must be performed to configure the SIM Cyclic Redundancy Check Block for operation:

1. Enable the CRC block by using the CNTL register.
   a. Use the CRCEN bit to enable the CRC block.
   b. Use the XMT_CRC_LRC bit to enable the transmission of the CRC Characters after the last character in the Transmit FIFO is sent. Refer to the T=1 Programming model for more details.

44.4.2 Using the SIM Receiver

Once the SIM has been properly configured (such as correct baud rate and correct data format), SIM receptions can be enabled by setting the receive enable bit, RCV_EN, in the ENABLE register. As bytes are received, they will be placed in the 288 byte deep receive data FIFO. Unread bytes can be accessed from this FIFO at any time. There is no need to disable the receiver to access the FIFO. The FIFO should only be read when the receive FIFO data flag, RFD, in the RCV_STATUS register is set. The RFD flag, which cannot create an interrupt, is high any time there is at least one unread byte in the receive FIFO. If the receive FIFO is read when RFD is low, it will simply produce the last byte read.

The receive data register full flag, RDRF, in the RCV_STATUS register can be used to determine when the receive FIFO has reached a given threshold value. This flag will create an interrupt if the RIM bit in the INT_MASK register is clear. To control at which point RDRF is set, program the receive data threshold, RDT, in the RCV_THRESHOLD register. If the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT, RDRF will be set.

NOTE

A value of 0x0 in RDT implies that there must be 288 unread bytes in the receive FIFO to trigger RDRF.

The standard flow for receiving bytes from the SIM card is to set RDT to the appropriate value, wait for RDRF to cause an interrupt (RIM clear), and then read bytes out of the receive FIFO as long as RFD is high. In addition to checking RFD between every byte, it is also recommended that software check for the existence of a set OEF flag as well.

44.4.2.1 Receive Parity Errors and Parity NACK Generation

The SIM receiver checks every byte received for proper parity. The IC control bit in the DATA_FORMAT register controls whether it checks for odd parity or even parity. When checking for odd parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be odd. Likewise, when checking for even parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be even.

If NACK generation on errors is disabled (ANACK = 0 in the CNTL register), the PORTx_PE bit will be set when the SIM receiver detects a parity error, the received data byte and its PORTx_PE bit are placed into FIFO. There is no need to clear the parity error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A parity error cannot cause an interrupt, and it is up to the software to discard data bytes with parity errors.

If NACK generation on errors is enabled (ANACK =1), the SIM can automatically request the SIM card to re-send a byte that has a parity error by generating a NACK pulse on the SIM XMT pin. In this case, Bytes with parity errors are discarded and will not placed into the FIFO.

To control NACK generation by the SIM receiver use the RTH[3:0], Receive NACK threshold, in the RCV_THRESHOLD register. This set of bits specify the number of consecutive NACK's generated by the SIM module on a received byte, before generating an RTE (receive threshold interrupt flag) in the RCV_STATUS register. The RTE flag would also force the SIM port to power down the card if the SAPD0,1 bit is set in the PORT0,1_CNTL register.

NOTE

The ANACK bit must be set in the CNTL register to enable this feature. The control bit ANACK is also used in initial character mode to enable the retransmission of initial characters in the event that an invalid initial character is received.

When a valid character has been received by the SIM, the internal counter keeping track of the number of NACK's transmitted on the current byte resets to zero. Clearing the receive threshold error (RTE) bit would also clear that counter.

When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

44.4.2.2 Receive Frame Errors

The SIM receiver checks every byte received for a proper stop bit. A stop bit should exist during at least the first half of the 11th ETU time after the start of the character. If this is not true, a frame error is flagged. When a frame error is detected on a given byte, the PORTx_FE bit for that byte is set in the FIFO. The PORTx_FE flag for each byte is read out of the FIFO when the data itself is read. There is no need to clear the frame error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A frame error cannot cause an interrupt, nor can it create a NACK pulse to the receiver asking for a retransmission of the corrupted data.

44.4.2.3 Receive Overrun Errors and Overrun NACK Generation

When there already exists 288 unread bytes in the FIFO, a received character will cause the SIM receiver to flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 288 unread bytes in the FIFO unaltered.

It is possible for the SIM to automatically request that the SIM card re-send the byte that caused the overrun condition by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse for the byte that caused the overrun condition if the control bit ONACK in the CNTL register is set. In this case, the existence of an OEF flag does not indicate the loss of data, but rather a NACK (that is, retransmission request) due to a full receive FIFO. As opposed to transmit NACK generation, there is no limit to the number of times an overrun condition will cause a NACK other than to disable ONACK itself. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

If the control bit ONACK is clear, the existence of a high OEF flag indicates the loss of data. The OEF flag can create an interrupt if the OIM bit in the INT_MASK register is clear.

To clear the OEF flag, software must simply write a "one" to the OEF bit position in the RCV_STATUS register. A high OEF flag has no effect on the operation of the SIM receiver other than to create an interrupt if OIM is clear.

44.4.2.4 Using Initial Character Mode and Resulting Receive Data Formats

The SIM receiver supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential values that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. Essentially, inverse convention differs from direct convention in that the order of the data is flipped msb for lsb, and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

To place the SIM into initial character mode, set the ICM bit in the CNTL register. Once a valid initial character is received, the IC bit in the DATA_FORMAT register will be set accordingly by the hardware, and the ICM bit will be cleared. Software can read the state of the IC bit to determine which mode the SIM is currently using.

The 0x3B (as decoded by direct convention) with parity bit high will cause direct convention to be used (IC set to logic 0), whereas a 0x3F (as decoded by inverse convention) with parity bit high will cause inverse convention to be used (IC set to logic 1).

When the receiver is in initial character mode, all received bytes will continue to be placed into the receive FIFO whether they be valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes will be decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if the IC bit is low, and the correct initial character for setting inverse convention is received, that character and all subsequently received characters will be stored in the FIFO after having been decoded using inverse convention (for example, the initial character will be stored as 0x3F).

If the receiver is in initial character mode (ICM is high), and an invalid initial character is received, the SIM can be configured to automatically request that the initial character be retransmitted by setting the ANACK control bit in the CNTL register. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETUs. The invalid initial character will be placed into the receive FIFO and marked with a parity error to signify that this is an invalid initial character.

44.4.2.5 Initial Character Mode Programming Notes

The usage of the initial character mode requires close attention to the programming model. There is a condition where a parity error in the initial byte for direct convention (0x3B) could be decoded as what appears to be a valid initial character for inverse convention (that is, 0x3F). The SIM module will not recognize this as a valid initial character for inverse convention and will mark the character by setting the parity error flag. The software must look for the existence of a parity error before recognizing a character as a valid initial character.

44.4.2.6 Automatic Receiver Mode

The SIM module has an automatic receive mode that inhibits the data being transmitted by the SIM module from entering the SIM receive buffer through the feedback path of the SIM data pin. The SIM module receiver should normally be enabled while the transmitter is operational. Automatic receive mode saves the software from having to actively manage the transition from transmitter to receiver. The auto receive mode is always active whenever the receiver is enabled.

44.4.2.7 Using the SIM Receiver with "T=1" SIM Cards

The SIM module provides hardware support for "T=1" type SIM cards. These type of cards present several requirements above and beyond the standard "T=0" cards. The features provided to meet the requirements that pertain to the SIM receiver are as follows:

- 11 ETU Characters
  - "T=1" cards can transmit with character lengths of 11 ETUs (that is, one STOP bit). The SIM module provides the RCVR11 bit in the GUARD_CNTL register in order to configure the receiver state machine to accept 11 ETU characters.
- Character Wait Time Counter
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters received from the SmartCard. The value of CWT can range from 12 ETU to 32779 ETU. The SIM module provides a 16-bit counter with programmable comparator clocked at the ETU bit rate to identify when the CWT has been exceeded by the SIM card.
- Block Waiting Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The block wait time must not exceed a value that is programmable. The SIM module provides a 32-bit Block Wait Timer that can be used to identify when the BWT has been violated (divided in two registers).

- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The block guard must be greater than a value that is programmable. The SIM module provides a 16-bit Block Guard Timer that can be used to identify when the BGT has been violated.
- Error Detection Code
  - "T=1" cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operations.

44.4.3 Using the SIM Transmitter

Once the SIM has been properly configured (such as data XMT enabled, correct baud rate, and correct data format), the transmitter can be enabled by setting the XMT_EN bit in the ENABLE register. If data had been previously written to the transmit FIFO, the transmitter will begin to send the first character. If no data is written to the transmit FIFO before enabling the transmitter, then the transmitter will wait until the first character is written before beginning transmission. Clearing the XMT_EN bit while the transmitter is in operation, will halt any transmission in progress, flush the transmit FIFO, and reset the transmit state machine.

Data can be written to the transmit FIFO at any time.

The transmit data threshold flag, TDTF, in the XMT_STATUS register can be used to determine when the transmit FIFO has reached a given threshold value. This flag will create an interrupt if the TDTFM bit in the INT_MASK register is clear. To control at which point TDTF is set, program the transmit data threshold, TDT[3:0], in the XMT_THRESHOLD register. If the number of bytes remaining in the transmit FIFO is equal to or less than the value set by TDT[4:0], TDTF will be set.

NOTE

A value of 0x0 in TDT[3:0] implies that the transmit FIFO must be empty to trigger TDTF.

The value in TDT[3:0] can be changed at any time to alter this threshold level. The comparison between the number of remaining bytes in the transmit FIFO and the value set by TDT[3:0] is continuously updated so that any change in either will be immediately reflected in the state of TDTF. Unlike the RDRF flag for the receive FIFO, TDTF is latched and remains set until the software writes a 1 to the TDTF bit location in the XMT_STATUS register. For instance, if TDT[3:0] is set to 5, and there are 6 bytes remaining in the FIFO, changing TDT[3:0] to 6 will immediately cause TDTF to be set. However, setting TDT[3:0] back to 5 will not cause TDTF to clear.

The standard flow for transmitting bytes from the SIM card is to set TDT[3:0] to the appropriate value, write up to 16 bytes to the transmit FIFO, enable the transmitter, wait for TDTF to cause an interrupt (TDTFM clear), and then write additional bytes to the transmit FIFO.

NOTE

For the SIM module to transmit successfully, the transmit pin must be connected to the receive pin of the same port. This connection is necessary for the SIM to decode transmit NACKs sent to it by the SIM card. Without this connection, all transmissions will appear to have a NACK thereby causing the first byte to be transmitted continuously. When operating in external one wire interface mode (3VOLT0 or 3VOLT1), this connection is made internal to the SoC.

44.4.3.1 Transmit Data Formats

There are two possible data formats that the SIM module uses when transferring data to the SIM: card - inverse convention or direct convention. The format used depends on the state of inverse convention bit (IC in the DATA_FORMAT register). Software can set the IC bit, or it can be set automatically by hardware when using the initial character mode.

44.4.3.2 Transmit NACK

The SIM transmitter can respond to NACKs created by the SIM card. A NACK will be decoded if the SIM card creates a logic low level on the SIM receive pin during the STOP bit time at the end of a transmitted byte. To prevent a situation where the SIM interface is stalled by an infinite number of NACK pulses on a given byte, the SIM module can be configured to limit the number of times it will respond to NACKs. The XTH[3:0] control field in the XMT_THRESHOLD register allows software to set a threshold on the number of times a given byte will be retransmitted. If the threshold is reached, a transmit threshold error, XTE in the XMT_STATUS register, is asserted.

When XTE is set, the SIM transmitter is halted, and all pending transfers are aborted, and the TC, ETC, AND TFE flags are set. All bytes remaining in the transmit FIFO are lost. There is no way to restart the transmission on the next byte in the FIFO. The transmitter remains frozen until XTE is cleared by software. The only way to clear XTE is to write a 1 to the XTE bit in the XMT_THRESHOLD register. The XTE flag can create an interrupt if the XTM mask in the XMT_THRESHOLD register is clear.

It is possible to disable the detection of NACKs from the SIM card by setting XTH[3:0] to 0x0. By setting XTH[3:0] to 0x1, it is possible to disable all retransmissions while still setting XTE on the first NACK received. In general, XTE is set on the NACK that causes the threshold set by XTH[3:0] to be reached. This final NACK will not cause a retransmission, whereas all previous NACKs will cause a retransmission.

44.4.3.3 Transmit Guard Time

The time between data bytes sent from the SIM transmitter can be altered using the transmit guard time control. By default, the minimum time between start bits of successive transmitted bytes is 12 ETUs (Elementary Time Units). An ETU is equivalent in time to 1 bit time. Therefore, the amount of time an ETU consumes is determined by the baud rate chosen. The 12 ETUs that form the default character transmission time consist of a start bit, 8 data bits, a parity bit and two stop bits. The number of stop bits (idle bits) can be extended by an integer number of ETUs. The number of additional ETUs can be programmed directly into the GETU[7:0] bits of the GUARD_CNTL register. Programming the GETU[7:0] bits to 0xFF will configure the SIM transmitter to use only one stop bit for each character transmission.

44.4.3.4 Using SIM Transmit with "T=1" SIM Cards

The SIM module provides hardware support for "T=1" type SIM cards. These type of cards present several requirements above and beyond the standard "T=0" cards. The features provided to meet the requirements that pertain to the SIM transmitter are as follows:

- 11 ETU Characters
  - The SIM module transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 (0xFF) in the GETU[7:0] bits in the GUARD_CNTL register will set the number of ETUs per character transmitted to 11.
- Character Waiting Time
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The time between transmitted characters is controlled by the programmable guard time in the GUARD_CNTL register. However, the time between the last byte in the transmit FIFO, and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The SIM transmitter provides a Transmit FIFO threshold (TDTF) interrupt to signal the

system when the expected number of characters have been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the TDTF interrupt and write new data to the transmit FIFO before the last character in the Transmit FIFO has been sent.

- Block Waiting Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The SIM transmitter provides a General Purpose Counter that can be used to track the BWT. The BWT is purely determined by software response time to the transmit interrupts.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BGT is 22 ETU. The SIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after the XMT_EN bit is set. The BGT will be determined by the speed at which the software can react to an interrupt and enable the transmitter.
- Error Detection Code
  - "T=1" cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operation.

44.4.4 Suggested "T=1" Compliant Programming Model

This section describes the suggested programming model for supporting "T=1", "T=0", and known "special" cards using the SIM module on the device. This should be used as a rough guide for how to configure the SIM module for use with SIM cards specified by ISO 7816-3 and EMV. Some details are not addressed. Other uses for some of the SIM features are not included (for example, GP Counter uses for some ISO timing requirements).

44.4.4.1 Answer To Reset (ATR) Detection

The first step to communicating with a SIM card is to provide power and a clock signal to the card. Once the card is detected as present (using the presence detect features, or some other method), the SIM card should be powered up according to the power up sequence specified in the ISO 7816-3 specification.

1. Apply voltage to the SIM card by setting the SVEN0 or SVEN1 bit in the PORTx_CNTL register
2. Select the appropriate clock frequency for the SIM card by programming the CLK_PRESCALER[7:0] register.
3. Enable the clock to the SIM card by setting the SCEN0 or SCEN1 bit in the PORTx_CNTL register.
4. Remove the card from reset by setting the SRST0 or SRST1 bit in the PORTx_CNTL register.

   The first communication between the SIM card and the SIM module will be a block of data sent from the SIM card to the SIM module after the card is powered and the card reset is removed. This block is called the Answer To Reset (ATR). To receive the ATR, the SIM module should be configured for 12 ETU character reception. According to the ISO 7816-3 spec, both "T=0" and "T=1" cards will communicate initially using 12 ETU character durations.

NOTE

We are aware of some card manufacturers that communicate at 11.5 ETU character durations (Geldkarte). This will complicate the initial card detection sequence shown below.

5. Clear RCVR11 bit in the GUARD_CNTL register

   The next item to configure for ATR reception is the NACK capability of the SIM module. The ISO 7816-3 spec allows the SIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

NOTE

The Europay Mastercard and VISA (EMV) cards are similar to "T=1", but do not allow the SIM module to NACK during the initial communication. This will again complicate the initial card detection sequence shown below.

6. Set ANACK bit in the CNTL register to enable NACK generation

   In order for the SIM module to notify when characters are received, the receive interrupt(s) can be enabled. A threshold can be set for the number of characters to receive before generating an interrupt.

7. Enable RDRF and OEF interrupts by setting RIM and OIM bits in INT_MASK register
8. Set desired threshold for received characters by writing RDT in RCV_THRESHOLD register.

The SIM should be setup to perform in initial character mode. This will cause the hardware to identify the first valid character sent during the ATR as an initial character. This character will automatically configure the hardware for the data convention used by the SIM card.

9. Set Initial Character Mode by setting the ICM bit in the CNTL register

   The ISO 7816-3 spec requires that SIM cards meet certain timing restrictions. One of these is the time from the deassertion of the card reset to the beginning of the ATR sequence. The SIM module General Purpose Counter can be used to verify that the SIM card begins its ATR within the 400 to 40,000 clock cycle range.

1. Set General Purpose Counter Comparator to 0x9C40 using GPCNT register.
2. Enable General Purpose Counter Interrupt by clearing GPCNTM in INT_MASK register.
3. Enable General Purpose Counter by programming the GPCNT_CLKSEL[1:0] bits to 01 so the card clock is used for counting.

The ISO7816-3 spec states that the maximum allowed time between two characters during the ATR is 9600 ETUs (Initial Waiting Time). The Character Wait Time Counter should be setup to detect any errors for this condition.

1. Set Character Wait Time Counter Comparator to 9600 using the CHAR_WAIT register
2. Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register
3. Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register

   The last step in preparing for ATR reception is to enable the receiver.

4. Set RCV_EN bit in ENABLE register

The SIM module will generate interrupts once a threshold number of characters is received. The software should react to these interrupts and read the characters from the receive FIFO (RCV_BUF) until the complete ATR has been received. If a General Purpose Counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 spec. Otherwise, once a valid ATR is received, the software will know from the ATR information the specific characteristics for this card (refer to the ISO 7816-3 spec for details).

44.4.4.2 Programming Considerations for Geldkarte Cards

There is at least one manufacturer of SIM cards (Geldkarte) that we know of that does not send the ATR in 12 ETU mode or support NACKs. This creates an issue with detecting a valid ATR from a SIM card. Since any kind of card can be attached to the SIM module, the software and hardware do not know how to begin communication. Basically, if the card fails to send a valid ATR on the first try, disable NACKs, configure the SIM module for 11 ETU and try again. The software should toggle between 12 and 11 ETU modes with and without NACKs enabled until a valid ATR is received, or the number of attempts to communicate passes a predetermined error threshold. The figure below shows the flow chart for the suggested Geldkarte Compliant SIM Initialization.

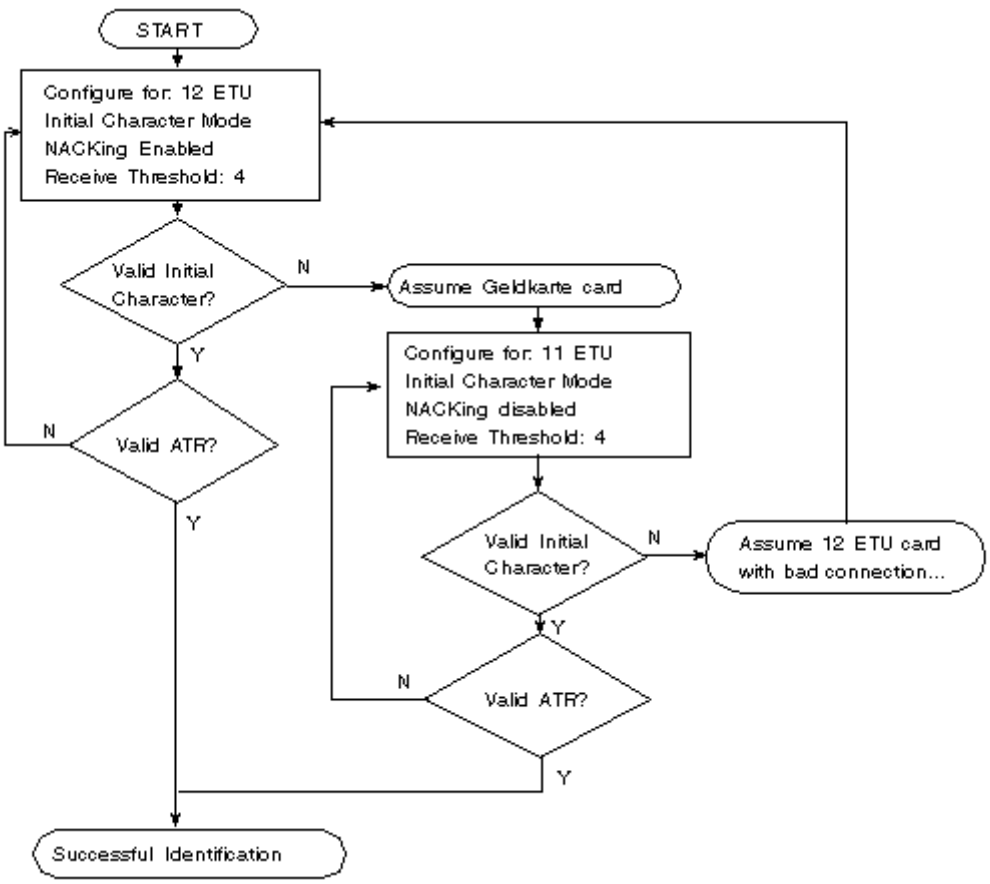


Figure 44-21. Suggested "T=1", EMV, Geldkarte Compliant SIM Initialization

44.4.4.3 Programming Considerations for T=0 SIM Cards

If the card is of type T=0, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register
2. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register
3. Adjust NACK capability by modifying the values of the ONACK and ANACK bits in the CNTL register
4. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register
5. Adjust the level of transmit NACK re-transmissions allowed by modifying the value of the XTH[3:0] bits in the XMT_THRESHOLD register
6. Adjust the level for the Receive NACK threshold by modifying the RTH[3:0] bits in the RCV_THRESHOLD register.

If a negotiation with the SIM card is desired, the software sends a PPS response to the SIM card. To send the response, the following steps should be performed:

1. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register
2. Write the characters to be sent as response (max 16) to the transmit FIFO using the XMT_BUF register
3. Clear all transmit interrupt flags in the XMT_STAT register by writing a one to them
4. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
5. Enable the transmitter by setting the XMT_EN bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=0 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

44.4.4.4 Programming Considerations for T=1 SIM Cards

If the card is of type T=1, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register
2. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register. Setting GETU[7:0] to 0xFF configures the SIM transmitter for 11 ETU transmissions
3. Disable NACK capability by clearing the ONACK and ANACK bits in the CNTL register. T=1 cards do not allow NACKs.
4. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register
5. Set Character Wait Time Counter Comparator to value specified in the ATR by using the CHAR_WAIT register
6. Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register
7. Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register
8. Enable CRC or LRC error checking according to the ATR information by setting either the CRCEN or LRCEN bit in the CNTL register. These bits will never be set at the same time!

   For T=1 cards, the ATR is sent using a T=0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the SIM card is desired, the software will send a PPS response to the SIM card. Otherwise, the protocol is initiated with a block transfer from the SIM module. In order to send the response or the first block, the following steps should be performed:

9. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register
10. Write the characters to be sent as response (max 16) to the transmit FIFO using the XMT_BUF register
11. Clear all transmit interrupt flags in the XMT_STAT register by writing a 1 to them
12. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
13. Enable the transmission of the error checking characters (LRC or CRC) by setting the XMT_CRC_LRC bit in the CNTL register.

NOTE

If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is completed. If so, do not set the XMT_CRC_LRC bit during the PPS exchange.

14. Enable the transmitter by setting the XMT_EN bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=1 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

44.5 SIM Memory Map/Register Definition

SIM memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SIM Port1 Control Register (SIM_PORT1_CNTL)	32	R/W	0000_0000h	44.5.1/2853
4	SIM Setup Register (SIM_SETUP)	32	R/W	0000_0000h	44.5.2/2855
8	SIM Port 1 Detect Register (SIM_PORT1_DETECT)	32	R/W	0000_0001h	44.5.3/2855
C	SIM Transmit Buffer Register (SIM_XMT_BUF)	32	R/W	0000_0000h	44.5.4/2857
10	SIM Receive Buffer Register (SIM_RCV_BUF)	32	R	0000_0000h	44.5.5/2858
14	SIM Port0 Control Register (SIM_PORT0_CNTL)	32	R/W	0000_0000h	44.5.6/2859
18	SIM Control Register (SIM_CNTL)	32	R/W	0000_0006h	44.5.7/2861
1C	SIM Clock Prescaler Register (SIM_CLK_PRESCALER)	32	R/W	0000_0002h	44.5.8/2863
20	SIM Receive Threshold Register (SIM_RCV_THRESHOLD)	32	R/W	0000_0001h	44.5.9/2864
24	SIM Enable Register (SIM_ENABLE)	32	R/W	0000_0000h	44.5.10/ 2865
28	SIM Transmit Status Register (SIM_XMT_STATUS)	32	w1c	0000_00B8h	44.5.11/ 2867
2C	SIM Receive Status Register (SIM_RCV_STATUS)	32	w1c	0000_0042h	44.5.12/ 2869
30	SIM Interrupt Mask Register (SIM_INT_MASK)	32	R/W	0000_3FFFh	44.5.13/ 2871
3C	SIM Port0 Detect Register (SIM_PORT0_DETECT)	32	R/W	0000_0001h	44.5.14/ 2873
40	SIM Data Format Register (SIM_DATA_FORMAT)	32	R/W	0000_0000h	44.5.15/ 2875
44	SIM Transmit Threshold Register (SIM_XMT_THRESHOLD)	32	R/W	0000_0000h	44.5.16/ 2875
48	SIM Transmit Guard Control Register (SIM_GUARD_CNTL)	32	R/W	0000_0000h	44.5.17/ 2876

Table continues on the next page...

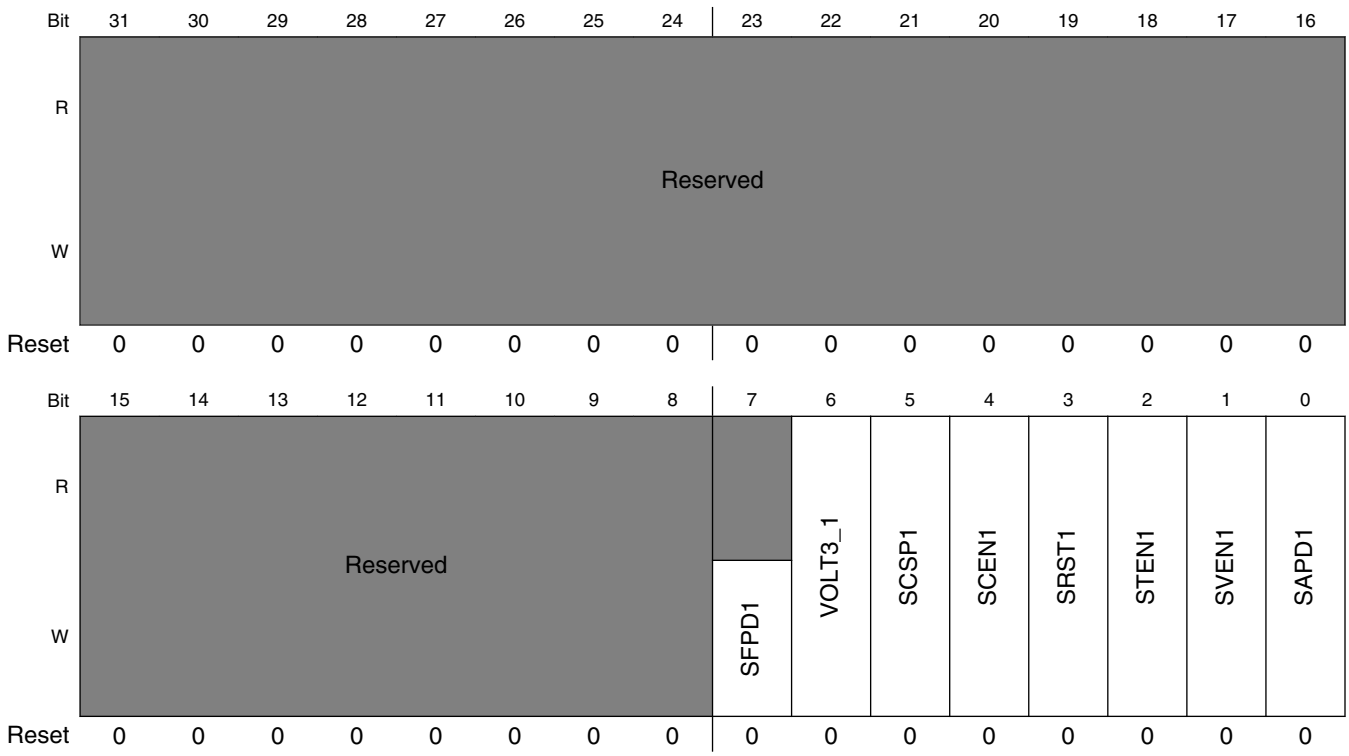
SIM memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4C	SIM Open Drain Configuration Control Register (SIM_OD_CONFIG)	32	R/W	0000_0000h	44.5.18/ 2877
50	SIM Reset Control Register (SIM_RESET_CNTL)	32	R/W	0000_0000h	44.5.19/ 2878
54	SIM Character Wait Time Register (SIM_CHAR_WAIT)	32	R/W	0000_FFFFh	44.5.20/ 2879
58	SIM General Purpose Counter Register (SIM_GPCNT)	32	R/W	0000_FFFFh	44.5.21/ 2880
5C	SIM Divisor Register (SIM_DIVISOR)	32	R/W	0000_00FFh	44.5.22/ 2880
60	SIM Block Wait Time Register (SIM_BWT)	32	R/W	0000_FFFFh	44.5.23/ 2881
64	SIM Block Guard Time Register (SIM_BGT)	32	R/W	0000_0000h	44.5.24/ 2881
68	SIM Block Wait Time Register HIGH (SIM_BWT_H)	32	R/W	0000_FFFFh	44.5.25/ 2882
6C	SIM Transmit FIFO Status Register (SIM_XMT_FIFO_STAT)	32	R	0000_0000h	44.5.26/ 2883
70	SIM Receive FIFO Counter Register (SIM_RCV_FIFO_CNT)	32	R	0000_0000h	44.5.27/ 2883
74	SIM Receive FIFO Write Pointer Register (SIM_RCV_FIFO_WPTR)	32	R	0000_0000h	44.5.28/ 2884
78	SIM Receive FIFO Read Pointer Register (SIM_RCV_FIFO_RPTR)	32	R	0000_0000h	44.5.29/ 2884

44.5.1 SIM Port1 Control Register (SIM_PORT1_CNTL)

See the figure below for illustration of valid bits in the SIM Port1 Control Register and the table below for description of the bit fields.

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



SIM_PORT1_CNTL field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 SFPD1	Auto Power Down port1. Writing a "1" to this location will start the auto power down sequence for port 1. This bit will autoclear. A read of this bit will give a "0". 0 No effect 1 Start Auto Power down
6 VOLT3_1	External one wire interface for SIM Card port1. Used to configure the Port 1 transmit pin as bi-directional. This allows the Port 1 Receive pin to be re-used as General Purpose I/O. This operation is restricted to bi-directional data SIM cards only. This bit should not be changed while the receiver or transmitter is enabled! 0 Port1 uses both rcv and xmt pins 1 Port1 XMT pin bidirectional. Port1 RCV PIN unused

Table continues on the next page...

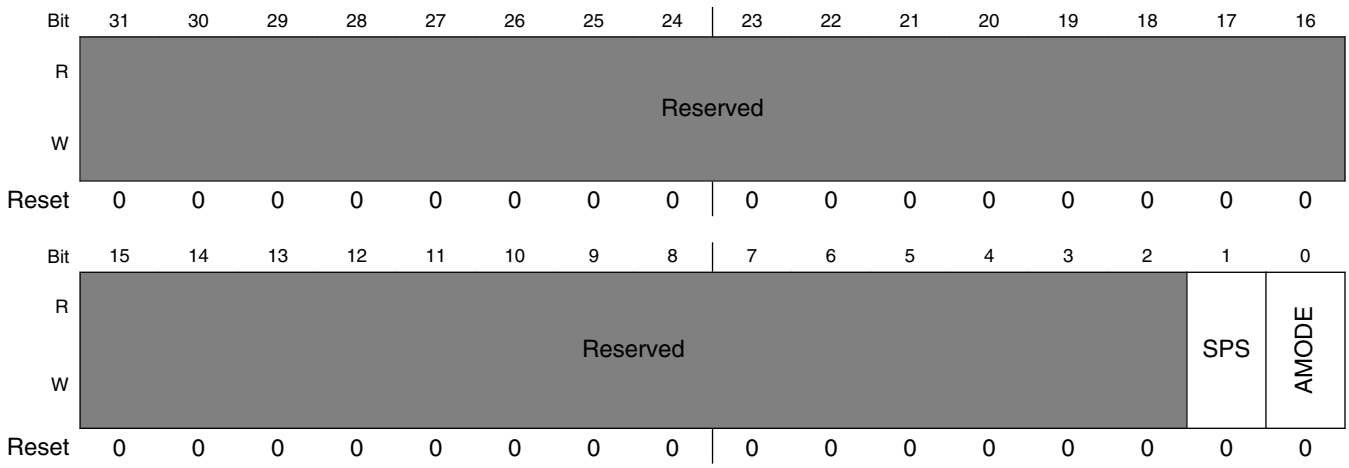
SIM_PORT1_CNTL field descriptions (continued)

Field	Description
5 SCSP1	SIM Card Clock Stop Polarity Port1. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN1. It will be forced low by hardware during the auto power down sequence. This forces the clock be a logic 0 when stopped by auto power down as required by ISO 7816 spec. 0 Clock is logic 0 when stopped by SCEN1 1 Clock is logic 1 when stopped by SCEN1
4 SCEN1	SIM card Clock Enable Port 1. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 SIM Card Clock Disabled Port1 1 SIM Card Clock Enabled Port1
3 SRST1	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port1 inactive (default) 1 SIM Card reset Port1 active
2 STEN1	SIM card Transmit Enable Port 1. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port1 Transmit Data is forced to zero (default) 1 Port 1 Transmit Data controlled by SIM module
1 SVEN1	SIM card Vcc Enable Port 1. Used to control the state of the SVEN1 pin on SIM card port 1. The SVEN1 pin controls the SIM card V_{CC} enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 1 disabled (default) 1 SIM card Voltage Port 1 enabled
0 SAPD1	SIM card Auto Power Down Port 1. Used to enable/disable the auto power down function for port 1. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 1 disabled (default) 1 Auto power down Port 1 enabled

44.5.2 SIM Setup Register (SIM_SETUP)

See the figure below for illustration of valid bits in the SIM Setup Register and the table below for description of the bit fields.

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



SIM_SETUP field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
1 SPS	**SIM card Port Select.** Controls which port the SIM interface uses. **NOTE:** The AMODE bit must be zero when the SPS bit is set to 1. 0 Port 0 Enabled (default) 1 Port 1 Enabled
0 AMODE	**Alternate SIM Card Mode enable.** Enables an alternate SIM module to control SIM card Port 1. **NOTE:** The SPS bit must be 0 to give the alternate SIM module control. 0 Alternate Port Disabled (default) 1 Alternate Port Enabled

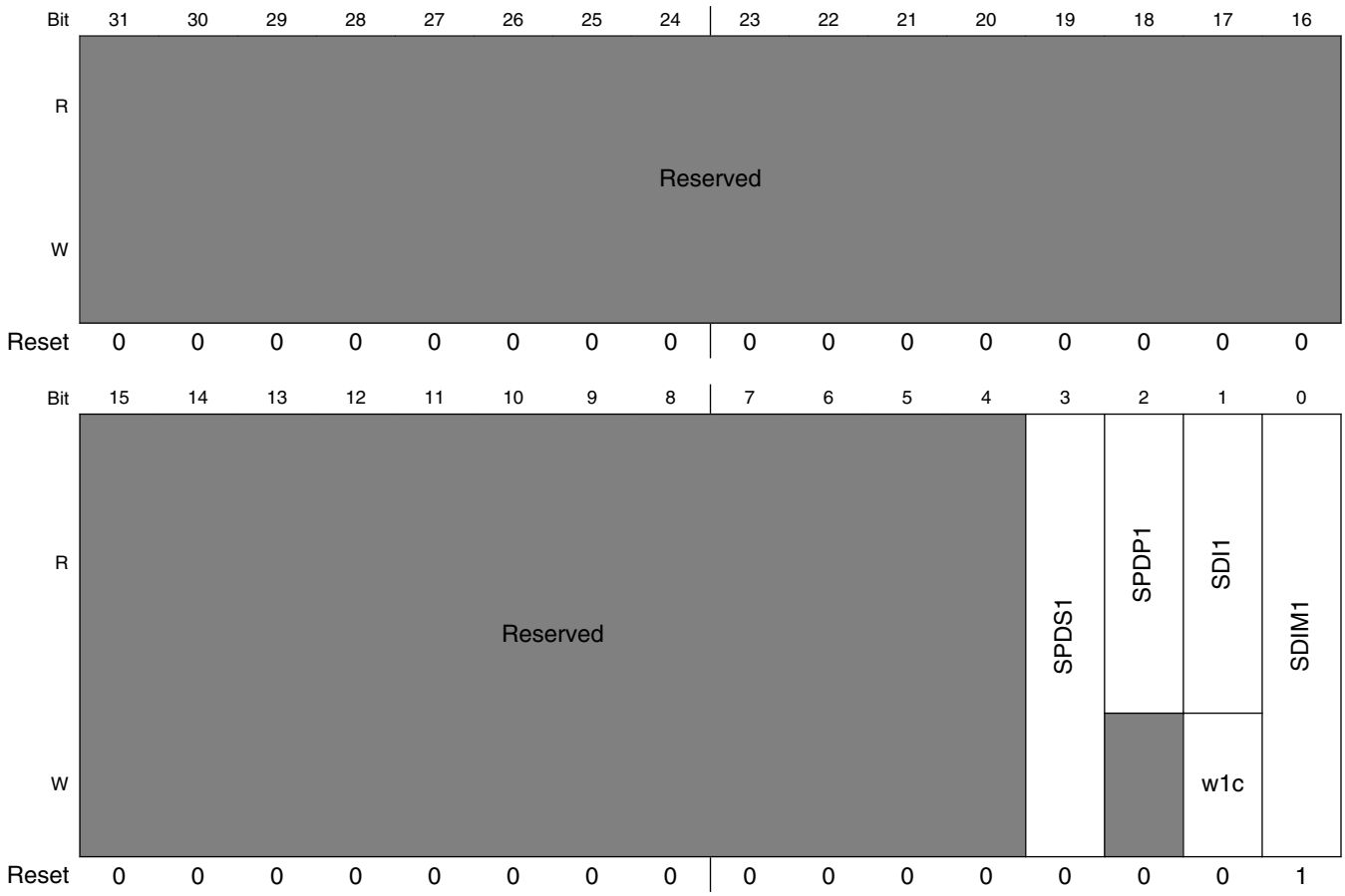
44.5.3 SIM Port 1 Detect Register (SIM_PORT1_DETECT)

See the figure below for illustration of valid bits in the SIM Port1 Detect Register and the table below for description of the bit fields.

Summary:

spds1 determines which edge transition of the SIMPD1 pin is used for SIM card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the SIMPD1 edge specified by spds1 will cause the following: sdi1 to be set; if the sdim1 mask is clear, an interrupt on SIMIRQ_N; and if sapd1 in the PORT1_CNTL register is set, an auto power down sequence to begin. If SIM card insertion is expected, sapd1 can be set low to avoid the auto power down sequence. There is no auto power up sequence. The bit spdp1 can be used to determine the current state of the SIMPD1 pin.

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



SIM_PORT1_DETECT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3 SPDS1	SIM **Presence Detect Select Port 1.** Controls which edge of the SIMPD1 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD1 Input (default) 1 Rising edge of SIMPD1 Input
2 SPDP1	**SIMPD1 input pin status.** This bit reflects the state of the SIMPD1 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD1 pin itself.

Table continues on the next page...

SIM_PORT1_DETECT field descriptions (continued)

Field	Description
	0 SIMPD1 pin is logic low 1 SIMPD1 pin is logic high
1 SDI1	**SIM Detect Interrupt Flag Port 1.** Status flag to indicate the insertion or removal of a SIM card has been detected on port 1. Can create an interrupt to the MCU if SDIM1 is low. Write a "1" to this bit to clear. 0 No insertion or removal of SIM card detected on Port 1(default) 1 Insertion or removal of SIM card detected on Port 1
0 SDIM1	**SIM Detect Interrupt Mask Port 1.** Interrupt mask for the sdi1 interrupt flag. 0 SDI1 enabled 1 SDI1 masked (default)

44.5.4 SIM Transmit Buffer Register (SIM_XMT_BUF)

See the figure below for illustration of valid bits in the SIM Transmit Buffer Register and the table below for description of the bit fields.

Address: 0h base + Ch offset = Ch

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

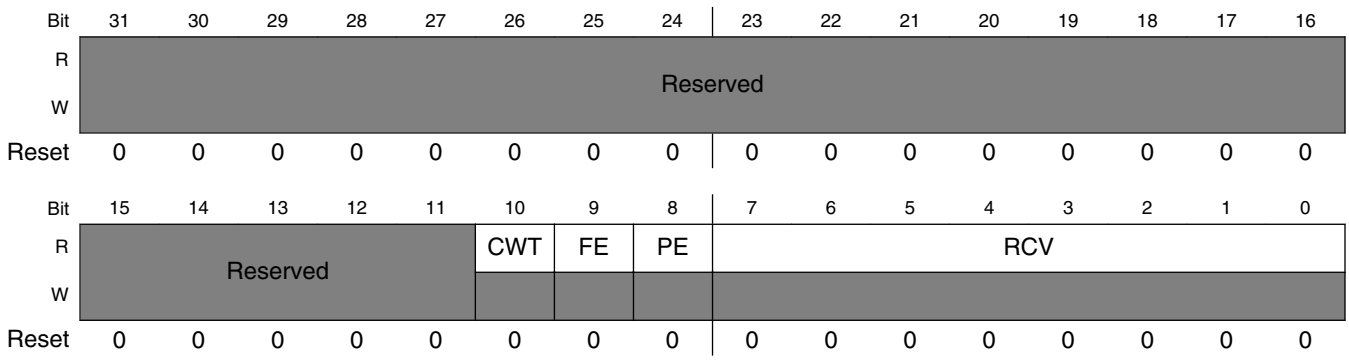
SIM_XMT_BUF field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
XMT	**Transmit Buffer.** Write to the next available location in the transmit buffer.

44.5.5 SIM Receive Buffer Register (SIM_RCV_BUF)

See the figure below for illustration of valid bits in the SIM Receive Buffer Register and the table below for description of the bit fields.

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



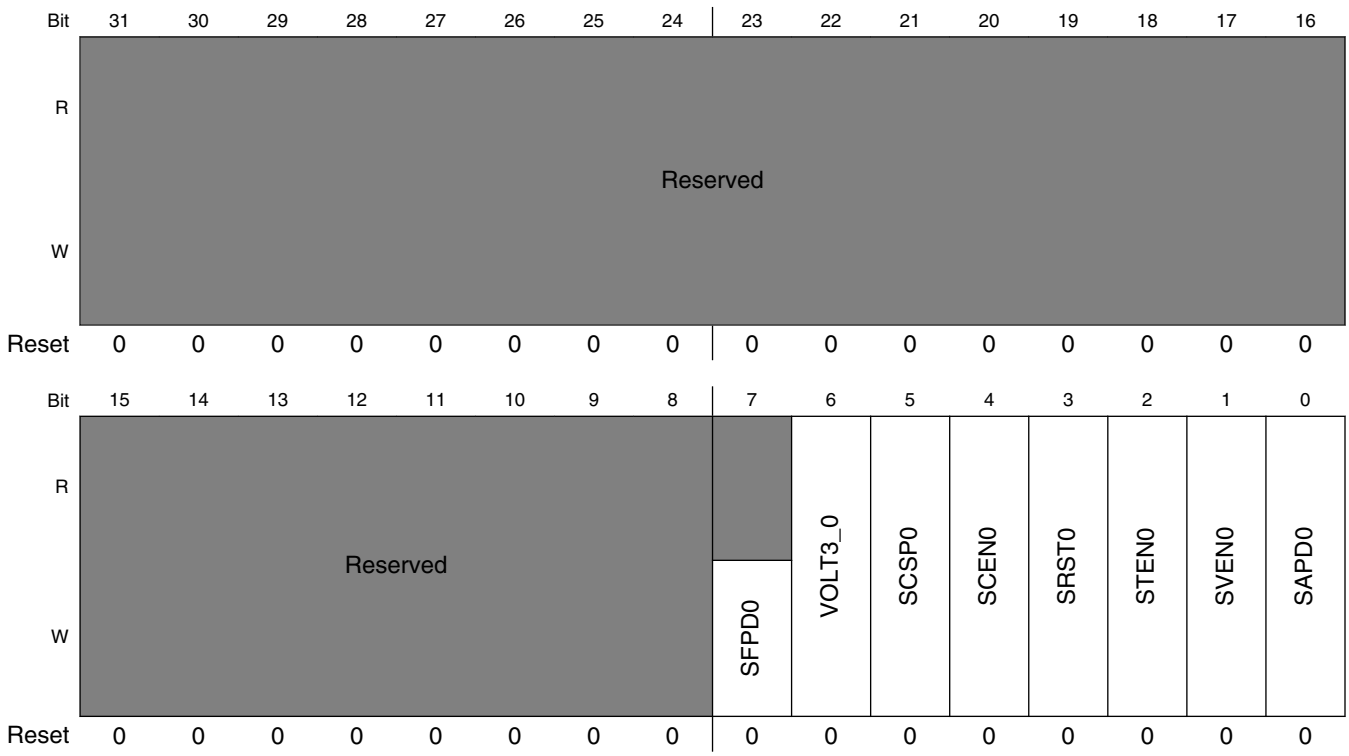
SIM_RCV_BUF field descriptions

Field	Description
31–11 -	This field is reserved. Reserved
10 CWT	**CWT flag.** The CWT indicates that this byte was late. It is not necessary to clear the byte since it is overwritten by the next byte received into that location of the FIFO. 0 Byte was on time 1 Byte was late
9 FE	**Frame Error flag.** The FE flag indicates whether a frame error was detected during the reception of the corresponding byte read in the RCV field. The FE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte contains no framing error (default) 1 Byte contains a framing error
8 PE	**Parity Error flag.** The PE flag indicates whether a parity error was detected during the reception of the corresponding byte read in the RCV field. The PE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. A parity error can create a NACK pulse. 0 Byte contains no parity error (default) 1 Byte contains a parity error
RCV	**Receive buffer.** Read from the next location in the receive buffer.

44.5.6 SIM Port0 Control Register (SIM_PORT0_CNTL)

See the figure below for illustration of valid bits in the SIM Port0 Control Register and the table below for description of the bit fields.

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



SIM_PORT0_CNTL field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 SFPD0	Auto Power Down port0. Writing a "1" to this location will start the auto power down sequence for port 0. This bit will autoclear.A read of this bit will give a "0". 0 No effect 1 Start Auto Power down
6 VOLT3_0	External one wire interface for SIM Card port0. Used to configure the Port 0 transmit pin as bi-directional. Only bi-directional data SIM cards are supported. Hence user should program this bit to 1. This bit should not be changed while the receiver or transmitter is enabled! Port0 uses both RCV and XMT pins 1 Port0 XMT pin bidirectional. Port0 rcv pin unused

Table continues on the next page...

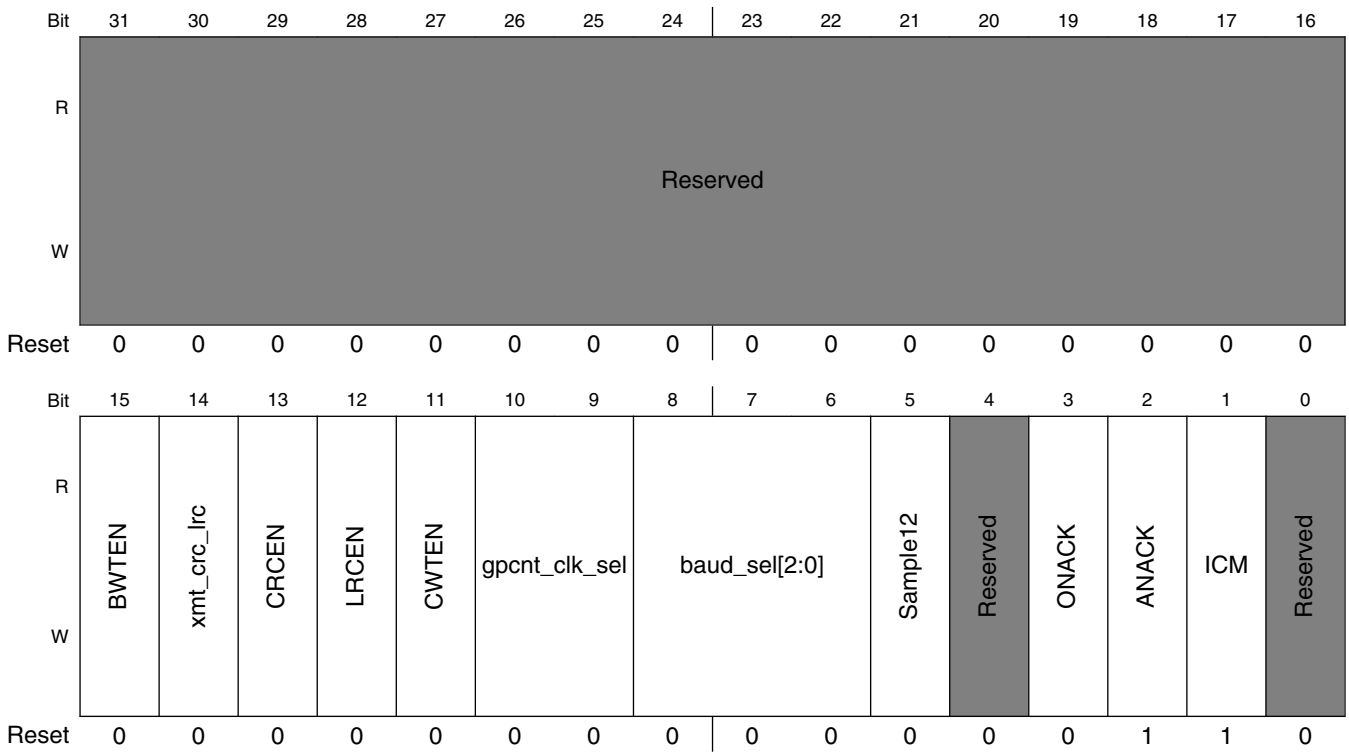
SIM_PORT0_CNTL field descriptions (continued)

Field	Description
5 SCSP0	SIM Card Clock Stop Polarity port0. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN0. It will be forced low by hardware during the auto power down sequence. This forces the clock to be a logic 0 when stopped by auto power down as required by ISO 7816 spec. 0 Clock is logic 0 when stopped by scen0 1 Clock is logic 1 when stopped by scen0
4 SCEN0	SIM card Clock Enable Port 0. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 SIM Card Clock Disabled Port 0 1 SIM Card Clock Enabled Port 0
3 SRST0	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port0 inactive (default) 1 SIM Card reset Port0 active
2 STEN0	SIM card Transmit Enable Port 0. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port0 Transmit Data is forced to zero (default) 1 Port 0 Transmit Data controlled by SIM module
1 SVEN0	SIM card V_{CC} Enable Port 0. Used to control the state of the SVEN0 pin on SIM card port 0. The SVEN0 pin controls the SIM card V_{CC} enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 0 disabled (default) 1 SIM card Voltage Port 0 enabled
0 SAPD0	SIM card Auto Power Down Port 0. Used to enable/disable the auto power down function for port 0. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 0 disabled (default) 1 Auto power down Port 0 enabled

44.5.7 SIM Control Register (SIM_CNTL)

See the figure below for illustration of valid bits in the SIM Control Register and the table below for description of the bit fields.

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



SIM_CNTL field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 BWTEN	**Block wait time enable.** Writing a "1" to this bit will enable the bwt and bgt functions. The bwt and bgt functions can then be individually selected using the interrupt mask. 0 Disable BWT, BGT 1 Enable BWT, BGT
14 xmt_crc_lrc	**Transmit CRC or LRC.** This bit specifies whether or not to transmit the redundancy checking data at the end of a transmission (that is, when the FIFO becomes empty). 0 No Redundancy check info transmitted (default) 1 Transmit LRC or CRC info when FIFO empties (whichever is enabled)
13 CRCEN	**CRC Enable.** This bit enables the calculation of the 16-bit CRC value for both receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and reflected in the

Table continues on the next page...

SIM_CNTL field descriptions (continued)

Field	Description
	crcok bit in the RCV_STAT register. Clearing this bit resets the current CRC residual value in the SIM hardware. 0 16-bit Cyclic Redundancy Checking disabled (default) 1 16-bit Cyclic Redundancy Checking enabled
12 LRCEN	**LRC Enable.** This bit enables the calculation of the 8-bit LRC value for both receiver and transmitter. The result of the calculation is continuously compared to zero and reflected in the lrcok bit in the RCV_STAT register. Clearing this bit resets the current LRC value in the SIM hardware. 0 8-bit Linear Redundancy Checking disabled (default) 1 8-bit Linear Redundancy Checking enabled
11 CWTEN	**Character Wait Time Counter Enable.** Enables the character wait time counter. Clearing this bit resets the counter to zero. 0 Character Wait time Counter off (default) 1 Character Wait time counter on
10–9 gpcnt_clk_sel	**General Purpose Counter Clock Select.** Selects which clock source is used by SIM Module general purpose counter. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are enabled. These input clocks are enabled through other register bits of the SIM module (KILL_CLOCK, RCV_EN, and XMT_EN respectively). 00 Disabled / Reset 01 Card Clock 10 Receive Clock 11 ETU Clock (transmit clock)
8–6 baud_sel[2:0]	SIM **Baud Rate Select.** Selects the asynchronous baud rate divisor of the clock When set to "111", the divisor is set to the value programmed in the DIVISOR register. This allows for more flexible baud rate determination. 000 31 (512/1 Fi/Di) 001 32 (512/2 Fi/Di) 010 16 (512/4 Fi/Di) 011 8 (512/8 Fi/Di) 100 4 (512/16 Fi/Di) 101 2 (512/32 Fi/Di) 110 1 (512/64 Fi/Di) 111 DIVISOR Reg
5 Sample12	**Sample12.** Set the third stage divider. This sets the corresponding sample rate which is the number of times a bit being received is sampled. 0 divide by 8(default) 1 divide by 12
4 -	This field is reserved. Reserved
3 ONACK	**Overrun NACK Enable.** Enables overrun NACK generation. 0 NACK generation on overrun is disabled (default) 1 NACK generation on overrun is enabled

Table continues on the next page...

SIM_CNTL field descriptions (continued)

Field	Description
2 ANACK	**Automatic NACK Enable.** Enables nack generation for parity errors or invalid initial characters when in ICM mode. 0 NACK generation on errors disabled 1 NACK generation on errors enabled (default)
1 ICM	**Initial Character Mode.** Enables initial character mode. Will be automatically cleared by hardware once a valid initial character is received. 0 Initial Character Mode disabled 1 Initial Character Mode enabled (default)
0 -	This field is reserved. Reserved

44.5.8 SIM Clock Prescaler Register (SIM_CLK_PRESCALER)

See the figure below for illustration of valid bits in the SIM Clock Prescaler Register and the table below for description of the bit fields.

Address: 0h base + 1Ch offset = 1Ch

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

SIM_CLK_PRESCALER field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
CLK_ PRESCALER	**Clock prescaler divisor register.** The value written to this register will determine the first stage divider setting. If the ipg_perclk is 66Mhz, a typical setting would be 0x0e. This would set the SIM card clock to 66Mhz/14 = 4.7Mhz. The duty cycle of divided clock will be between 45% and 55% according to ISO7816 requirement. For the odd divider factor (2K+1), the duty cycle will be K/(2K+1) and (K+1)/(2k+1). So for 0x03, the duty cycle is 33%-66%. For 0x05, the duty cycle is 40%-60%. For 0x07, the duty cycle is 43%-57%, and for 0x09, the duty cycle is 44%-56%. For all other values, the clock duty cycle can meet the ISO7816 requirement. **NOTE:** Not all values are listed 0x00 ~0x02 ipg_perclk/ 2 0x03 ipg_perclk / 3 0x04 ipg_perclk / 4 0x05 ipg_perclk / 5 0x06 ipg_perclk / 6 0xFD ipg_perclk / 253

Table continues on the next page...

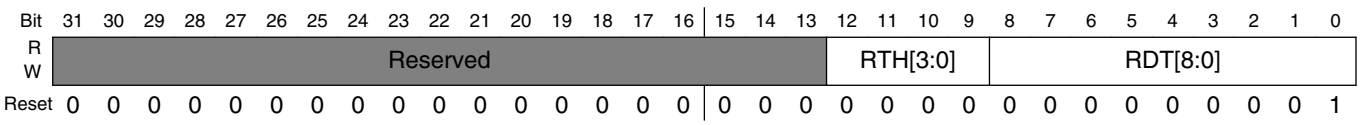
SIM_CLK_PRESCALER field descriptions (continued)

Field	Description
	0xFE ipg_perclk / 254 0xFF ipg_perclk / 255

44.5.9 SIM Receive Threshold Register (SIM_RCV_THRESHOLD)

See the figure below for illustration of valid bits in the SIM Receive Threshold Register and the table below for description of the bit fields.

Address: 0h base + 20h offset = 20h



SIM_RCV_THRESHOLD field descriptions

Field	Description
31–13 -	This field is reserved. Reserved
12–9 RTH[3:0]	**Receive Nack Threshold.** Used to specify the number of consecutive NACK's transmitted by the SIM module, for a given character, before the receive threshold error (RTE) flag is triggered. A value of 0 indicates that RTE is never set. When a valid character is received by the SIM, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. If the ANACK bit is clear in the CNTL register, RTH has no effect.
RDT[8:0]	**Receive Data Threshold.** Determines the number of unread bytes that must exist in the FIFO to trigger the receive data register full (RDRF) interrupt flag. If the number of unread bytes in the receive FIFO is greater than or equal to the value in RDT, the RDRF flag in the RCV_STATUS register will be set. A value of zero indicates that there must be 288 unread bytes in the FIFO to trigger RDRF. If a value of more than 288 is entered in this register, the register value will remain 288.

44.5.10 SIM Enable Register (SIM_ENABLE)

See the figure below for illustration of valid bits in the SIM Enable Register and the table below for description of the bit fields.

Address: 0h base + 24h offset = 24h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXCL	ESTOP_EXE	ESTOP_EN	NACK_DD_EN	TXDMA_EN	RXDMA_EN	XMT_EN	RCV_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_ENABLE field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 RXCL	Reception data latch disable. Used to disable/enable the cross clock domain synchronization operation of reception data in SIM receive state machine. 0 Enable the synchronization operation. 1 Disable the synchronization operation.
6 ESTOP_EXE	Enforce reception early stop execution. Used to execute the function of enforce reception early stop when ESTOP_EN bit set to 1. If this bit is 0 the early stop function can’t be executed when NACK signal occur. 0 Enforce reception early stop function will be executed without NACK. 1 Enforce reception early stop function will be executed at all times.
5 ESTOP_EN	Enforce reception early stop enable. Used to early stop the SIM receive state machine. The state machine will stop earliey than 10.8 ETUs when 11 ETUs mode or earlier than 11.8 ETUs when non 11 ETUs mode, if enable this bit. This function is used for the test case tramsmission stop at 10.8 ETUs from the falling edge of start signal. **NOTE:** This bit should be set to 1 when receive data, and set to 0 when transmit data. 0 Disable the enforce reception early stop function. 1 Enable the enforce reception early stop function.
4 NACK_DD_EN	NACK delay detection enable. Used to enable/disable the function of NACK delay detection. The SIM receive state machine will detect the NACK signal at the 16th phase and use the values of12th, 14th and

Table continues on the next page...

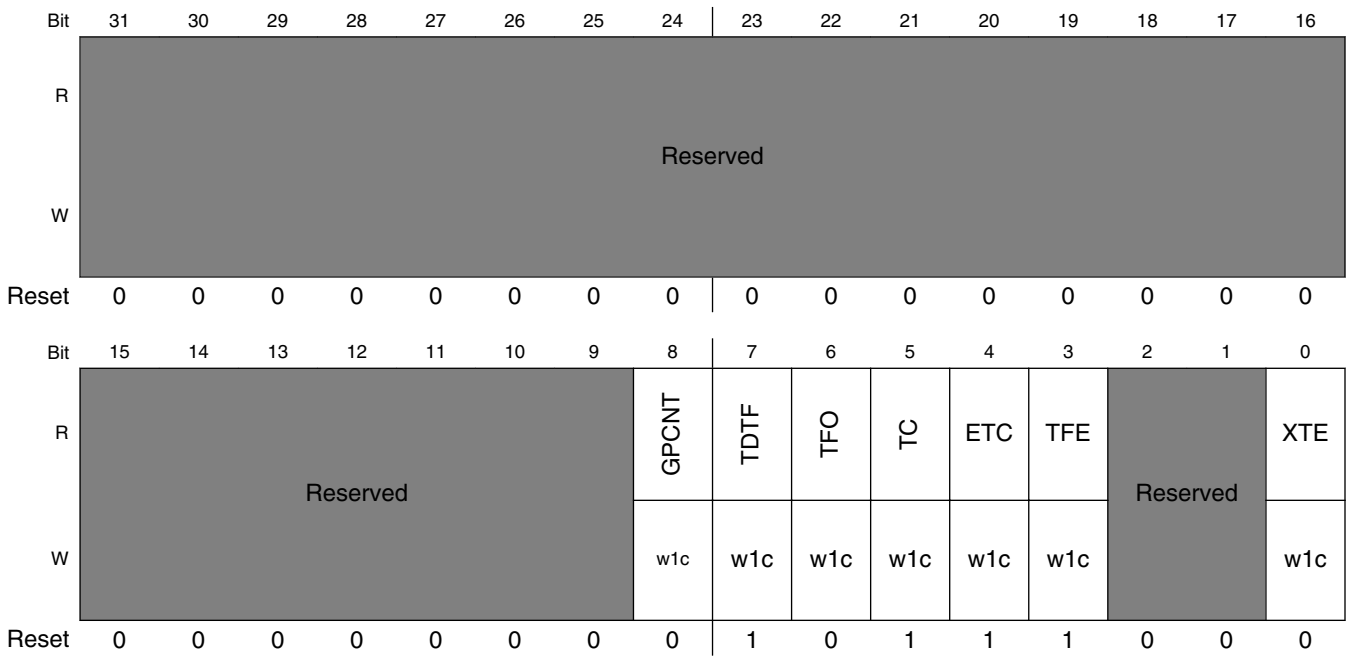
SIM_ENABLE field descriptions (continued)

Field	Description
	15th sample point, if enable this bit. This function is used for the test case NACK signal send at 10.7 ETUs from the falling edge of start signal. 0 Disable the delay detection function. 1 Enable the delay detection function.
3 TXDMA_EN	Transmitter DMA Enable. This bit allows SIM to request for DMA transfers. When enabled, DMA requests are generated when the number of bytes in the transmit FIFO is less than or equal to the value programmed in the tdt[3:0] bits in the XMT_THRESHOLD register and XMT_EN is set. 0 SIM Transmitter DMA requests disabled. 1 SIM Transmitter DMA requests enabled.
2 RXDMA_EN	Receiver DMA Enable. This bit allows SIM to request for DMA transfers. When enabled, DMA requests are generated when when the number of bytes in the receive FIFO is more than or equal to the value programed in the RDT[8:0] in the RCV_THRESHOLD register and RCV_EN is set 0 SIM Receiver DMA request disabled. 1 SIM Receiver DMA request enabled.
1 XMT_EN	SIM **Transmit Enable.** Used to enable/disable the SIM transmit state machine. When the SIM is being used to receive data, XMT_EN should be deasserted. This bit also enables the xmt_clk and rcv_clk inputs to the General Purpose Counter. **NOTE:** Setting this bit (transition from 0 to 1) will reset the CRC and LRC values. 0 SIM Transmitter disabled (default) 1 SIM Transmitter enabled
0 RCV_EN	SIM **Receiver Enable.** Used to enable/disable the SIM receive state machine. The RCV_EN bit should be left high whenever the SIM module is in use. The SIM module has an automatic receive mode operation that disables the reception of characters when the transmitter is operational. Once the transmitter has completed sending the last character, the receiver is automatically enabled. This bit also enables the RCV_CLK input to the General Purpose Counter. 0 SIM Receiver disabled (default) 1 SIM Receiver enabled

44.5.11 SIM Transmit Status Register (SIM_XMT_STATUS)

See the figure below for illustration of valid bits in the SIM Transmit Status Register and the table below for description of the bit fields.

Address: 0h base + 28h offset = 28h



SIM_XMT_STATUS field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 GPCNT	**General purpose Counter Flag.** Used to indicate when the General purpose counter has reached the value in the GPCNT register. 0 GPCNT time not reached, or bit has been cleared. (default). 1 General Purpose counter has reached the GPCNT value.
7 TDTF	**Transmit Data Threshold Flag.** Used to indicate when the number of bytes in the transmit FIFO is less than or equal to the value programmed in the tdt[3:0] bits in the XMT_THRESHOLD register. 0 Number of bytes in FIFO is greater than tdt[3:0], or bit has been cleared. 1 Number of bytes in FIFO is less than or equal to tdt[3:0] (default)
6 TFO	**Transmit FIFO Overfill Error.** Used to indicate when the Transmit FIFO has been written with more than 16 bytes. The tfo bit can only be cleared by setting the flush_xmt or soft_reset bit in the RESET_CNTL register. 0 No transmit FIFO overfill error has occurred (default). 1 A Transmit FIFO overfill error has occurred.

Table continues on the next page...

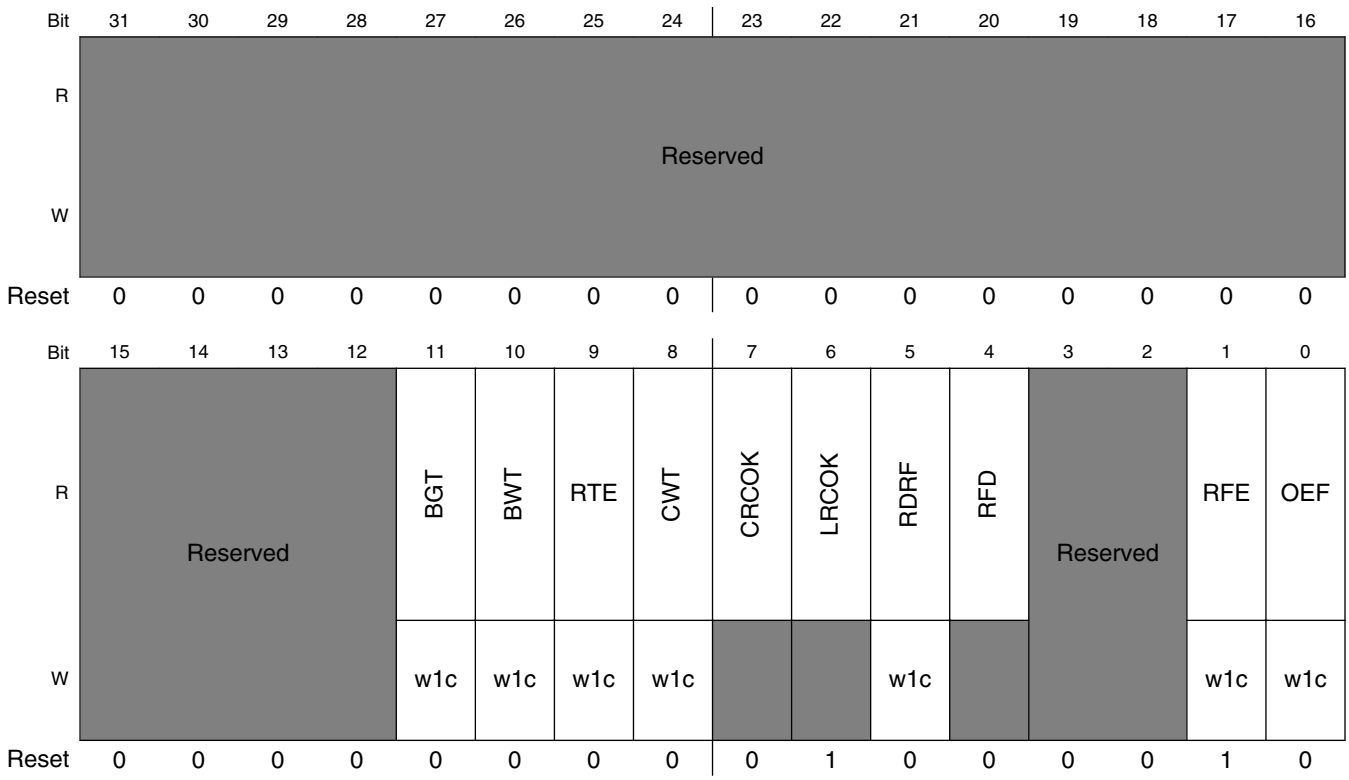
SIM_XMT_STATUS field descriptions (continued)

Field	Description
5 TC	**Transmit Complete.** Used to indicate whether the SIM transmitter is ready for a new transmission. The tc flag becomes set after the guard time has expired for the last byte in the transmit FIFO. The tc flag will create an interrupt if tcim in the INT_MASK register is low. The tc bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
4 ETC	**Early Transmit Complete.** Used to indicate that the SIM transmitter has finished sending the current byte and the transmit FIFO is empty. This bit differs from the tc bit in that it is set before the guard time of the last byte has elapsed. The ETC flag will create an interrupt if ETCIM in the INT_MASK register is low. The ETC bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
3 TFE	**Transmit FIFO Empty.** Used to indicate that the SIM transmit FIFO has emptied. This bit will be set when the last byte in the transmit FIFO has been transferred to the SIM transmitter shift register. The TFE flag will create an interrupt if TFEIM in the INT_MASK register is low. The TFE bit is a write-one-to-clear bit. 0 Transmit FIFO is not empty 1 Transmit FIFO is empty (default)
2–1 -	This field is reserved. Reserved
0 XTE	**Transmit NACK Threshold Error.** Used to indicate the transmit NACK threshold has been reached. When XTE is high, no further transmissions will be done until the XTE flag is cleared. Any data transmissions still pending in the transmit FIFO will be aborted, and the TC, ETC, and TFE flags will be set. The XTE flag will create an interrupt if XTM in the IMT_MASK register is low. The XTE bit is a write-one-to-clear bit. 0 Transmit NACK threshold has not been reached (default) 1 Transmit NACK threshold reached; transmitter frozen

44.5.12 SIM Receive Status Register (SIM_RCV_STATUS)

See the figure below for illustration of valid bits in the SIM Receive Status Register and the table below for description of the bit fields.

Address: 0h base + 2Ch offset = 2Ch



SIM_RCV_STATUS field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11 BGT	**Block guard time error flag.** Used to indicate if the block guard time was too small. The threshold is set by the block guard time register. 0 Block guard time was sufficient. 1 Block guard time was too small.
10 BWT	**Block wait time error flag.** Used to indicate if the block wait time has been exceeded. The threshold is set by the block wait time registers. 0 Block wait time not exceeded. 1 Block wait time was exceeded.
9 RTE	**Receive NACK threshold error flag.** Used to indicate whether the number of consecutive NACKs generated by the SIM module in response to receive parity errors, for the byte being received, equals the

Table continues on the next page...

SIM_RCV_STATUS field descriptions (continued)

Field	Description
	value programmed in the RTH[3:0] in the RCV_THRESHOLD register. This bit would never be set unless the ANACK bit is set in the CNTL register. The SAPDx bit in the PORTx_CNTL resister must be set to enable the threshold error to trigger the auto power down sequence. RTE is a write-one-to-clear bit. Clearing this bit also resets the internal counter for consecutive NACKs being transmitted for a given byte. 0 Number of NACKs generated by the receiver is less than the value programmed in RTH[3:0] 1 Number of NACKs generated by the receiver is equal to the value programmed in RTH[3:0]
8 CWT	**Character Wait Time Counter Flag.** Used to indicate when the time between received characters is equal to or greater than the value programmed in the CHAR_WAIT register. 0 No CWT violation has occurred (default). 1 Time between two consecutive characters exceeded the value in CHAR_WAIT.
7 CRCOK	**Cyclic Redundancy Check Okay flag.** Used to indicate when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters from the point the CRCEN bit is set in the CNTL register. The current CRC residual can be reset by three mechanisms: Clear CRCEN bit in CNTL register Set XMT_EN bit in ENABLE register Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current CRC value does not match remainder. 1 Current calculated CRC value matches the expected result.
6 LRCOK	Linear Redundancy Check Okay flag. Used to indicate when the calculated 8-bit LRC value is zero value for the current input data stream. The value is calculated across all received characters from the point the LRCEN bit is set in the CNTL register. The current LRC residual can be reset by three mechanisms: Clear LRCEN bit in CNTL register Set XMT_EN bit in ENABLE register Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current LRC value does not match remainder. 1 Current calculated LRC value matches the expected result (that is, zero).
5 RDRF	Receive Data Register Full. Used to indicate when the number of bytes in the receive FIFO is more than or equal to the value programed in the RDT[8:0] in the RCV_THRESHOLD register. 0 Number of unread bytes in receive buffer < value set by RDT[8:0] (default), or bit has been cleared. 1 Number of unread bytes in receive buffer >= value set by RDT[8:0].
4 RFD	**Receive FIFO has unread Data.** Used to indicate that there is at least one unread byte in the receive data FIFO. Can only be cleared by reading all bytes out of the receive FIFO. The RFD bit cannot be used to create an interrupt. Normally, the SIM triggers the interrupt with RDRF and software uses rfd to read all of the bytes out of the receive FIFO. 0 There are no unread bytes in the receive FIFO (default). 1 There is at least one unread byte in the receiver FIFO.
3–2 -	This field is reserved. Reserved
1 RFE	**Receive FIFO Empty.** Used to indicate that the SIM receive FIFO has emptied. This bit will be set when the last byte in the receive FIFO has been transferred to the IPS bus. The RFE flag will create an interrupt if RFEIM in the INT_MASK register is low. The RFE bit is a write-one-to-clear bit.

Table continues on the next page...

SIM_RCV_STATUS field descriptions (continued)

Field	Description
	0 Receive FIFO is not empty 1 Receive FIFO is empty (default)
0 OEF	**Overrun Error Flag.** Used to indicate that the SIM was unable to store received data due to already having 288 unread bytes in the FIFO. It does not necessarily indicate that data has been lost. If the ONACK control bit in the CNTL register is set, there will be a NACK pulse generated on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the SIM card which implies that no data has actually been lost. In this case, the OEF flag is just an indicator that this situation has occurred which may be helpful in system debug. For the case where ONACK is not set, a set OEF flag does indicate a loss of data since all bytes received with the OEF flag set will indeed be lost (including the byte that caused the bit to be set). The OEF flag will cause an interrupt if the oim bit in the INT_MASK register. The OEF flag is a write-one-to-clear bit. 0 No overrun error has occurred (default). 1 A byte was received when the received FIFO was already full.

44.5.13 SIM Interrupt Mask Register (SIM_INT_MASK)

See the figure below for illustration of valid bits in the SIM Interrupt Mask Register and the table below for description of the bit fields.

Address: 0h base + 30h offset = 30h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		RFEM	BGTM	BWTM	RTM	CWTM	GPCNTM	TDTFM	TFOM	XTM	TFEIM	ETCIM	OIM	TCIM	RIM
Reset	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SIM_INT_MASK field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13 RFEM	**Receive fifo empty interrupt mask.** Used to enable/disable the ability of the RFE flag in the RCV_STATUS register to generate SIM interrupts.

Table continues on the next page...

SIM_INT_MASK field descriptions (continued)

Field	Description
	0 RFE interrupt enabled 1 RFE interrupt masked (default)
12 BGTM	**Block guard time interrupt mask.** Used to enable/disable the ability of the bgt flag in the RCV_STATUS register to generate SIM interrupts. 0 BGT interrupt enabled 1 BGT interrupt masked (default)
11 BWTM	**Block wait time interrupt mask.** Used to enable/disable the ability of the bwt flag in the RCV_STATUS register to generate SIM interrupts. 0 BWT interrupt enabled 1 BWT interrupt masked (default)
10 RTM	**Receive Nack threshold interrupt mask.** Used to enable/disable the ability of the RTE flag in the RCV_STATUS register to generate SIM interrupts. 0 RTE interrupt enabled 1 RTE interrupt masked (default)
9 CWTM	**Character Wait Time Interrupt Mask.** Used to enable/disable the ability of the CWT flag in the RCV_STATUS register to generate SIM interrupts. 0 CWT interrupt enabled 1 CWT interrupt masked (default)
8 GPCNTM	**General Purpose Counter Interrupt Mask.** Used to enable/disable the ability of the GPCNT flag in the XMT_STATUS register to generate SIM interrupts. 0 GPCNT interrupt enabled 1 GPCNT interrupt masked (default)
7 TDTFM	**Transmit Data Threshold Interrupt Mask.** Used to enable/disable the ability of the TDTF flag in the XMT_STATUS register to generate SIM interrupts. 0 TDTF interrupt enabled 1 TDTF interrupt masked (default)
6 TFOM	**Transmit FIFO Overfill Error Interrupt Mask.** Used to enable/disable the ability of the TFO flag in the XMT_STATUS register to generate SIM interrupts. 0 TFO interrupt enabled 1 TFO interrupt masked (default)
5 XTM	**Transmit Threshold Interrupt Mask.** Used to enable/disable the ability of the XTE flag in the XMT_STATUS register to generate SIM interrupts 0 XTE interrupt enabled 1 XTE interrupt masked (default)
4 TFEIM	**Transmit FIFO Empty Interrupt Mask.** Used to enable/disable the ability of the TFE flag in the XMT_STATUS register to generate SIM interrupts. 0 TFE interrupt enabled 1 TFE interrupt masked (default)
3 ETCIM	**Early Transmit Complete Interrupt Mask.** Used to enable/disable the ability of the ETC flag in the XMT_STATUS register to generate SIM interrupts.

Table continues on the next page...

SIM_INT_MASK field descriptions (continued)

Field	Description
	0 ETC interrupt enabled 1 ETC interrupt masked (default)
2 OIM	**Overrun Interrupt Mask.** Used to enable/disable the ability of the OEF flag in the RCV_STATUS register to generate SIM interrupts. 0 OEF interrupt enabled 1 OEF interrupt masked (default)
1 TCIM	**Transmit Complete Interrupt Mask.** Used to enable/disable the ability of the TC flag in the XMT_STATUS register to generate SIM interrupts. 0 TC interrupt enabled 1 TC interrupt masked (default)
0 RIM	**Receive Interrupt Mask.** Used to enable/disable the ability of the RDRF flag in the RCV_STATUS register to generate SIM interrupts. 0 RDRF interrupt enabled 1 RDRF interrupt masked (default)

44.5.14 SIM Port0 Detect Register (SIM_PORT0_DETECT)

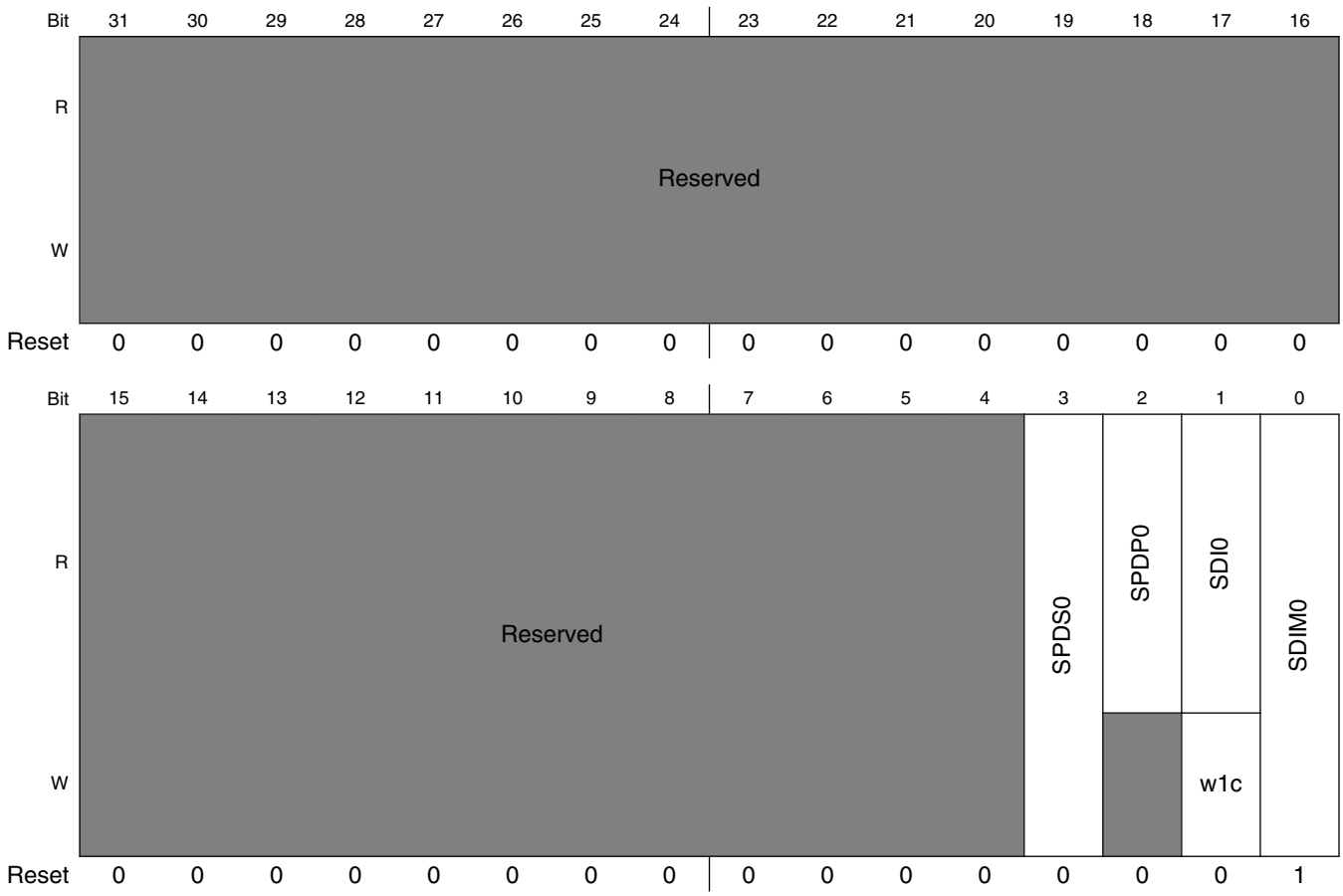
See the figure below for illustration of valid bits in the SIM Port0 Detect Register and the table below for description of the bit fields.

SPDS0 determines which edge transition of the SIMPD0 pin is used for SIM card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the SIMPD0 edge, specified by SPDS0, will cause the following:

- SDI0 to be set
- If the SDIM0 mask is clear, an interrupt to occur on SIMIRQ_N
- If SAPD0 in the PORT0_CNTL register is set, auto power down sequence will begin

If a SIM card insertion is expected, SAPD0 can be set low to avoid the auto power down sequence. There is no auto power up sequence. SPDP0 can be used to determine the current state of the SIMPD0 pin.

Address: 0h base + 3Ch offset = 3Ch



SIM_PORT0_DETECT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3 SPDS0	**SIM Presence Detect Select Port 0.** Controls which edge of the SIMPD0 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD0 Input (default) 1 Rising edge of SIMPD0 Input
2 SPDP0	**SIMPD0 input pin status.** This bit reflects the state of the SIMPD0 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD0 pin itself. 0 SIMPD0 pin is logic low 1 SIMPD0 pin is logic high
1 SDI0	**SIM Detect Interrupt flag Port 0.** Status flag to indicate the insertion or removal of a SIM card has been detected on port 0. Can create an interrupt to the MCU if SDIM0 is low. Write a "1" to this bit to clear. 0 No insertion or removal of SIM card detected on Port 0 (default) 1 Insertion or removal of SIM card detected on Port 0
0 SDIM0	**SIM Detect Interrupt Mask Port 0.** Interrupt mask for the SDI0 interrupt flag. 0 sdi0 enabled 1 sdi0 masked (default)

44.5.15 SIM Data Format Register (SIM_DATA_FORMAT)

See the figure below for illustration of valid bits in the SIM Data Format Register and the table below for description of the bit fields.

Address: 0h base + 40h offset = 40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

SIM_DATA_FORMAT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 IC	**Inverse Convention.** Used to configure the SIM to use either inverse convention or direct convention for its data format. The IC bit can be controlled by software, but it is normally set by hardware as a result of the interpretation of the initial character when in ICM mode. 0 Direction convention transfers enabled (default). 1 Inverse convention transfers enabled.

44.5.16 SIM Transmit Threshold Register (SIM_XMT_THRESHOLD)

See the figure below for illustration of valid bits in the SIM Transmit Threshold Register and the table below for description of the bit fields.

Address: 0h base + 44h offset = 44h

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

SIM_XMT_THRESHOLD field descriptions

Field	Description
31–8 -	This field is reserved. Reserved

Table continues on the next page...

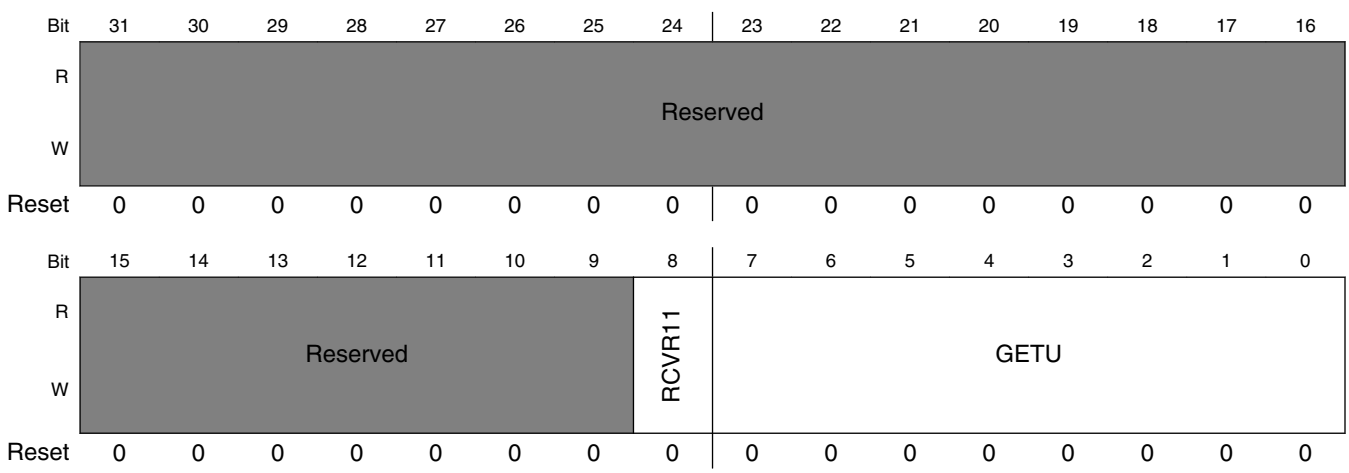
SIM_XMT_THRESHOLD field descriptions (continued)

Field	Description
7–4 XTH	XTH[3:0] **Transmit NACK Threshold.** Used to set the NACK threshold for the transmitter. Once the threshold number set by XTH has been reached for the current byte being transmitted, the error flag XTE in the XMT_STATUS register will be set. Setting of XTE causes the remaining transmissions queued in the transmit FIFO to be aborted and no more transmissions to occur until software clears XTE. To trigger XTE, a given byte being transmitted must reach the XTH threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next. 0x0 XTE will never be set; retransmission after NACK reception is disabled. 0x1 XTE will be set after 1 nack is received; 0 retransmissions occurs. 0x2 XTE will be set after 2 nacks are received; at most 1 retransmission occurs. 0x3 XTE will be set after 3 nacks are received; at most 2 retransmissions occurs. 0xX XTE will be set after X nacks are received; at most (X - 1) retransmissions occurs. 0xF XTE will be set after 15 nacks are received; at most 14 retransmissions occurs.
TDT	**Transmit Data Threshold.** Used to set the threshold value for the Transmit FIFO at which the TDTF bit in the XMT_STATUS register will be set. When the number of bytes in the Transmit FIFO is less than or equal to TDT[3:0], TDTF will be set.

44.5.17 SIM Transmit Guard Control Register (SIM_GUARD_CNTL)

See the figure below for illustration of valid bits in the SIM Transmit Guard Control Register and the table below for description of the bit fields.

Address: 0h base + 48h offset = 48h



SIM_GUARD_CNTL field descriptions

Field	Description
31–9 -	This field is reserved. Reserved

Table continues on the next page...

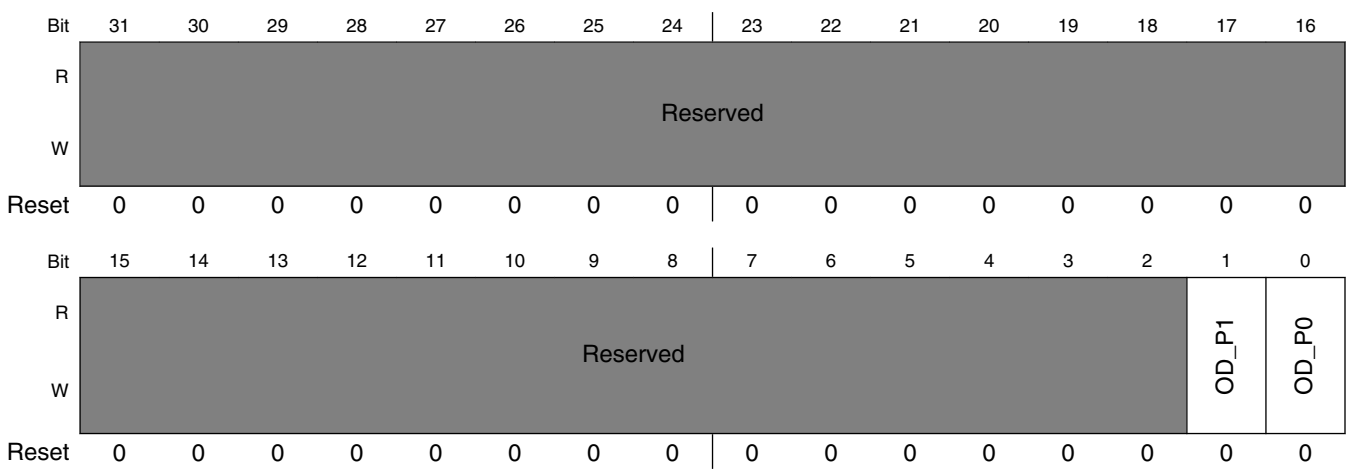
SIM_GUARD_CNTL field descriptions (continued)

Field	Description
8 RCVR11	**Receiver use 11 ETUs.** Used to configure the SIM module receiver for 11 ETU operation (that is, 1 Stop bit). This bit is provided for support of T=1 cards. 0 Receiver configured for 12 ETU operation (default) 1 Receiver configured for 11 ETU operation
GETU	**Transmit Guard ETUs.** Used to control the number of additional Elementary Time Units (ETUs) inserted between bytes transmitted by the SIM transmitter. An ETU is equivalent to one bit time at the given baud rate (for example, the length of a START bit). The guard time has no effect on the SIM receiver. A value of 0x00 inserts no additional ETUs, while a value of 0xFE inserts 254 additional ETUs. A value of 0xFF subtracts one ETU by reducing the number of STOP bits from two to one.

44.5.18 SIM Open Drain Configuration Control Register (SIM_OD_CONFIG)

See the figure below for illustration of valid bits in the SIM Open Drain Configuration Control Register and the table below for description of the bit fields.

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



SIM_OD_CONFIG field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
1 OD_P1	**Open Drain control for Port 1.** Used to control whether the XMT data line on port 1 is open-drain. If AMODE bit in SETUP register is set, this bit will have no effect. 0 XMT pin on port 1 is push-pull (default). 1 XMT pin on port 1 is open-drain.

Table continues on the next page...

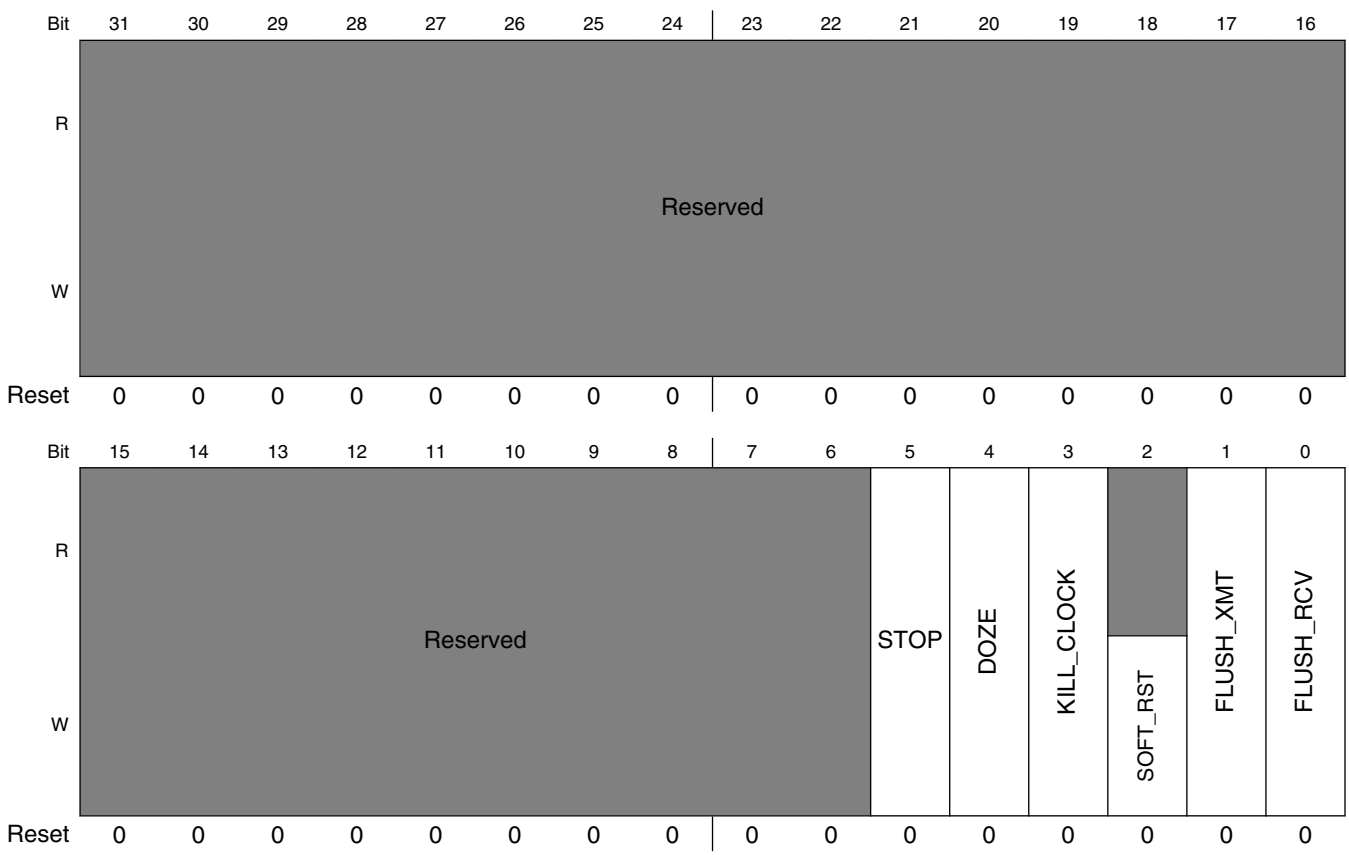
SIM_OD_CONFIG field descriptions (continued)

Field	Description
0 OD_P0	**Open Drain control for Port 0.** Used to control whether the XMT data line on port 0 is open-drain. 0 XMT pin on port 0 is push-pull (default). 1 XMT pin on port 0 is open-drain.

44.5.19 SIM Reset Control Register (SIM_RESET_CNTL)

See the figure below for illustration of valid bits in the SIM Reset Control Register and the table below for description of the bit fields.

Address: 0h base + 50h offset = 50h



SIM_RESET_CNTL field descriptions

Field	Description
31–6 -	This field is reserved. Reserved

Table continues on the next page...

SIM_RESET_CNTL field descriptions (continued)

Field	Description
5 STOP	**STOP.** Used to configure the operation of the SIM module when a processor STOP instruction is executed. This bit is added to provide support for SIM cards that do not allow the SIM Card clock to be stopped while power is applied. 0 STOP instruction shuts down all SIM clocks (default). 1 STOP instruction shuts down all clocks except for the BAUD_CLK (clock provided to SIM Card).
4 DOZE	DOZE. Used to configure the operation of the SIM module when a processor DOZE instruction is executed. 0 DOZE instruction has no effect on SIM module (default). 1 DOZE instruction will cause SIM module to gate SIM clocks when the transmit FIFO is empty.
3 KILL_CLOCK	**Kill** SIM **Clock.** Used to enable/disable the SIM clock input to the SIM module. This bit will gate all SIM clocks including the SIM card clock regardless of the state of the STOP bit described above. 0 SIM input clock enabled (default). 1 SIM input clock disabled.
2 SOFT_RST	**Software Reset.** Used to reset the entire SIM module. This acts the same as a hardware reset for the SIM module. This bit is self-clearing. **NOTE:** Software should allow a minimum of 4 reference clock cycles (CKIH) before attempting to access the SIM module after a software reset. 0 SIM Normal operation (default). 1 SIM held in Reset.
1 FLUSH_XMT	**Flush Transmitter.** This bit operates as a SIM transmitter reset. The receive portion of the SIM module is not affected. The software must clear this bit before the SIM transmitter can operate. 0 SIM Transmitter normal operation (default). 1 SIM Transmitter held in Reset.
0 FLUSH_RCV	**Flush Receiver.** This bit operates as a SIM receiver reset. The transmit portion of the SIM module is not affected. The software must clear this bit before the SIM receiver can operate. 0 SIM Receiver normal operation (default). 1 SIM Receiver held in Reset.

44.5.20 SIM Character Wait Time Register (SIM_CHAR_WAIT)

See the figure below for illustration of valid bits in the SIM Character Wait Time Register and the table below for description of the bit fields.

Address: 0h base + 54h offset = 54h

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

SIM_CHAR_WAIT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
CWT	**Character Wait Time.** The value written to this register will specify the number of ETU times allowed between characters. Default is 0xFFFF

44.5.21 SIM General Purpose Counter Register (SIM_GPCNT)

See the figure below for illustration of valid bits in the SIM General Purpose Counter Register and the table below for description of the bit fields.

Address: 0h base + 58h offset = 58h

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

SIM_GPCNT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
GPCNT	**General Purpose Counter.** The value written to this register will be used to compare to the general purpose counter in the SIM module. Once the General purpose counter reaches this value, the GPCNT flag in the XMT_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration. Default is 0xFFFF

44.5.22 SIM Divisor Register (SIM_DIVISOR)

See the figure below for illustration of valid bits in the SIM Divisor Register and the table below for description of the bit fields.

Address: 0h base + 5Ch offset = 5Ch

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

SIM_DIVISOR field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
DIVISOR	**DIVISOR Register.** The value written to this register will be used to generate the SIM Receive clock. The BAUD_SEL[2:0] bits in the CNTL register must be set to '111' in order to control the divisor value using the DIVISOR register. Default is 0xFF

44.5.23 SIM Block Wait Time Register (SIM_BWT)

See the figure below for illustration of valid bits in the SIM Block Wait Time Register and the table below for description of the bit fields.

Address: 0h base + 60h offset = 60h

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

SIM_BWT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
BWT	**BWT Register 16 LSB.** The value in this register is the block wait time 16 LSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag will be set. Default is 0xFFFF

44.5.24 SIM Block Guard Time Register (SIM_BGT)

See the figure below for illustration of valid bits in the SIM Block Guard Time Register and the table below for description of the bit fields.

Address: 0h base + 64h offset = 64h

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

SIM_BGT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
BGT	**BGT Register.** The value in this register is the block guard time. Time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be greater than this value. If it is not, then the BGT flag will be set. Default is 0x0000

44.5.25 SIM Block Wait Time Register HIGH (SIM_BWT_H)

See the figure below for illustration of valid bits in the SIM Block Wait Time Register HIGH and the table below for description of the bit fields.

Address: 0h base + 68h offset = 68h

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

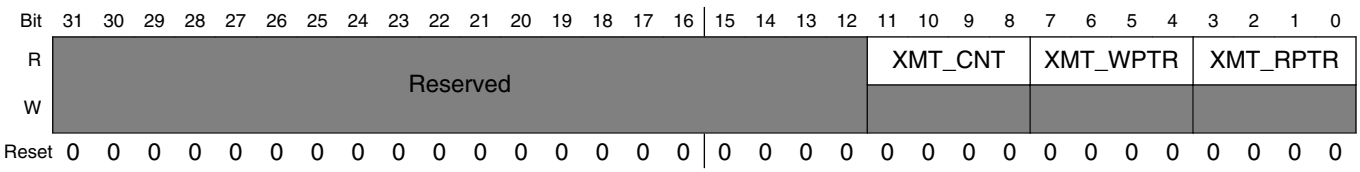
SIM_BWT_H field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
BWT_H	**BWT Register 16 MSB.** The value in this register is the block wait time 16 MSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag is set. Default is 0xFFFF

44.5.26 SIM Transmit FIFO Status Register (SIM_XMT_FIFO_STAT)

See the figure below for illustration of valid bits in the SIM Transmit FIFO Status Register and the table below for description of the bit fields.

Address: 0h base + 6Ch offset = 6Ch



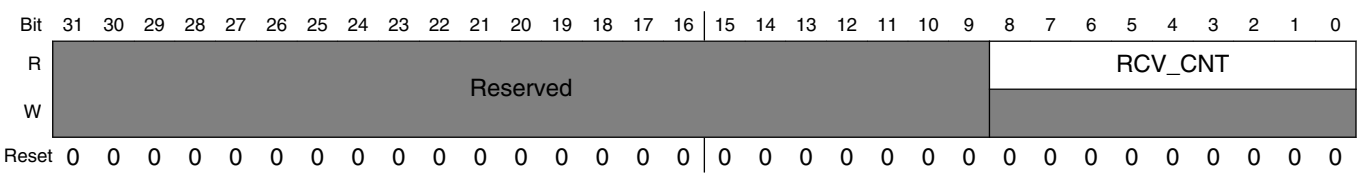
SIM_XMT_FIFO_STAT field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 XMT_CNT	These bits indicate the number of Bytes in the transmit FIFO. '0' value means FIFO is empty or full.
7–4 XMT_WPTR	These bits indicate the transmit FIFO Write Pointer.
XMT_RPTR	These bits indicate the transmit FIFO Read Pointer.

44.5.27 SIM Receive FIFO Counter Register (SIM_RCV_FIFO_CNT)

See the figure below for illustration of valid bits in the SIM Receiver FIFO Counter Register and the table below for description of the bit fields.

Address: 0h base + 70h offset = 70h



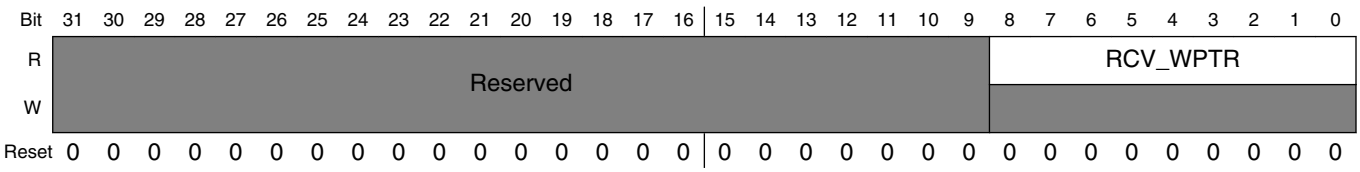
SIM_RCV_FIFO_CNT field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
RCV_CNT	These bits indicate the number of Byte can be written into the receive FIFO. 0 value means FIFO is empty or full.

44.5.28 SIM Receive FIFO Write Pointer Register (SIM_RCV_FIFO_WPTR)

See the figure below for illustration of valid bits in the SIM Receive FIFO Write Pointer Register and the table below for description of the bit fields.

Address: 0h base + 74h offset = 74h



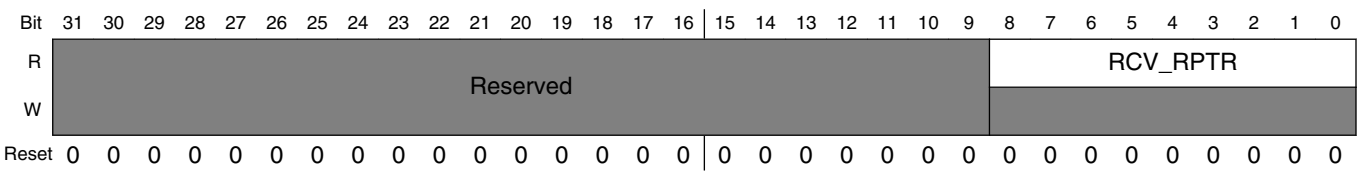
SIM_RCV_FIFO_WPTR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
RCV_WPTR	These bits indicate the receive FIFO Write pointer.

44.5.29 SIM Receive FIFO Read Pointer Register (SIM_RCV_FIFO_RPTR)

See the figure below for illustration of valid bits in the SIM Receive FIFO Read Pointer Register and the table below for description of the bit fields.

Address: 0h base + 78h offset = 78h



SIM_RCV_FIFO_RPTR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
RCV_RPTR	These bits indicate the receiver FIFO read pointer.

Chapter 45
System JTAG Controller (SJC)

45.1 Overview

The System JTAG Controller (SJC) provides debug and test control with the maximum security.

The test access port (TAP) is designed to support features compatible with the IEEE Standard 1149.1 v2001 (JTAG).

The figure below shows an overview of the JTAG architecture.

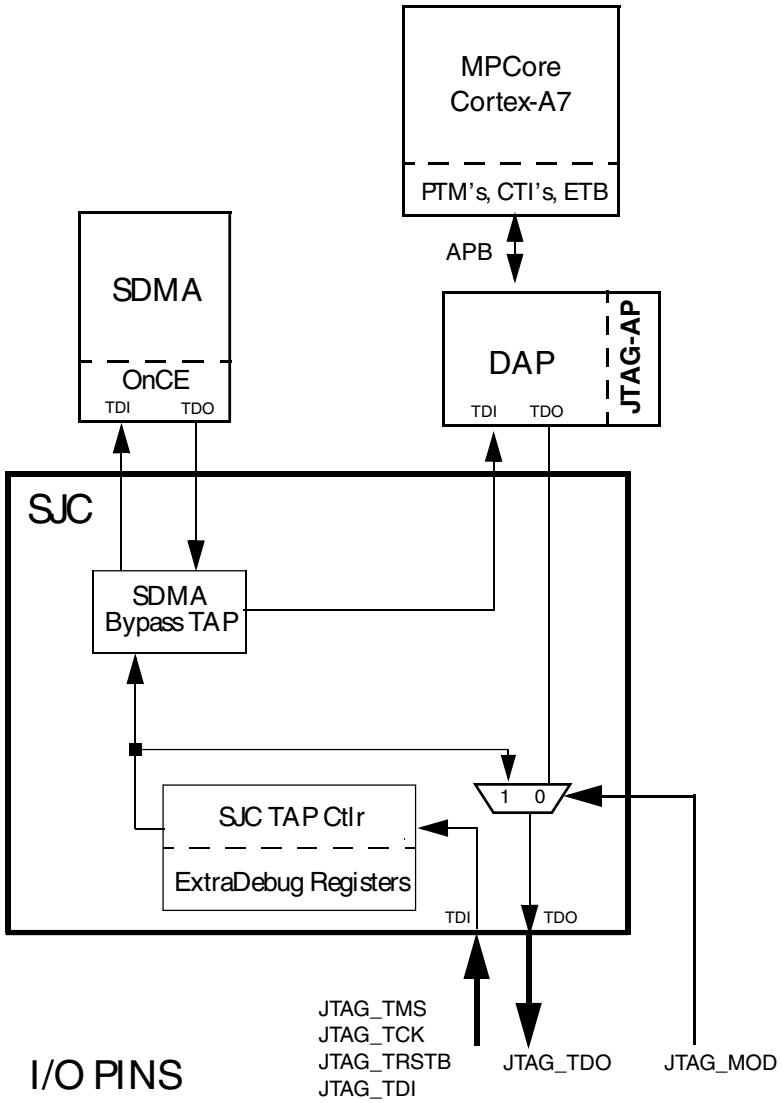


Figure 45-1. System JTAG Controller (SJC) Block Diagram

45.1.1 Features

The System JTAG Controller (SJC) provides the following capabilities:

- JTAG IEEE1149.1 mandatory instructions, see EXTEST Instruction, SAMPLE/ PRELOAD Instruction , and BYPASS Instruction .
- JTAG IEEE1149.1 optional instructions, see ID_CODE Instruction (IDCODE) , and HIGHZ Instruction.
- JTAG IEEE P1149.1 (standard JTAG) interface to off-chip test and development equipment including an SJC-only mode for true IEEE 1149.1 compliance, used primarily for board-level implementation of boundary scan.

- IEEE P1149.6 (JTAG) mandatory instructions, see EXTEST_PULSE instruction and EXTEST_TRAIN instruction. These two instructions enable edge-detecting behavior on the signal path containing AC pins.
- Debug-related control and status, such as putting selected cores into reset and/or debug mode and the ability to monitor individual core status signals via JTAG.
- Provides means for accessing each OnCE/ICE TAP controller independently to control a target system (see Modes of Operation).
- ExtraDebug logic (see ENABLE_ExtraDebug Instruction).
- The maximum clock speed of the SJC is one-eight of the lowest frequency of the accessed OnCE/ICE. For example in normal operation (no core in low-power mode), this frequency is one-eight of the SDMA frequency if this core is present in the TDI-TDO chain (serially connected with other cores or standalone). The user must also consider the 25 MHz frequency limitation on the CE bus.
- Core compliant modes to support standalone core debuggers (see Modes of Operation).
- Multi-cores daisy chained mode (default one) to support multi-core debuggers (see Modes of Operation).

Detailed information about the SJC is provided in the Security Reference Manual. Contact your Freescale representative for information about obtaining this document.

45.1.2 Modes of Operation

The SJC modes are controlled through both the TAP select register (SJC_TSR) and the MOD input port.

The MOD port (typically connected to pad of the same name) selects between two possible topologies of TAP connections, as seen at SoC level:

- Negating it (this should be the default state) selects all the TAPs (SJC, SDMA, DAP and ARM/ETM) to be connected in the TDI-TDO chain, which is referred to as "daisy chain" mode, throughout this chapter.
- Asserting it only selects the SJC TAP to be connected in the TDI-TDO chain.

IEEE1149.1 standard features are enabled by configuring the SJC input pin: MOD. Refer to the following table for MOD settings details:

Table 45-1. SJC Modes

MOD	Name	Description
0	Daisy chain ALL	For common SW debug (High speed and production)
1	SJC only	IEEE 1149.1 JTAG compliant mode

The following figure shows the SJC mode selection flow. The numbers shown in parenthesis below each block name indicates the TAP's IR length.

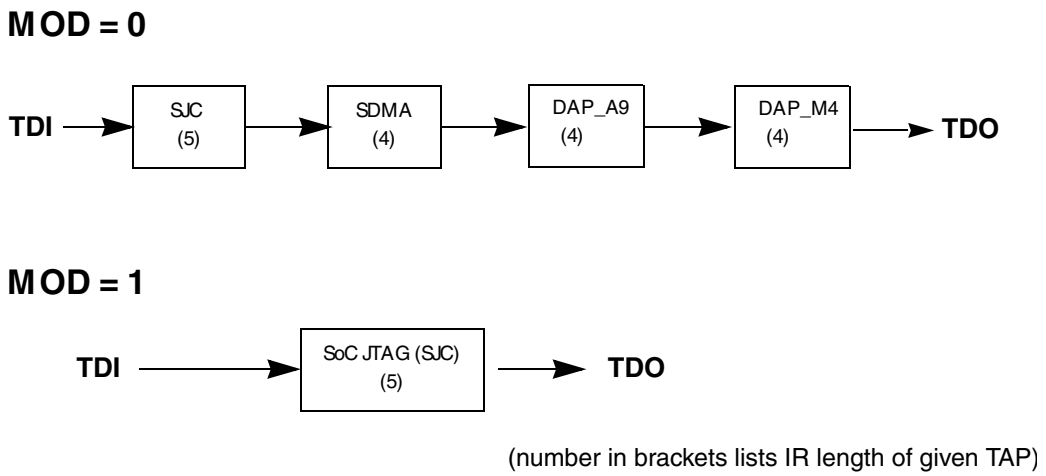


Figure 45-2. SJC Mode Selection Using MOD Pin Sampling

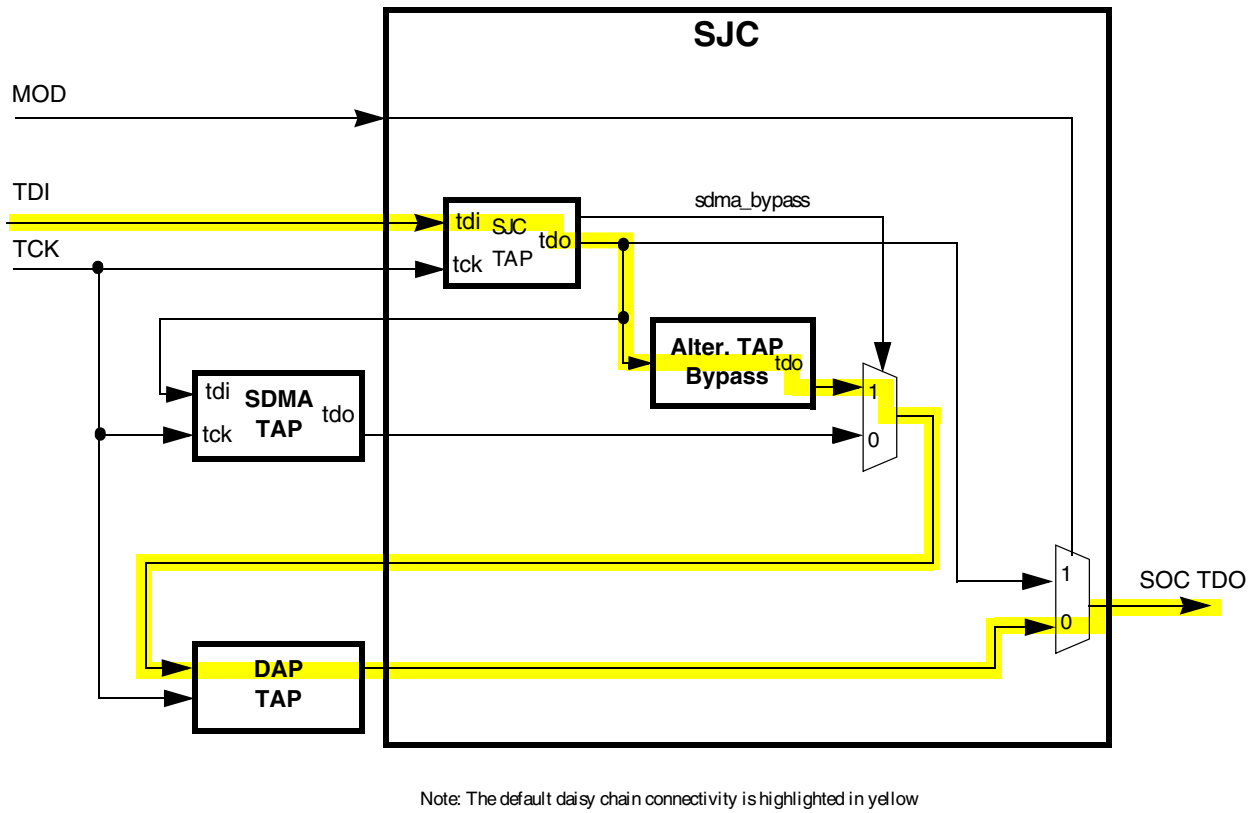
The Connect SDMA bit inside TAP select register controls the SDMA TAP bypass.

- When negated (should be the default state), the SDMA TAP is bypassed with a single D-FF (Flip-flop) during Shift-Dr path
- When asserted SDMA TAP is connected inside the chain
- When taking the SDMA into bypass or out of bypass (by writing to tapsel reg), additional cycle with TMS '0' should be given

The TAP selection block (TSB) provides a simple method of integrating various pieces of IP that have embedded TAPs.

- Provides a way to connect up multiple TAPs within a single SoC
- Identify the SJC TAP as the master TAP which controls the boundary chain (for IEEE 1149.1 standard compliance)
- Follow the state of SJC TAP, and when the Test-Logic-Reset (TLR) state is reached, reset all TAPs

The figure below shows the TAP Selection Block and SOC TAP Chain Scheme.



Note: The default daisy chain connectivity is highlighted in yellow

Figure 45-3. TAP Selection Block and SoC TAP Chain Scheme

NOTE

It is the responsibility of the user to ensure that in any configuration of the TAP controllers chosen, all of the TAPs in the chain comply with the demands of TCK clock frequency as well as the required ratio between TCK clock frequency and that of the core's to which the TAP refers.

45.2 External Signals

The table found here describes the external signals of SJC.

Table 45-2. SJC External Signals

Signal	Description	Pad	Mode	Direction
JTAG_DE_B	SoC debug request/acknowledge pin. The DE_IN_B pin is used to propagate an external debug request event to the core(s). This functionality must be enabled first, by set of DE_to_ARM /	UART2_CTS_B	ALT7	IO

Table continues on the next page...

Table 45-2. SJC External Signals (continued)

Signal	Description	Pad	Mode	Direction
	DE_to_SDMA bits in SJC's DCR register. It is SoC implementation dependent, whether this pin can also be used to reflect the debug acknowledge event back from the cores (in the case where an Open-Drain scheme is used externally).			
JTAG_MOD	SJC mode selection. This pin is sampled at TRST reset to determine two possible modes for the TAP connection configuration.	JTAG_MOD	No muxing	I
JTAG_TCK	Test Clock (TCK). This is used to synchronize the test logic and includes an internal pull-up resistor	JTAG_TCK	ALT0	I
JTAG_TDI	Test Data Input (TDI). Serial test instruction and data are received through the test data input (TDI) pin. TDI is sampled on the rising edge of TCK and includes an internal pullup resistor	JTAG_TDI	ALT0	I
JTAG_TDO	Test Data Output (TDO). The serial output for test instructions and data. TDO is tri-statable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK	JTAG_TDO	ALT0	O
JTAG_TMS	Test Mode Select (TMS). This is used to sequence the test controller's state machine. TMS is sampled on the rising edge of TCK and includes an internal pullup resistor	JTAG_TMS	ALT0	I
JTAG_TRSTB	Test Reset (TRST). This is used to asynchronously initialize the test controller. The TRST pin has an internal pullup resistor	JTAG_TRST_B	ALT0	I

45.2.1 External Signal Overview

The SJC provides test and debug control with a minimum number of contacts.

The figure below shows SJC connections to external contacts and other chip blocks.

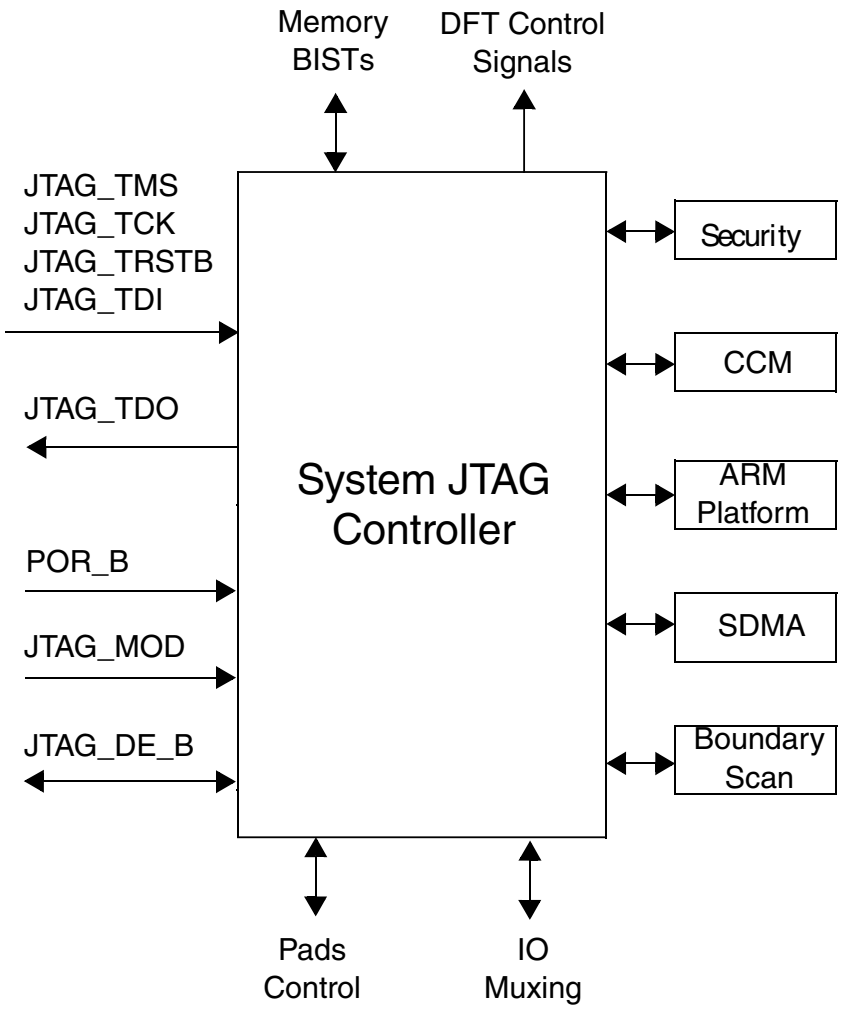


Figure 45-4. SJC Connections

45.2.2 TAP Controller

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, refer to the appropriate IEEE 1149.1 document.

The state machine is shown in the following figure.

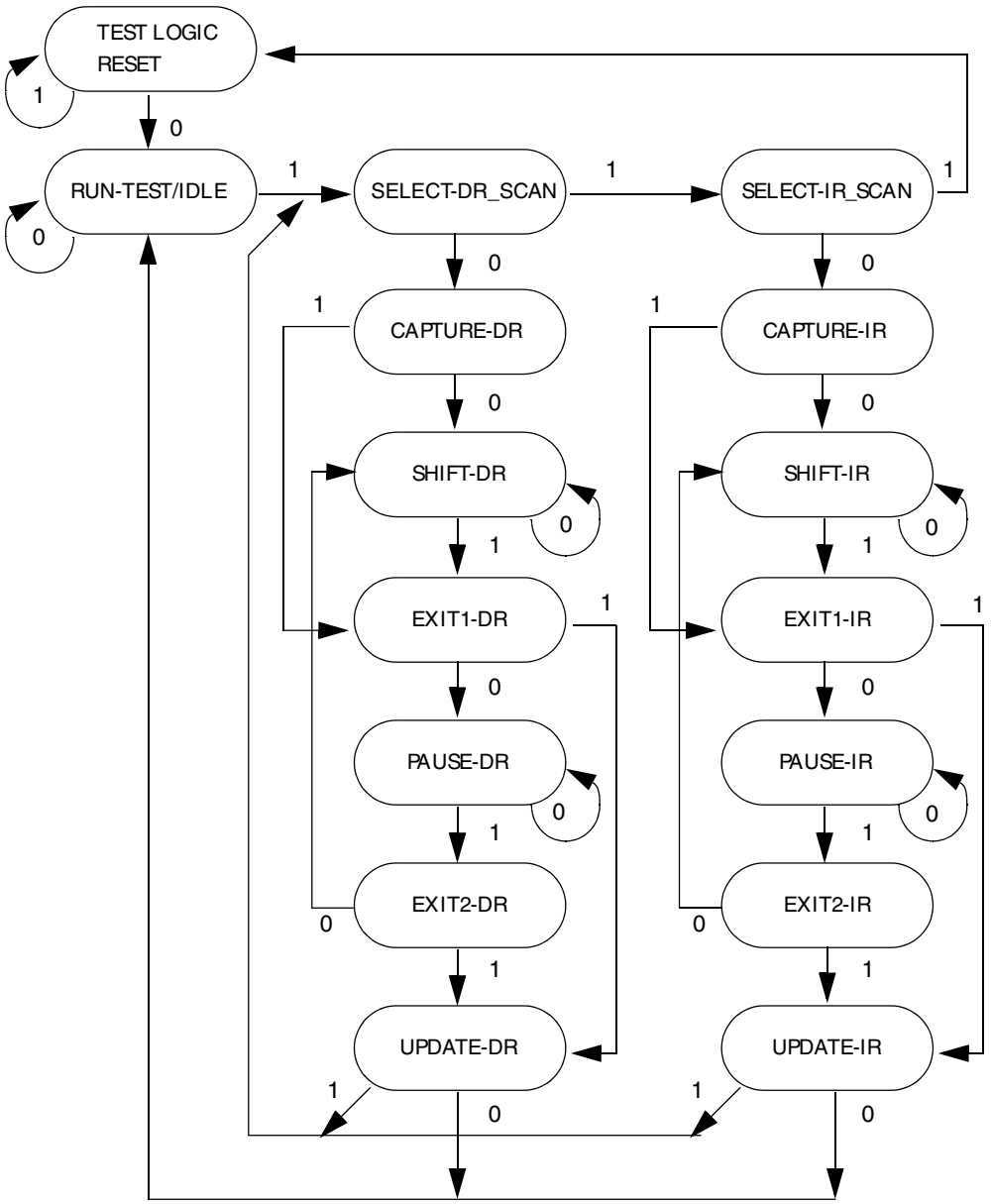


Figure 45-5. TAP Controller State Machine

The change of the JTAG state machine occurs on the rising edge of TCK. TMS and TDI change on the falling edge of TCK. TDO also changes on the falling edge of TCK following entry into the Shift_DR or Shift_IR states (TDO_EN is the enable of the tristate buffer driving the TDO output).

The figure below shows the timings of the SJC signals.

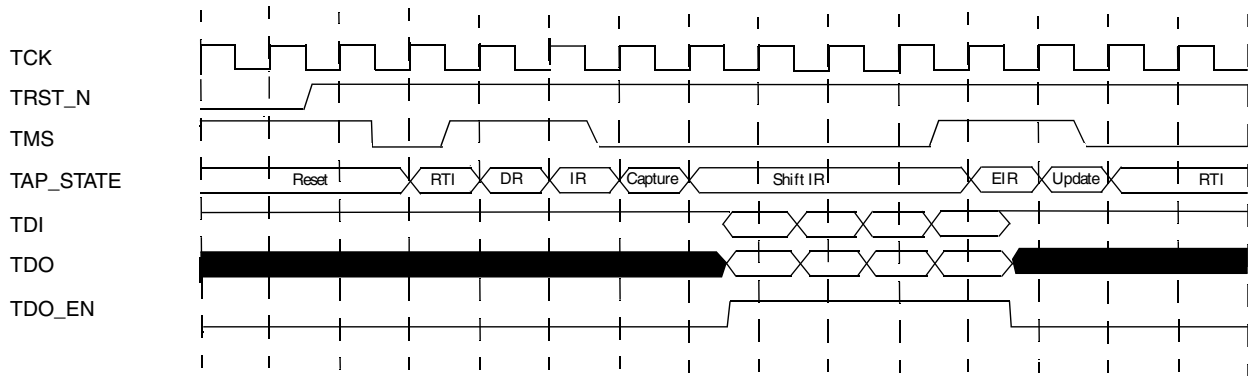


Figure 45-6. SJC Signals Timing Diagram

45.2.3 Accessing ExtraDebug Registers

Accessed through the Select-DR-Scan path, the ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bit data field (max length, see extradebug register description), a 5 bit address field and read/write bit.

The write actually takes place when the JTAG TAP controller enters the Update-DR state. On a read, the data field is ignored (the user should shift only 5 times to enter Read=1 and the address), the read takes place on the next path through DR at the Capture-DR state, the data is shifted-out during the Shift-DR state.

On the second path for a read access, simultaneous write access is not supported: command converter software shifts in zeros so the TAP decodes a write to the CSR (read-only register) which does not have any effect on the circuit.

The number of shift depends on the width of the accessed register as explained in the following diagrams.

First a write access (one path through Select-DR-Scan):

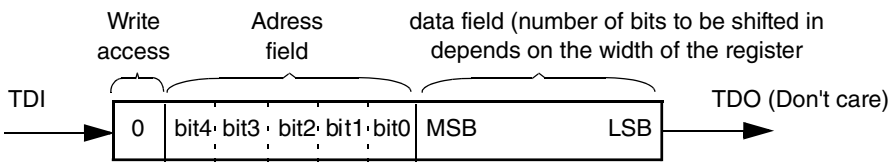


Figure 45-7. TDI/TDO on write access

Then a read access (requires two paths through Jtag DR Scan path):

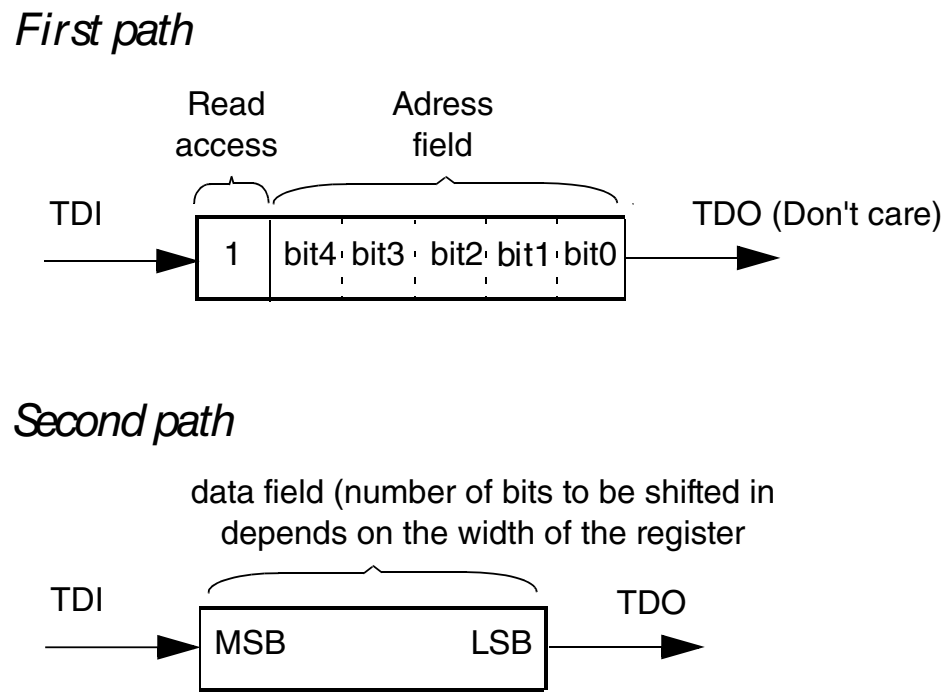


Figure 45-8. TDI/TDO on Read Access

For example, write value 0b1010_1100 to Debug Control Register (address = 0b00110).

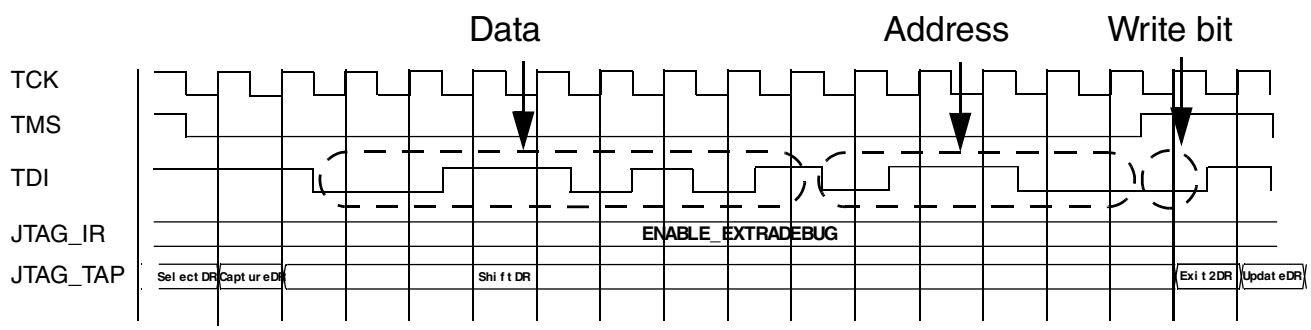


Figure 45-9. Example: Write Access to DCR

The SJC registers have different levels of security (Refer to JTAG Security Modes):

- Secured- accessible only in mode 2 (supposed correct response entered), mode 3 and mode 4.
- Unsecured- accessible in all modes

The level of security of each register is indicated in its name or description, in "Programmable Registers" section.

A single DE_B pin is dedicated for debug request input/output in bidirectional open drain functionality (including an internal pull-up device).

Bits 6:5 in DCR register serve as mask bits, controlling the propagation of external debug request to each recipients (ARM Platform, SDMA).

The bits 1:0 define the propagation enable of IR debug request to recipient cores.

Figure 45-10. DE_B Pin Select Logic

For security reasons, bits for output and input propagation control are at their negated values after reset. A user cannot put the cores in debug mode through DE_B without any Jtag access.

The configuration after reset prevents propagation of debug requests / acknowledges to or from the cores.

45.3 TAP Selection Block (TSB)

As described in Modes of Operation, the SJC can access cores in different modes selected through a TSB.

45.3.1 Select Mode Using Software

Conceptually, the SJC_TSR is a data register which is accessed through Access TSR IR instruction of SJC TAP.

The following figure shows the process of using reserved IR to access the SJC_TSR.

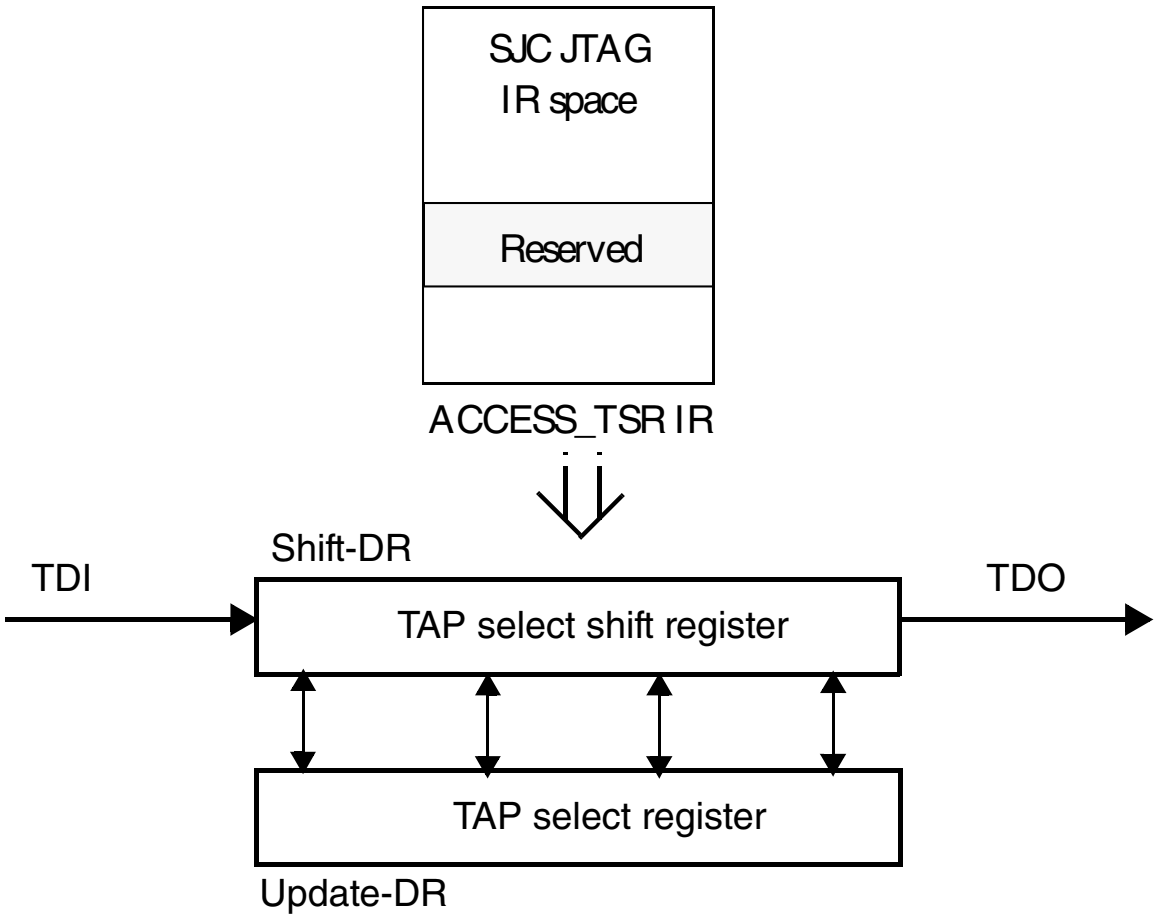


Figure 45-11. Using Reserved IR to Access the TAP Select Register (SJC_TSR)

The SJC_TSR can only be changed during the update-DR state of the TSB JTAG state machine. This is necessary to prevent a TAP that is being selected from losing synchronization with the TSB state machine when the TSB state machine returns to run-test-idle. Therefore, an associated shift register for the SJC_TSR is loaded into the SJC_TSR during the update-DR state (see the figure above). The shift register must also capture the state of the SJC_TSR when in the Capture-DR state for visibility of the contents of the SJC_TSR. See TAP Select Instruction , for more information.

45.4 Boundary Scan Register (BSR)

The Boundary Scan Register (BSR) in the JTAG implementation contains bits for all device signal and clock pins and associated control signals.

All SoC bidirectional pins have a single register bit in the boundary scan register for pin data, and are controlled by an associated control bit in the boundary scan register.

45.5 SoC JTAG Instruction Register (SJIR)

The SoC JTAG Instruction register can be found here.

The SoC JTAG Instruction register is 5 bits wide.

Table 45-3. SoC JTAG Instruction Register (SJIR)

Code					SJC IR
B4	B3	B2	B1	B0	
0	0	0	0	0	IDCODE
0	0	0	0	1	SAMPLE/PRELOAD
0	0	0	1	0	EXTEST
0	0	0	1	1	HI-Z
0	0	1	0	0	ENABLE_ExtraDebug
0	0	1	0	1	ENTER_DEBUG (secured)
0	0	1	1	0	Reserved
0	0	1	1	1	TAP select
0	1	0	0	0	EXTEST_PULSE
0	1	0	0	1	EXTEST_TRAIN
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Security Output challenge
0	1	1	0	1	Security Enter response
-					Reserved
1	1	1	1	1	BYPASS

The instruction register is reset to 0b00000 in the test-logic-reset controller state which is equivalent to the IDCODE instruction.

During the capture-IR controller state, the parallel inputs to the instruction register are loaded with the code 01 in the least significant bits as required by the standard; the most significant bits are loaded with the values 00, leading to a capture value of 0b00001.

45.5.1 ID_CODE Instruction (IDCODE)

Selects the ID register, and the system logic controls the I/O pins. This instruction is provided as a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP.

The table below shows the ID register configuration.

Table 45-4. ID Configuration Register (IDCODE)

IDCODE				ID Configuration Register												
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Version Information[3:0]				Part Number (Bits 27-16)											
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
Note:																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Part Number (Bits 15-12)				Manufacturer Identity											1
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	x	x	x	x	0	0	0	0	0	0	0	1	1	1	0	1
Note:																

Table 45-5. ID Configuration Register Description (IDCODE)

Field	Description
31-28 Version Information	IC/SoC Version information number. Initial value: '0000' This number is subject to changes, for new IC/SoC (System On A Chip) revision releases.
27-12 Part Number	Customer Part Number The 16-bit Part Number value is unique for every Freescale's SoC / IC. See System Debug chapter for exact register value for a specific SoC.
11-1 Manufacturer Identity	Manufacturer Identity Freescale's Manufacturer Identity code. Bits [11:1] - 00000001110
0	Tied to logic 1.

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to the IEEE 1149.1 standard, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design to determine the type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Once the IDCODE instruction is decoded, it selects the ID register which is a 32 Bit data register. Because the bypass register loads a logic 0 at the start of a scan cycle, whereas the ID register loads a logic 1 into its least significant bit, examination of the first bit of data shifted out of a component during a test data scan sequence immediate following exit from Test-Logic-Reset controller state shows whether such a register is included in the design. When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic as required by the IEEE 1149.1 standard.

45.5.2 SAMPLE/PRELOAD Instruction

Selects the boundary scan register and the system logic controls the I/O pins.

The SAMPLE/PRELOAD instruction provides two separate functions:

- First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. The data can be observed by shifting it transparently through the boundary scan register.
- The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data appears on the outputs when entering the EXTEST instruction.

NOTE

Because there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

For more details on the function and use of SAMPLE/PRELOAD, refer to the appropriate IEEE 1149.1 document.

45.5.3 EXTEST Instruction

Selects the boundary scan register, and the 1149.1 test logic has control of the I/O pins.

By using the TAP controller, the register is capable of:

- Scanning user-defined values into the output buffers,
- Capturing values presented to input pins
- Controlling the direction of bidirectional pins,
- Controlling the output drive of tri-statable output pins.

For more details on the function and use of EXTEST, refer to the appropriate IEEE 1149.1 document.

The EXTEST instruction also asserts internal reset for the cores (through CCM, refer to Figure 45-14) to force a predictable internal state while performing external boundary scan operations.

45.5.4 HIGHZ Instruction

All output drivers, including the two-state drivers, are turned off (that is, high impedance). The instruction selects the bypass register.

In this mode, all internal pullup resistors on all the pins (except for the TMS, TDI, TCK, TRSTB pins) are disabled. This disabling functionality is not built into SJC, but should be implemented by some logic in the SOC/IO Pads.

For more details on the function and use of HIGHZ, refer to the IEEE 1149.1 document.

The HIGHZ instruction also asserts internal reset for the cores (through CCM, refer to Figure 45-14) to force a predictable internal state while performing external boundary scan operations.

45.5.5 BYPASS Instruction

Selects the single Bit bypass register and the system logic controls the I/O pins.

This creates a shift-register path from TDI to the bypass register and, finally, to TDO, circumventing the boundary scan register. This instruction is used to enhance test efficiency when a component other than the SoC Core based device becomes the device under test.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register is always a logic zero.

For more details on the function and use of BYPASS, refer to the appropriate IEEE 1149.1 document.

45.5.6 ENABLE_ExtraDebug Instruction

The TDI and TDO pins are connected directly to the ExtraDebug registers, the SJC TAP controller remaining connected to TDI and TMS.

The ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bits data field (maximum length, see Accessing ExtraDebug Registers,), a 5 bits address field and read/$\overline{\text{write}}$ bit. On a register read, the data field does not need to be filled in. The particular ExtraDebug register connected between TDI and TDO at a given time is selected by the ExtraDebug controller depending on the ExtraDebug Address being currently decoded. All communication with the ExtraDebug controller is done through the Select-DR-Scan path of the JTAG TAP Controller.

45.5.7 ENTER_DEBUG instruction

The ENTER_DEBUG instruction is used to generate a debug request event to SDMA and the ARM Core Platform simultaneously (practically, inherited minimal skew is expected, due to difference in event signal propagation in the different modules).

The TDI and TDO are connected to the Instruction Register (IR). After the acknowledgment of the Debug Mode is received (can be checked by reading the Core Status Register part of the ExtraDebug logic), the user can perform system debug functions on the cores.

NOTE

The ENTER_DEBUG event issue to the cores, can be masked, by bits in DCR register.

NOTE

It is user's responsibility to shift-in another IR value (like IDCODE) before trying to bring the cores out of debug mode, as the debug request signals to the cores remains asserted as long as ENTER_DEBUG IR is in place.

NOTE

The user need to check that cores are in debug mode (watching debug acknowledge signal) before leaving ENTER_DEBUG instruction, otherwise debug request might not take affect.

45.5.8 TAP Select Instruction

By means of TAP select instruction a user can access TAP select register and by controlling its only bit SDMA Bypass, control whether SDMA TAP is bypassed or not.

Table 45-6. TAP Select Register (TSR)

	TAP Select Register
	BIT 0
	Connect SDMA
TYPE	rw
RESET	0
Note:	

Table 45-7. TAP Select Register Description

Field	Description
0 SDMA Bypass	Connect SDMA Control whether SDMA TAP is bypassed or not: • 0 - SDMA TAP is bypassed by the alternate TAP inside SJC (emulating 4-bit IR and 1-bit bypass path). • 1 - SDMA TAP is connected to the TDI-TDO chain. **NOTE:** Additional cycle with TMS '0' should be inserted, after writing to this register, to allow the SDMA tap be sync before SDMA get into / out of bypass.

45.5.9 EXTEST_PULSE instruction

The EXTEST_PULSE instruction implements new test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

45.5.10 EXTEST_TRAIN instruction

The EXTEST_TRAIN instruction implements new test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

45.6 Security

JTAG manipulation is one of the known hackers' ways of executing unauthorized program code, getting control over the OS and run code in privileged modes.

The SJC provides a debug access to several H/W blocks including the ARM processor and the system bus. This allows for program control and manipulation as well as visibility into system peripherals and memory. The ETM and NEXUS interfaces allow bus transactions to be traced. Together these tools provide the hacker all the access needed to completely comprise the system. Means must be provided to block any malicious JTAG access.

The SJC provides a way of regulating the JTAG access.

The following are the different JTAG security modes:

- **Mode #1: No Debug**-Maximum Security. All security sensitive JTAG features are permanently blocked.
- **Mode #2: Secure JTAG**-High security. JTAG use is regulated by secret key based authentication mechanism.
- **Mode #3: JTAG Enabled**-Low security. JTAG always enabled.

The JTAG security modes are configured using eFUSEs which can burned after packaging by applying electrical signals. The fuse burning is irreversible process, once a fuse is burned (e-fuse or laser fuse) it is impossible to change the fuse back to the un-burned state.

45.6.1 JTAG Security Modes

JTAG can be in one of JTAG security modes which is selected by setting the SJC eFUSE configuration. The physical location of the fuses is not in the SJC.

45.6.1.1 Mode 1: No Debug - Maximum Security

No Debug JTAG security mode provides the highest security level.

In this mode, all JTAG features are disabled except for:

- ScanBoundary Scan
- MBIST, all modes except for debug modes which enable controlled memory contents output
- PLL BIST
- BIST monitor mode, allowing routing to external pins BIST pass/fail/invoke information
- PLL bypass- Bypass ARM or/and USB PLL.
- Visibility of the following status bits: power mode - normal, standby, stop, shutdown, and so on

These features do not reduce the security level of the product, and they allows to perform important tests and board connectivity checks.

45.6.1.2 Mode 2: Secure JTAG - High Security

The Secure JTAG mode limits the JTAG access by using challenge/response based authentication mechanism. Any access to JTAG port is being checked. Only authorized debug devices (that is, devices having the right response) can access the JTAG, unauthorized JTAG access attempts are denied.

The intent of this mode is to allow return field testing. When a secured JTAG device is being returned for debugging, this mode allows authorized re-activation of the JTAG.

45.6.1.2.1 Challenge/Response Mechanism in System JTAG Mode

When SJC is in Sysytem JTAG mode the authentication process is as follows:

1. Shift Output Challenge instruction to IR.
2. Passing through Capture-DR state of the SJC and by performing Shift-DR operations Challenge code can be accessed from TDO.
3. Shift Enter Response instruction to IR. By performing Shift-DR, operations enter Response code value through TDI. As Update-DR state is entered, Response code is compared with the correct one.

In Fixed challenge-response pair mode, each part has its individual challenge - response pair which is determined at manufacturing time, and does not change later on. The SJC compares the user's response to the expected response.

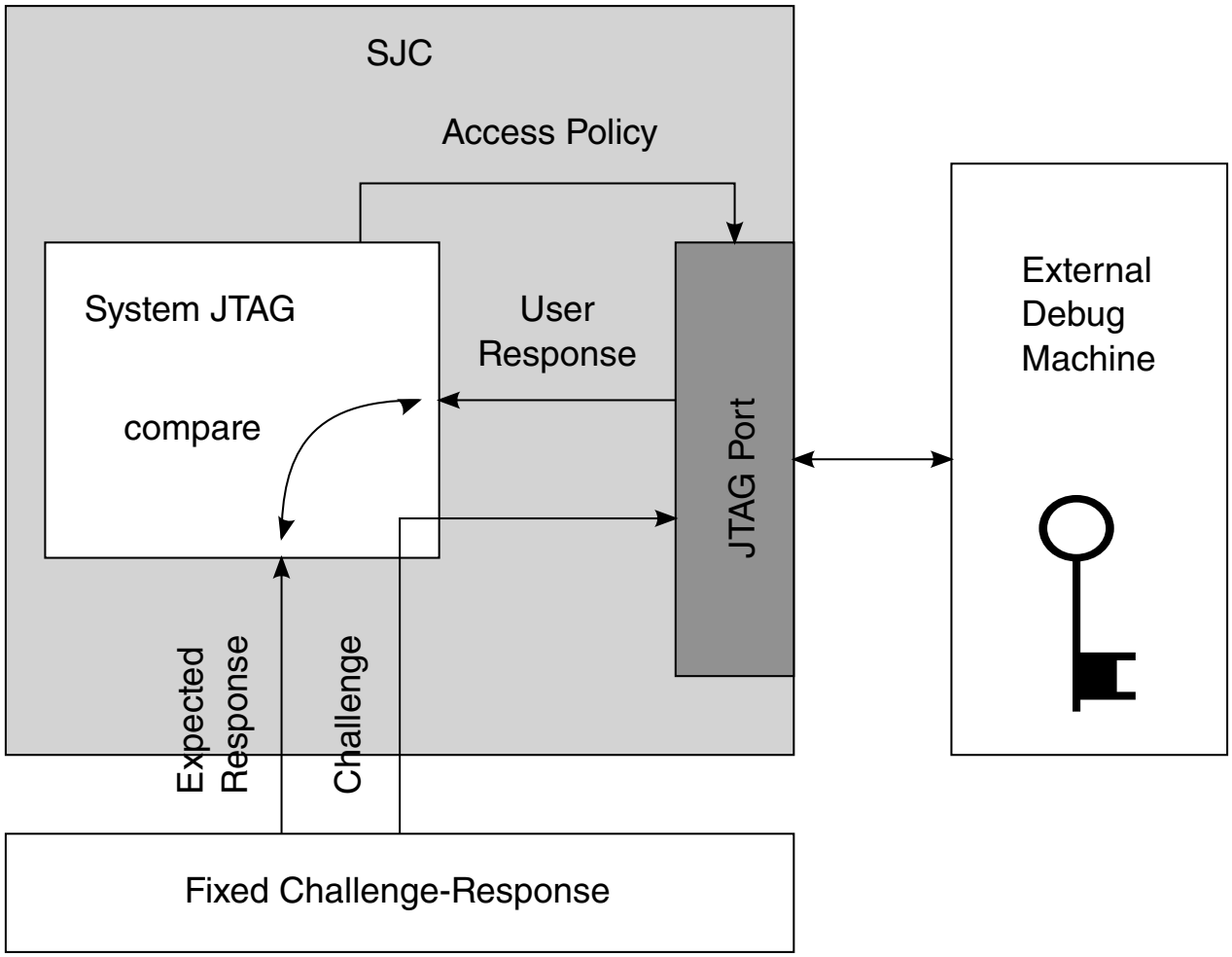


Figure 45-12. Mode #2 - Secure JTAG with Fixed Challenge-response Pair

45.6.1.3 Mode 3: JTAG Enabled - Low Security

In the JTAG Enabled JTAG security mode, all JTAG features are enabled.

45.6.2 Software Enabled JTAG

To increase the flexibility of the SJC, an option to enable the JTAG via software is added and is available only in Secure JTAG mode. By writing '1' to HAB_JDE (HAB JTAG DEBUG ENABLE) bit in the e-fuse controller module, the JTAG is opened, regardless of its security mode. It is the responsibility of software to assert or negate this bit.

Additionally, a corresponding lock bit is available (in the e-fuse control module) to ensure that only trusted software is able to set the JDE bit. When the LOCK bit is set, no future change of JDE is possible, until the next POR (power-on-reset) cycle.

The platform initialization software should set the LOCK bit for JDE bit before transferring control to the application code.

The S/W JTAG enable allows JTAG enabling without activating the challenge-Response mechanism (which requires JTAG access tool enhancement or special H/W). The JTAG S/W enable does not allow debug in case of boot or memory fault as it requires reset before entering debug.

This feature can be permanently blocked by burning the dedicated e-fuse.

NOTE

The S/W enabled JTAG feature reduces the overall security level of the system as it relies on S/W protections. If this feature is not required, it is strongly recommended to burn the JTAG_HEO e-fuse which disables this feature.

45.6.3 Kill Trace

The kill trace signal disables any output of the ETM block. The ETM can be accessed either via JTAG port and/or by direct software code. Blocking the JTAG port also yields assertion of the kill trace signal. This resulted in blocking of trace port. The intention of this action is to block any attempt to break into the system via software manipulation of the debug modules. The kill trace, when active, prevents trace output even in case where it can be activated via chip pin.

The kill trace feature needs to be activated by burning a dedicated e-fuse. If the fuse is left intact, kill trace is never activated as seen in Figure 45-13.

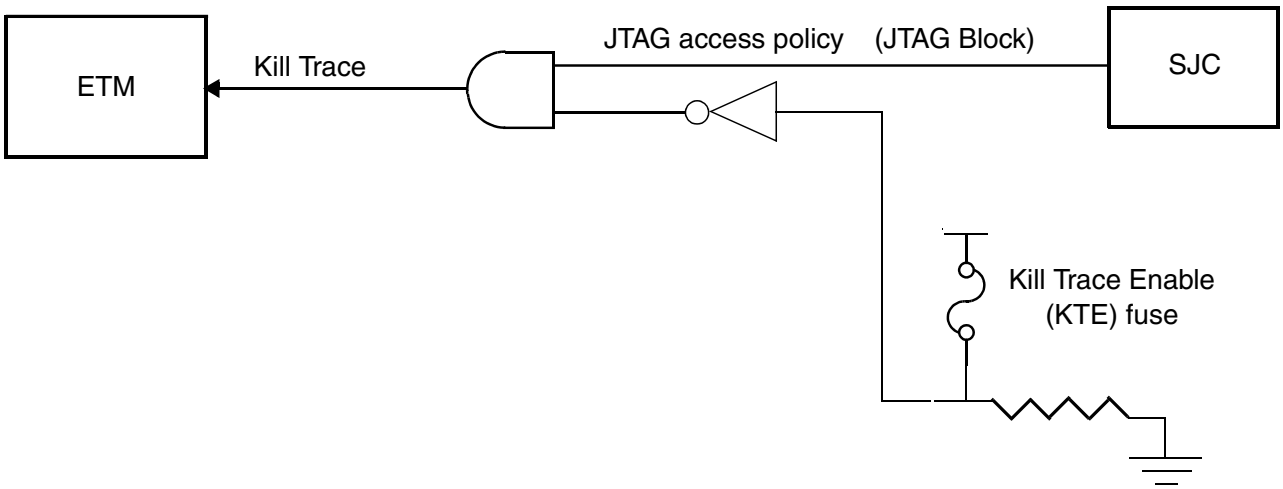


Figure 45-13. Kill Trace eFUSE

The kill trace is asserted when "kill trace enable" fuse is burned and "ipt_secur_block" signal in SJC is asserted, which happens when at least one of the following is true:

- Mode #2 (Secure JTAG) and no code has been entered
- Mode #2 (Secure JTAG) with burned Bypass and Re-enable fuses
- Mode #2 (Secure JTAG) with incorrect response entered
- Mode #1 (No debug)
- TRST_B signal is active
- POR has not ever been asserted

45.6.4 SJC Disable Fuse

In addition to the different JTAG security modes that are implemented internally in the System JTAG Controller (SJC), there is an option to disable the SJC functionality by eFUSE configuration. This creates additional JTAG mode that is, JTAG Disabled with highest level of JTAG protection. In this mode all JTAG features are disabled.

Specifically, the following debug features are disabled in addition to the features that were already disabled in No Debug JTAG mode:

- Memory BIST
- Boundary scan register (SJC_BSR)
- Non-Secure JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secure JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

45.7 Functional Description

This section provides a complete functional description of the block.

45.7.1 Static Core Debug

The SJC JTAG TAP controller is fully compatible with the IEEE 1149.1a-2001 Standard Test Access Port and Boundary Scan Architecture specifications.

The ARM platform has an integrated JTAG interface and a TAP controller to manage its own ICE. Also it can access an embedded trace ETM interface, see ARM core and ETM Technical reference guide for more information.

The SDMA has a TAP controller to manage its own OnCE, see SDMA OnCE specifications for more details.

The OnCE and ICE provide a mean of interacting with the cores and their peripherals non-intrusively so that a user may examine registers, memories to facilitate hardware and software development. Refer to TAP Selection Block (TSB), for more information.

45.7.2 Reset Mechanism

The following figure shows the SJC reset logic

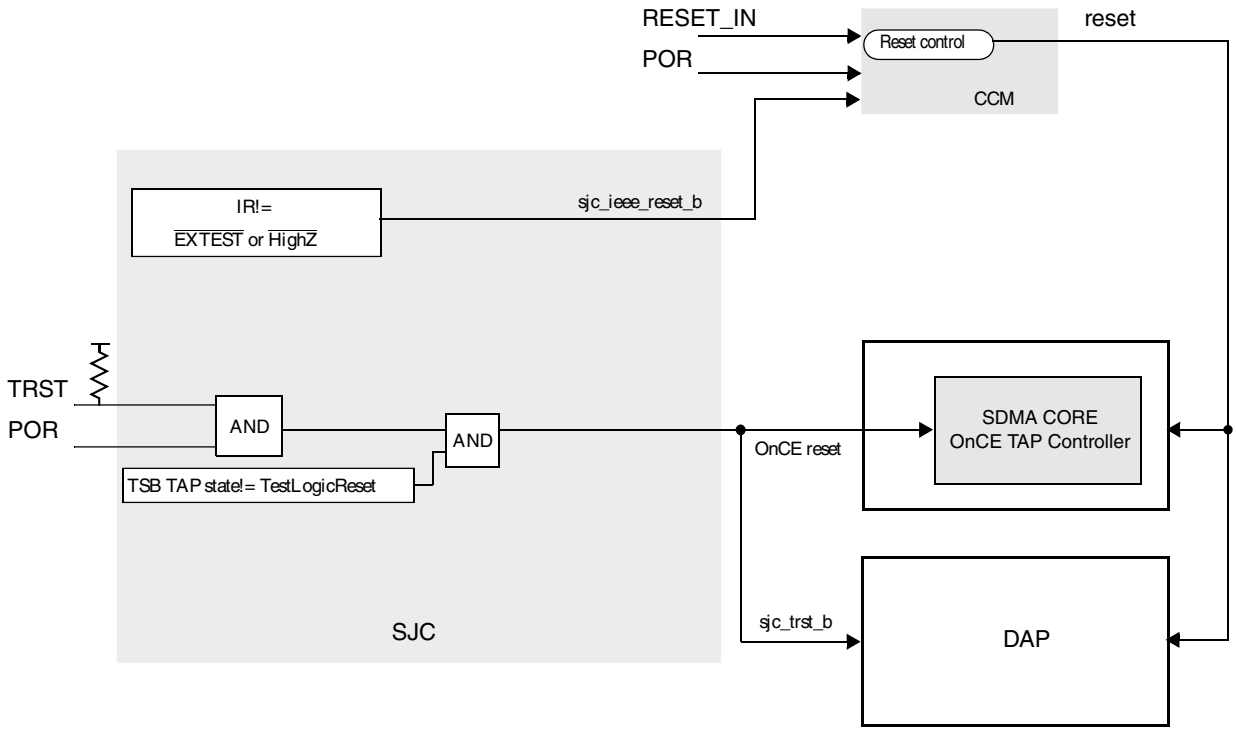


Figure 45-14. SJC Reset Logic

NOTE

- Asserting TRSTB in any scan mode resets the TCR loosing the testmode configuration and selects default TAP.
- SJC generates an IEEE reset signal to the CCM when in one of the IEEE modes HIGHZ or EXTEST. This signal generates a system reset to the cores until exit from one of these modes.
- The TSB generates Once/ICE reset (either TRSTB if implemented or other) when its TAP state reaches Test-Logic-Reset (meaning that TAP accessed is also reaching Test-Logic-Reset).

45.8 Initialization/Application Information

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the SJC output drivers are enabled into actively driven networks.

There are two constraints related to the JTAG interface:

- Ensure that the JTAG test logic is kept transparent to the system logic by forcing TAP into the Test-Logic-Reset controller state. During power-up, SJC's internal TRSTB is asserted as IC's POR_B is asserted which forces the TAP controller into this state. After that, if TMS either remains unconnected or is connected to VCC, then the TAP controller cannot leave the Test-Logic-Reset state, regardless of the state of TCK.
- DE_B is an IO pin with pullup and care must be taken of the direction when driving this signal.

45.9 SJC Memory Map/Register Definition

In addition to the standard accessible JTAG registers (per IEEE1149.1 standard) listed in SoC JTAG Instruction Register (SJIR) , the chip contains the following registers accessed using the ExtraDebug mechanism, controlled via "ENABLE_ExtraDebug" IR instruction.

NOTE

SJC registers are only accessible by JTAG interface. They are not memory mapped to processor address space, so the absolute addresses provided by default in the SJC memory map are not valid.

This section assumes the JTAG controller is accessed in standalone mode or daisy chained (defined by TAP Selection Block) using the appropriate TSB configuration.

See "System Debug" chapter for more details about the general purpose register descriptions that are unique to this chip.

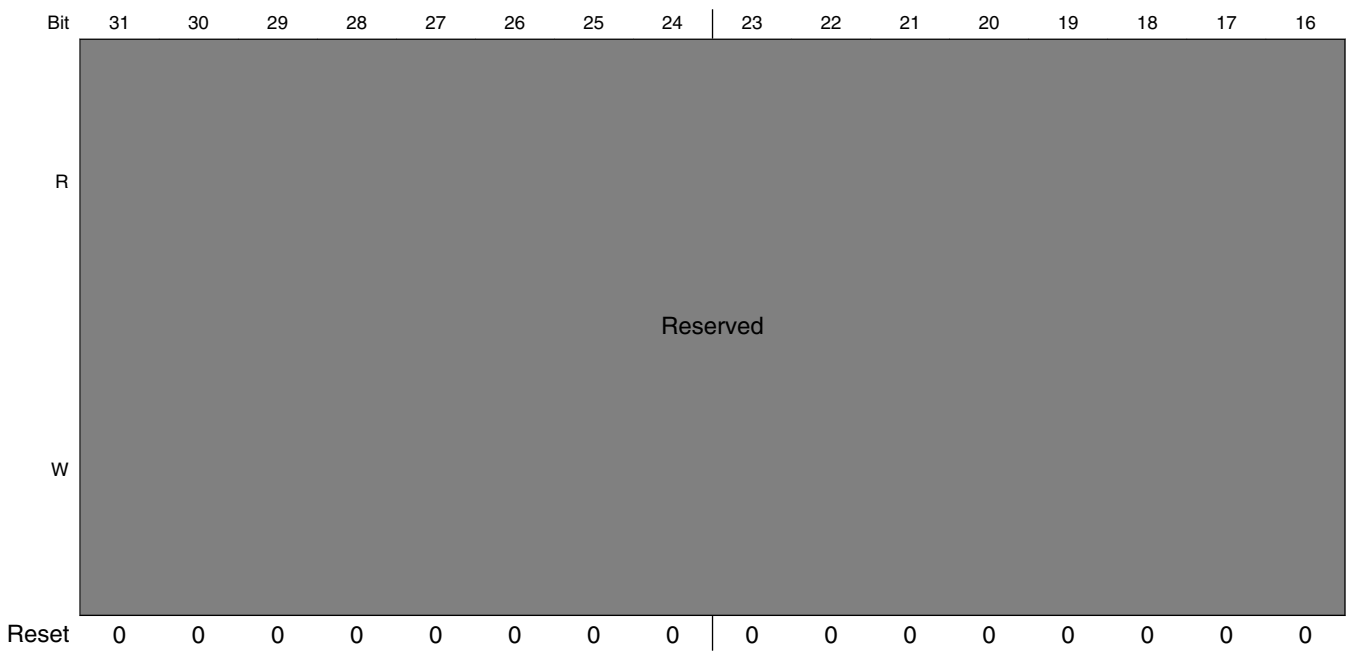
SJC memory map

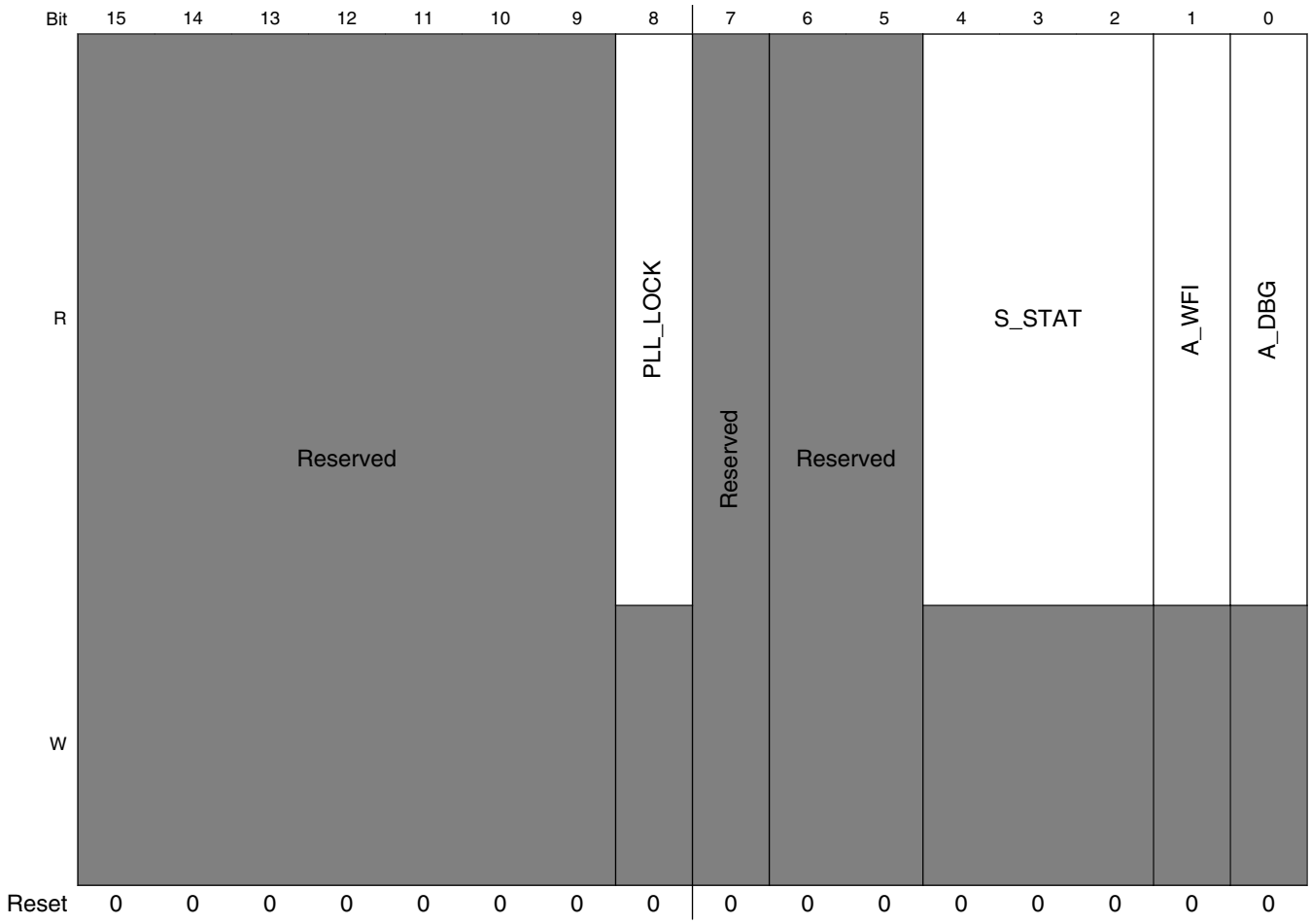
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	General Purpose Unsecured Status Register 1 (SJC_GPUSR1)	32	R	0000_0000h	45.9.1/2911
1	General Purpose Unsecured Status Register 2 (SJC_GPUSR2)	32	R	0000_0000h	45.9.2/2913
2	General Purpose Unsecured Status Register 3 (SJC_GPUSR3)	32	R	0000_0000h	45.9.3/2913
3	General Purpose Secured Status Register (SJC_GPSSR)	32	R	0000_0000h	45.9.4/2914
4	Debug Control Register (SJC_DCR)	32	R/W	1F9E_0000h	45.9.5/2915
5	Security Status Register (SJC_SSR)	32	R	See section	45.9.6/2917
7	General Purpose Clocks Control Register (SJC_GPCCR)	32	R/W	0C00_05A0h	45.9.7/2920

45.9.1 General Purpose Unsecured Status Register 1 (SJC_GPUSR1)

The General Purpose Unsecured Status Register 1 is a read only register used to check the status of the different Cores and of the PLL. The rest of its bits are for general purpose use.

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



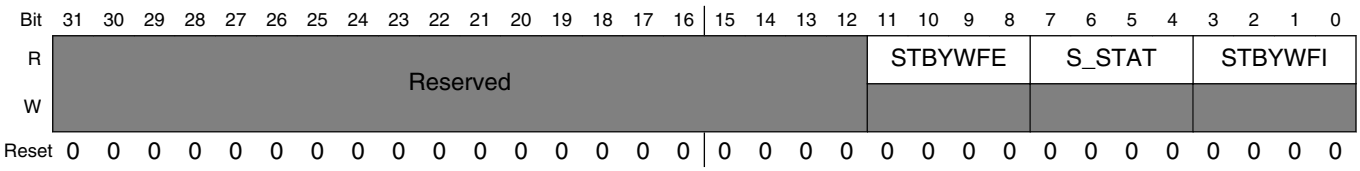


SJC_GPUSR1 field descriptions

Field	Description
31–9 -	This field is reserved. Reserved.
8 PLL_LOCK	PLL_LOCK A Combined PLL-Lock flag indicator, for all the PLL's.
7 -	This field is reserved. Reserved
6–5 -	This field is reserved. Reserved.
4–2 S_STAT	3 LSBits of SDMA core statusH.
1 A_WFI	ARM core wait-for interrupt bit Bit 1 is the ARM core standbywfi (stand by wait-for interrupt). When this bit is HIGH, ARM core is in wait for interrupt mode.
0 A_DBG	ARM core debug status bit Bit 0 is the ARM core DBGACK (debug acknowledge) DBGACK can be overwritten in the ARM core DCR to force a particular DBGACK value. Consequently interpretation of the DBGACK value is highly dependent on the debug sequence. When this bit is HIGH, ARM core is in debug.

45.9.2 General Purpose Unsecured Status Register 2 (SJC_GPUSR2)

Address: 0h base + 1h offset = 1h

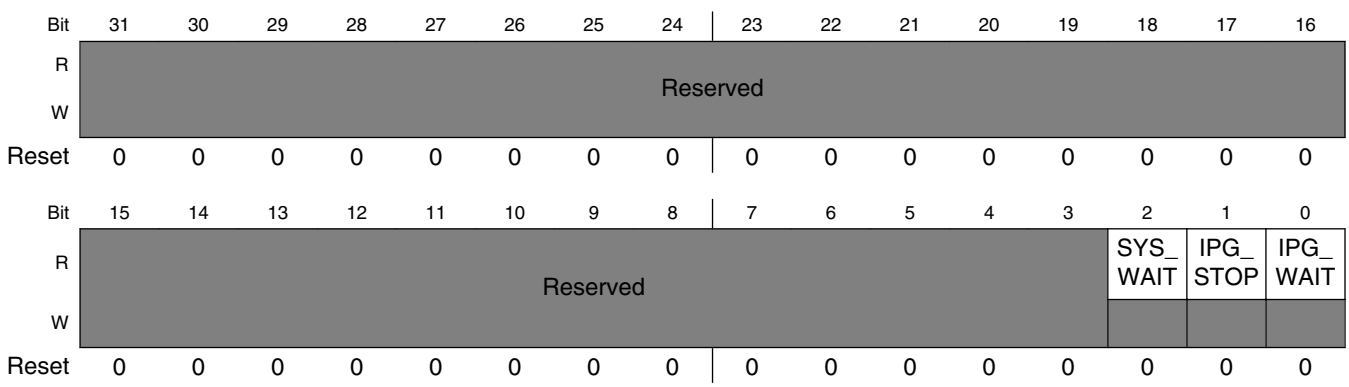


SJC_GPUSR2 field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 STBYWFE	STBYWFE[3:0] Reflecting the "Standby Wait For Event" signals of all cores.
7–4 S_STAT	S_STAT[3:0] SDMA debug status bits: debug_core_state[3:0]
STBYWFI	STBYWFI[3:0] These bits provide status of "Standby Wait-For-Interrupt" state of all ARM cores.

45.9.3 General Purpose Unsecured Status Register 3 (SJC_GPUSR3)

Address: 0h base + 2h offset = 2h



SJC_GPUSR3 field descriptions

Field	Description
31–3 -	This field is reserved. Reserved

Table continues on the next page...

SJC_GPUSR3 field descriptions (continued)

Field	Description
2 SYS_WAIT	System In wait Indication on System in wait mode (from CCM).
1 IPG_STOP	IPG_STOP CCM's "ipg_stop" signal indication
0 IPG_WAIT	IPG_WAIT CCM's "ipg_wait" signal indication

45.9.4 General Purpose Secured Status Register (SJC_GPSSR)

The General Purpose Secured Status Register is a read-only register used to check the status of the different critical information in the SoC. This register cannot be accessed in secure modes.

Address: 0h base + 3h offset = 3h

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

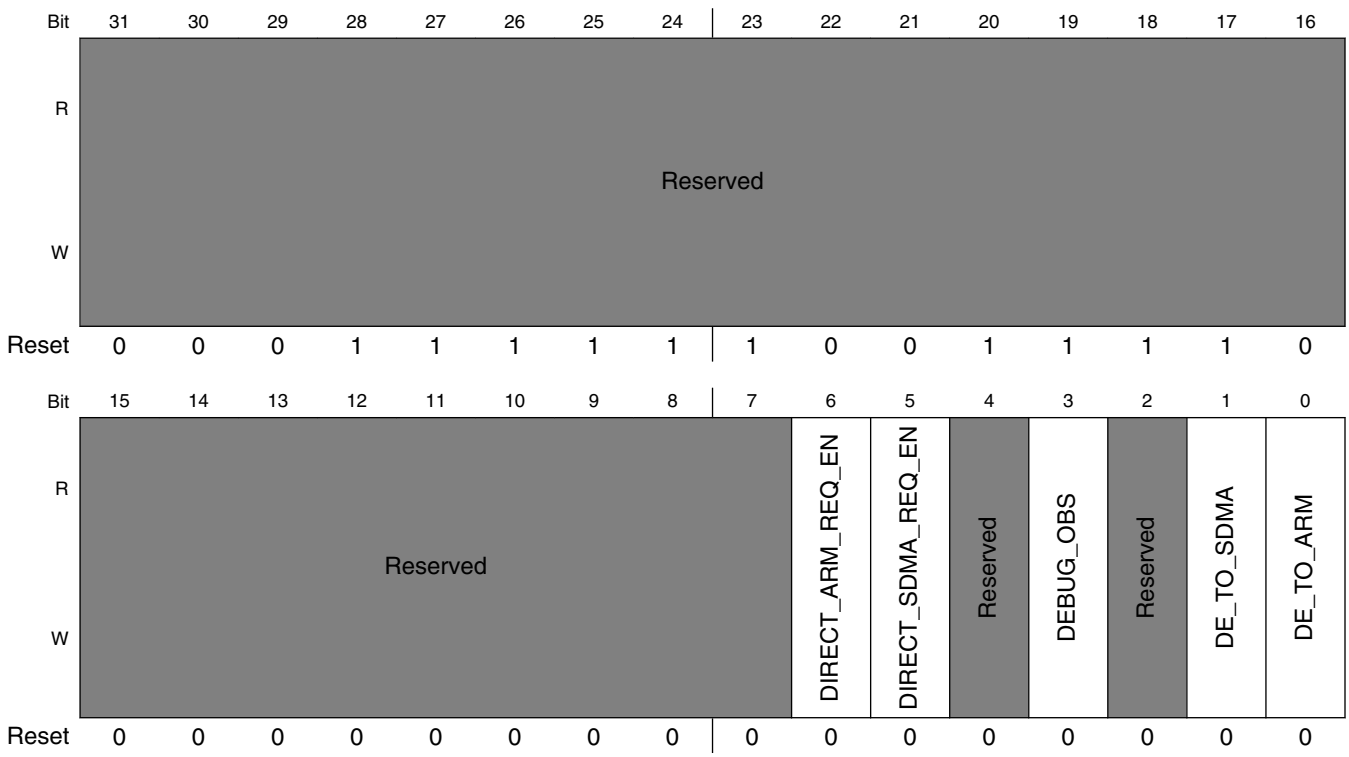
SJC_GPSSR field descriptions

Field	Description
GPSSR	General Purpose Secured Status Register Register is used for testing and debug.

45.9.5 Debug Control Register (SJC_DCR)

This register is used to control propagation of debug request from DE_B pad to the cores and debug signals from internal logic to the DE_B pad.

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



SJC_DCR field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6 DIRECT_ARM_REQ_EN	Pass Debug Enable event from DE_B pin to ARM platform debug request signal(s). This bit controls the propagation of debug request DE_B to the Arm platform. 0 Disable propagation of system debug to (DE_B pin) to Arm platform. 1 Enable propagation of system debug to (DE_B pin) to Arm platform.
5 DIRECT_SDMA_REQ_EN	Debug enable of the sdma debug request This bit controls the propagation of debug request DE_B to the sdma. 0 Disable propagation of system debug to (DE_B pin) to sdma. 1 Enable propagation of system debug to (DE_B pin) to sdma.
4 -	This field is reserved. Reserved

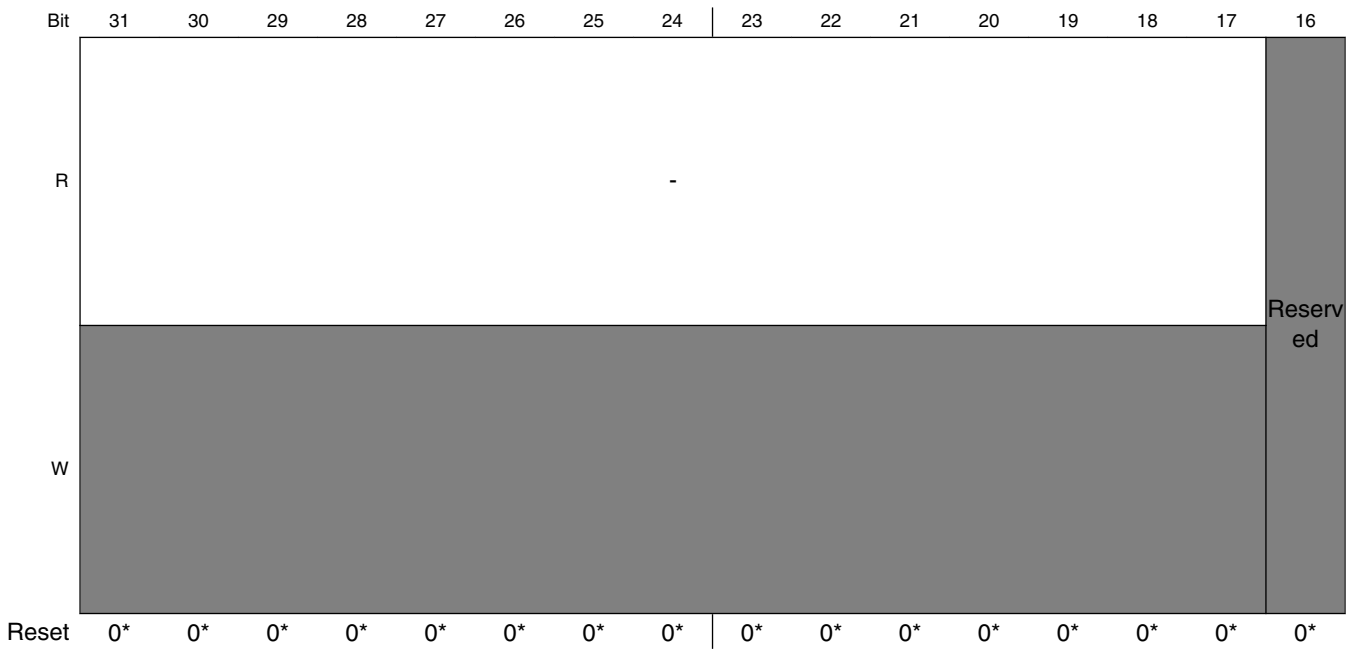
Table continues on the next page...

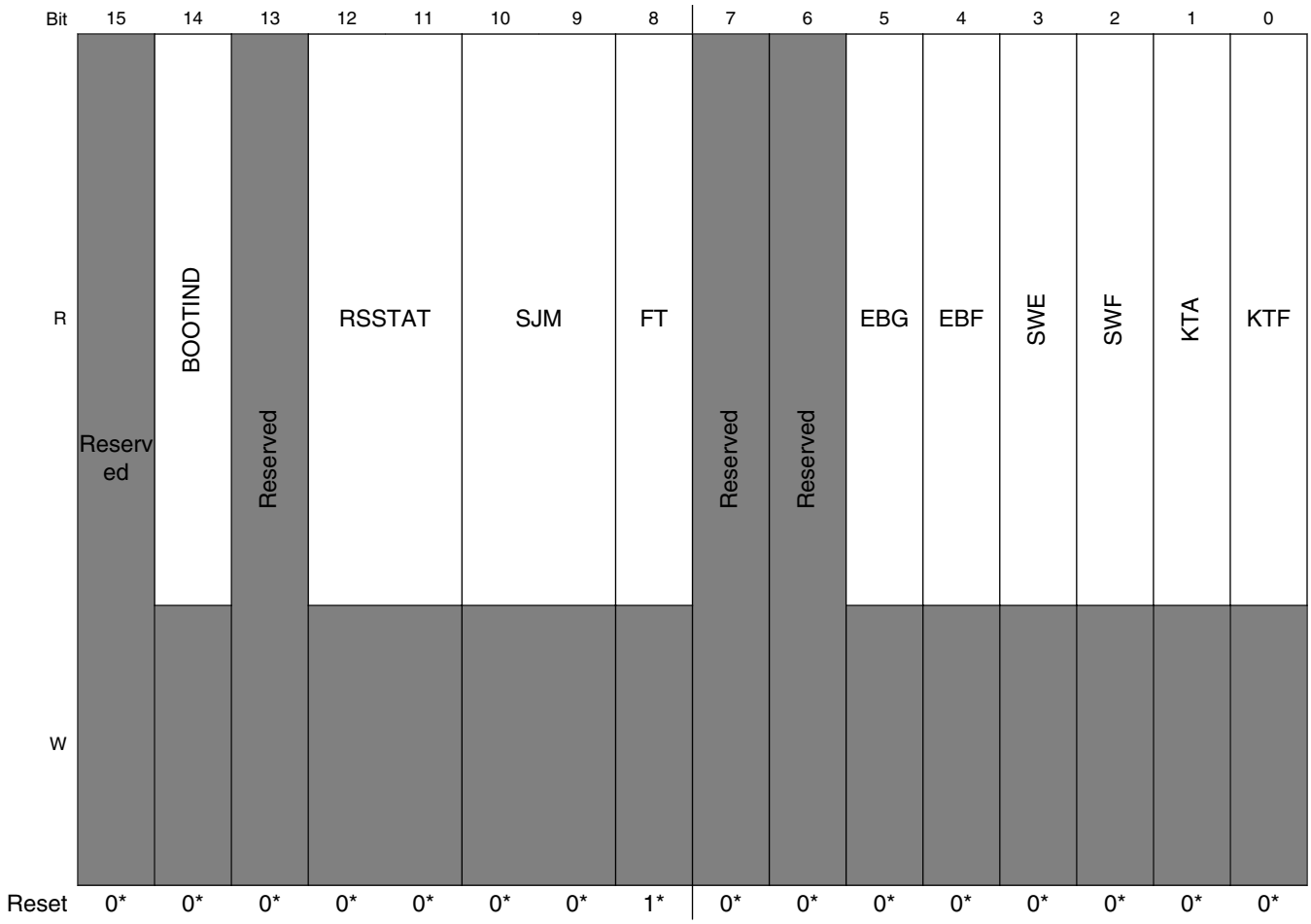
SJC_DCR field descriptions (continued)

Field	Description
3 DEBUG_OBS	Debug observability This bit controls the propagation of the "system debug" input to SJC (driven by the ECT logic), to the DE_B pad. (This logic can be used to pass debug acknowledge event from ECT out to the PAD, for example). The SJC's "system_debug" input is tied to logic HIGH value, therefore, set of "debug_obs" bit, will result in unconditional assertion of DE_B pad. 0 Disable propagation of system debug to DE_B pin 1 Unconditional assertion of pad DE_B
2 -	This field is reserved. Reserved
1 DE_TO_SDMA	SDMA debug request input propagation This bit controls the propagation of debug request to SDMA, when the JTAG state machine is put in "ENTER_DEBUG" IR instruction.. 0 Disable propagation of debug request to SDMA 1 Enable propagation of debug request to SDMA
0 DE_TO_ARM	ARM platform debug request input propagation This bit controls the propagation of debug request to ARM platform ("dbgreq"), when the JTAG state machine is put in "ENTER_DEBUG" IR instruction. 0 Disable propagation of debug request to ARM platform 1 Enable propagation of debug request to ARM platform

45.9.6 Security Status Register (SJC_SSR)

Address: 0h base + 5h offset = 5h





* Notes:
- The SJM reset value, reflects the JTAG security state, as defined by status of JTAG_SMODE[1:0] fuses. See the SJM bitfield description for details on valid values.

SJC_SSR field descriptions

Field	Description
31–17 -	Reserved.
16–15 -	This field is reserved. Reserved
14 BOOTIND	Boot Indication Inverted Internal Boot indication, i.e inverse of SRC: "src_int_boot" signal
13 -	This field is reserved. Reserved
12–11 RSSTAT	Response status Response status bits 00 Response wasn't entered 01 Response was entered but not verified

Table continues on the next page...

SJC_SSR field descriptions (continued)

Field	Description
	10 Response was entered and is incorrect 11 Response is correct
10–9 SJM	SJC Secure mode Secure JTAG mode, as set by external fuses. 00 No debug (#1) 01 Secure JTAG (#2) 10 Reserved 11 JTAG enabled (#3)
8 FT	Fuse type Fuse type bit - e-fuse or laser fuse 0 E-fuse technology 1 Laser fuse technology
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5 EBG	External boot granted External boot enabled, requested and granted 1 granted 0 not granted
4 EBF	External Boot fuse Status of the external boot disable fuse 0 (intact) - external boot is allowed 1 (burned) - external boot is disabled
3 SWE	SW enable SW JTAG enable status 1 enabled 0 disabled
2 SWF	Software JTAG enable fuse Status of the no SW disable JTAG fuse 0 (intact) - SW enable possible 1 (intact) - no SW enable possible
1 KTA	Kill Trace is active 1 active 0 not active
0 KTF	Kill Trace Enable fuse value 0 (intact) - kill trace is never active 1 (burned) - kill trace functionality enabled

45.9.7 General Purpose Clocks Control Register (SJC_GPCCR)

This register is used to configure clock related modes in SOC, see System Configuration chapter for more information. Those bits are directly connected to JTAG outputs. Bit 0 of GPCCR controls SDMA clocks invocation. When out of reset, the SDMA is in sleep mode with no SDMA clock running. Unlike events, debug requests does not wake SDMA if it is in sleep mode. The debug request is recognized by the SDMA only when it exits sleep mode upon reception of an event. To be able to enter debug mode even if no event is triggered, the SDMA clock on bit needs to be set prior to sending the debug request (clear at reset).

Address: 0h base + 7h offset = 7h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

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

SJC_GPCCR field descriptions

Field	Description
31–2 -	Reserved
1 ACLKOFFDIS	Disable/prevent ARM platform clock/power shutdown
0 SCLKR	SDMA Clock ON Register - This bit forces the clock on of the SDMA

Chapter 46
Secure Non-Volatile Storage (SNVS)

46.1 SNVS overview

The low-power (battery-backed) section incorporates a secure real time counter, a monotonic counter, and a general purpose register. This portion of the block is powered by a battery that maintains the state of the SNVS_LP registers when the chip is powered off.

Figure 46-1. Example SNVS Connectivity

46.1.1 SNVS features

The following table summarizes the features:

Table 46-1. SNVS feature summary

Feature	What it does
Non-secure Real time counter (HPRTC)	• The counter is driven by a dedicated clock, which is off when the system power is down • Programmable time alarm interrupt • Periodic interrupt can be generated with different frequencies
Monotonic counter	• The monotonic counter state is nonvolatile. • The counter can only increment. • The counter is a non-rollover counter • The counter value is invalidated in case of security violation.
General-purpose register	• The general-purpose register state is nonvolatile.
Register access protection	• Privileged software access policy • Registers can be programmed only when the system security monitor is in functional state. • Some registers/values can only be programmable once per boot cycle.

46.1.2 Modes of operation

The SNVS operates in either the system power-down or system power-up mode of operation.

During system power-down, SNVS_HP is powered-down. SNVS_LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode, SNVS_LP retains the state of its registers .

During system power-up, SNVS_HP and SNVS_LP are both powered-up and all SNVS functions are operational.

46.2 SNVS structure

The SNVS block is divided into two major submodules based on power supply: the high power domain (SNVS_HP) and the low power domain (SNVS_LP). They are powered as follows:

- SNVS_LP - dedicated always-powered-on domain
- SNVS_HP - system (chip) power domain

The following figure illustrates the low power and chip power domains of SNVS.

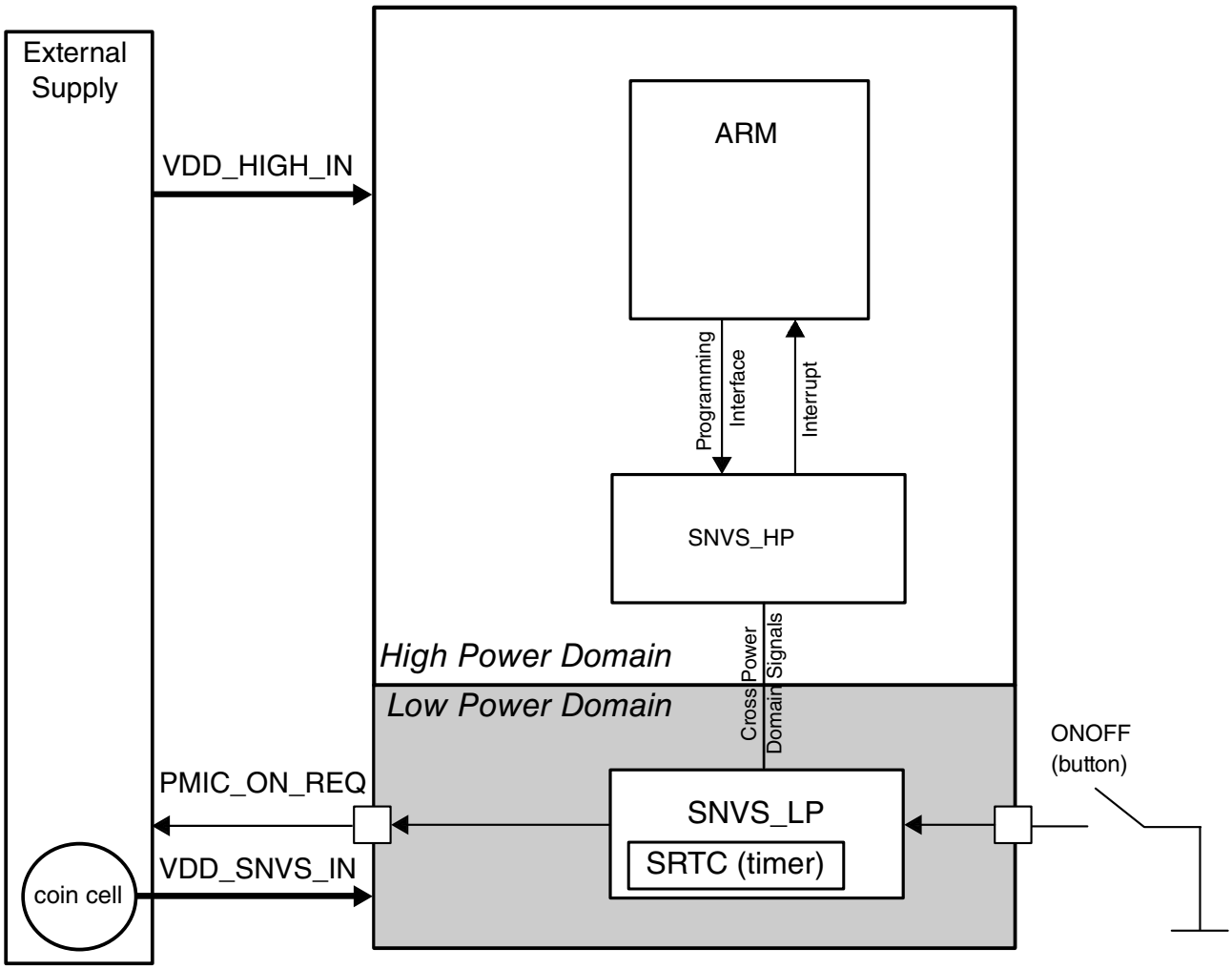


Figure 46-2. SNVS Power Domains

The SNVS_HP section implements all features that enable system communication and provisioning of the SNVS_LP section.

The SNVS_LP section provides hardware that enables secure storage and protection of sensitive data.

46.2.1 SNVS_HP (high power domain)

SNVS_HP is partitioned into the following functional units:

- IP bus interface
- SNVS_LP interface
- Real time counter with alarm
- Control and status registers

SNVS_HP is in the chip's power supply domain and thus receives power along with the rest of the chip. SNVS_HP provides an interface between SNVS_LP and the rest of the system; there is no way to access the SNVS_LP registers except through the SNVS_HP. For access to the SNVS_LP registers, SNVS_HP must be powered up. It uses a register access permission policy to determine whether access to particular registers is permitted.

46.2.2 Non-secure real time counter

SNVS _HP has an autonomous non-secure real time counter. The counter is not active and is reset when the system is powered down. The HP RTC can be used by any application; it has no privileged software access restrictions. The counter can be synchronized with the SNVS_LP SRTC by writing to a specific bit in the SNVS_HP Control Register.

46.2.2.1 Calibrating the time counter

The RTC accuracy may suffer from a drift in the clock, which is used to increment the RTC register. To compensate for this drift, a clock calibration mechanism can adjust the RTC value. It is up to the system processor to decide whether calibration is required or not. If RTC correction is required, enable the mechanism and set the calibration value in the control register. The calibration value is a 5 bit value including the sign bit, which is implemented in 2's complement.

If the calibration mechanism is enabled, the calibration value is added or subtracted from the RTC on a periodic basis, once per 32768 cycles of the RTC clock.

The following table shows the available correction range.

Table 46-2. Time counter calibration settings

Calibration value setting	Correction in counts per 32768 cycles of the counter clock
01111	+15
:	:
00010	+2
00001	+1
00000	0
11111	-1
11110	-2
:	:
10001	-15
10000	-16

46.2.2.2 Time counter alarm

The SNVS _HP non-secure RTC has its own time alarm register. Any application can update this register. The SNVS _HP time alarm can generate interrupts to alert the host processor and can wake up the host processor from one of its low-power modes. Note that this alarm cannot wake up the entire system if it is powered off because this alarm would also be powered off.

46.2.2.3 Periodic interrupt

The SNVS_HP non-secure RTC incorporates a periodic interrupt. The periodic interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the RTC. The periodic interrupt source is chosen from 16 bits of the HP RTC according to the PI_FREQ field setting in the HP Control Register. This bit selection also defines the frequency of the periodic interrupt.

The following figure shows the SNVS_HP RTC and its interrupts.

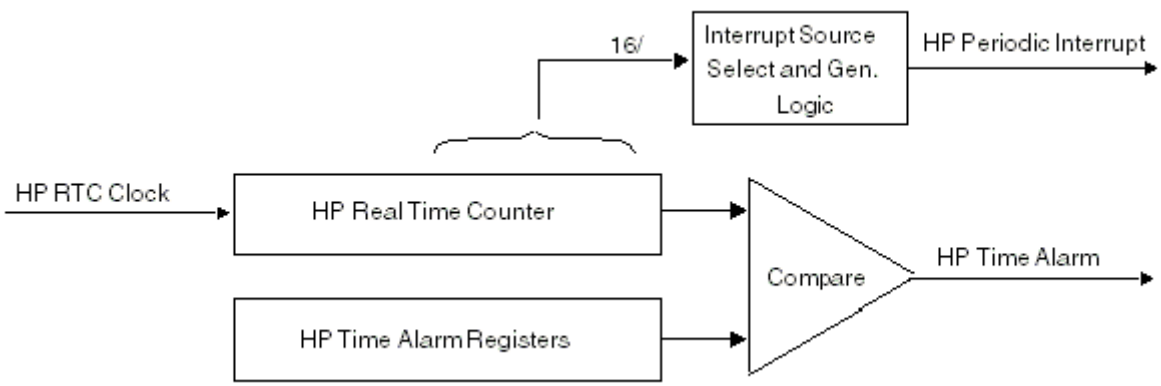


Figure 46-3. SNVS_HP RTC, alarm, and interrupts

46.3 SNVS_LP (low power domain)

SNVS_LP has the following functional units:

- Non-rollover monotonic counter
- General purpose register
- Control and status registers

The SNVS_LP is a data storage subsystem. Its purpose is to store and protect system data, regardless of the main system power state.

SNVS _LP is in the always-powered-up domain, which is a separate power domain with its own power supply.

46.3.1 Behavior during system power down

When the chip power supply domain loses power, SNVS_LP continues to operate normally, and it ignores all inputs from SNVS_HP.

46.3.2 Monotonic counter (MC)

The following figure shows the MC and its rollover security violation.

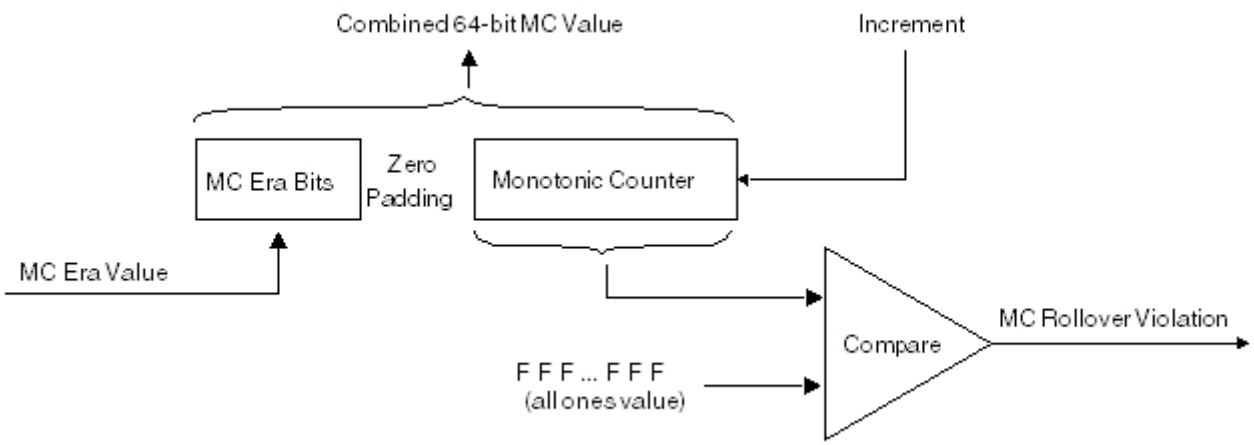


Figure 46-4. SNVS _LP monotonic counter

Some security applications require a monotonic counter (MC) that cannot be exhausted or returned to any previous value during the product's lifetime. Because the MC can never repeat a number, it cannot be reset or cycled back to its starting count. If it reaches its maximum value, it does not rollover. Instead, a monotonic counter rollover indication is generated to the SNVS _LP tamper monitor. This generates an interrupt to the host processor.

The SNVS uses an ERA value derived from the OTP elements as a mechanism for recovery from an MC failure (for example, due to a failure of LP power) where the MC value was compromised or cleared. The ERA value is prepended to the MC to form its

most significant bits. Once any of the ERA value bits are set, the MC can count up from any value, including zero. This guarantees that any future value of the combined monotonic counter will be greater than any of its past values.

46.4 SNVS reset and system power up

This table describes reset actions for SNVS.

Table 46-3. Reset summary

Reset	Source	Characteristics	Internally resets
HP Hard	ipg_hard_async_reset_b	active-low, asynchronous	All SNVS_HPSNVS_LP registers and flops.
LP Power On Reset (POR)	lp_por_b	active-low, asynchronous	All SNVS_LP registers and flops
LP software Reset	software	active-high, synchronous, 1 cycle	All SNVS_LP registers and flops. LP software Reset can be asserted if not disabled.

46.4.1 PMIC Interface

The On/Off logic inside of SNVS_LP allows for connecting directly to a PMIC or other voltage regulator device. The logic takes a button input signal and then outputs a PMIC "ON" Request and a "Power Off" Interrupt. PMIC logic also supports the SNVS_LP tamper logic which will allow waking the system up when a tamper event has happened while in the OFF state. The logic has two different modes of operation (Dumb and Smart mode).

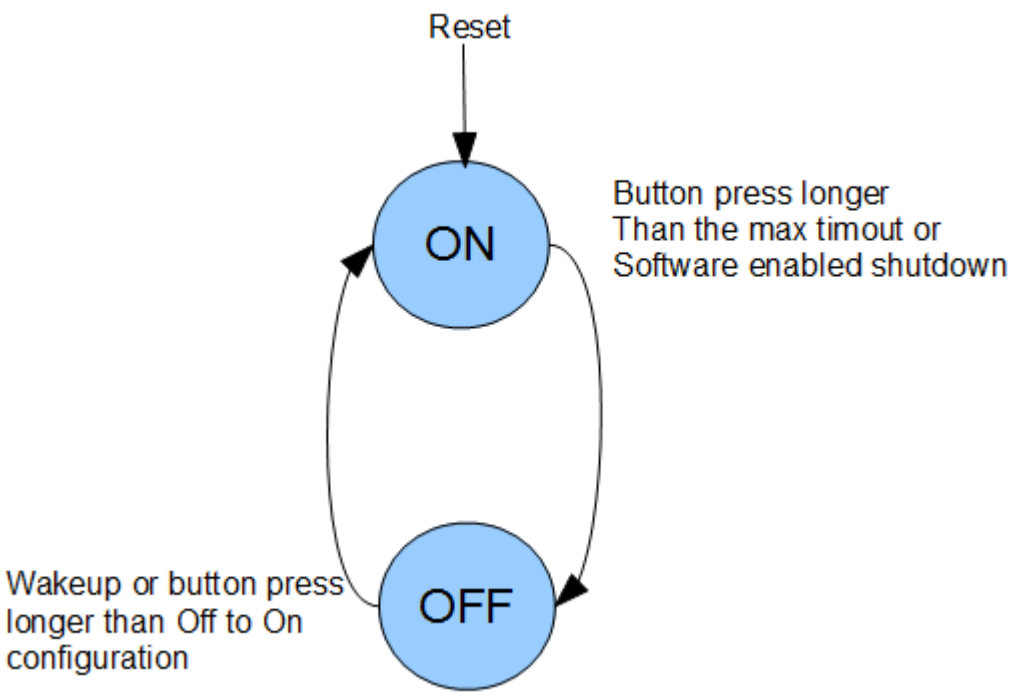
Dumb PMIC Mode:

The Dumb PMIC Mode uses PMIC "ON" Request to issue a level signal for on and off. Dumb pmic mode has many different configuration options which include (debounce, off to on time, and max time out).

- Debounce: The debounce configuration supports 0 msec, 50 msec, 100 msec and 500 msec. The debounce is used to generate the set_pwr_off_irq interrupt. While in the ON state and the button is pressed longer than the debounce time the set_pwr_off_irq is generated.

- Off to On Time: The Off to On configuration supports 0 msec, 50 msec, 100 msec, and 500 msec. This configuration supports the time it takes to request power on after the configured button press time has been reached. Once the button is pressed longer than the configuration time, the state machine will transition from the OFF to the ON state.
- Max Timeout: The max timeout configuration supports 5 secs, 10 secs, 15 secs and disable. This configuration supports the time it takes to request power down after the button has been pressed for the defined time.

The dumb PMIC mode uses a 2 state state machine, as shown below. The output of the pmic_en_b is generated by the state of the state machine.



Smart PMIC Mode:

The smart PMIC mode is meant to connect to another PMIC. The PMIC "ON" Request signal issues a pulse instead of a level signal. The only configuration option available for this mode is the Debounce configuration that is used for the "Power Off" Interrupt.

46.5 SNVS interrupts and alarms

SNVS provides the following interrupt and alarm lines:

- Functional interrupt (active low)

- Real-time clock period interrupt
- Power off (button) interrupt

The following table summarizes all SNVS interrupts and alarm sources.

Table 46-4. Interrupts and alarms summary

Interrupt	Source	Default configuration[1]	Configuration options
SNVS functional interrupt	RTC time alarm	Disable	Enable/Disable
	RTC periodic interrupt	Disable	Enable/Disable
SNVS power off (button) interrupt	BTN input signal	50msec debounce	Debounce time

1. Default behavior refers to the setting after LP/HP reset.

46.6 Programming Guidelines

This section provides initialization and application information for the SNVS module.

46.6.1 RTC control bits setting

All SNVS registers are programmed from the register bus. Therefore, any changes are synchronized with the IP clock. Several registers can also change synchronously with the RTC clock after they are programmed. To avoid IP clock and RTC clock synchronization issues, these values can only be programmed when the corresponding function is disabled. The following table presents the list of these values with the control bit setting required for programming.

Table 46-5. RTC synchronized values list

Function	Value/register	Control bit setting
HP section		
HP Real Time Counter	HPRTCMR and HPRTCLR Registers	RTC_EN = 0 - HPRTCMR/HPRTCLR can be programmed RTC_EN = 1 - HPRTCMR/HPRTCLR cannot be programmed
HP Time Alarm	HPTAMR and HPTALR Registers	HPTA_EN = 0 - HPTAMR/HPTALR can be programmed HPTA_EN = 1 - HPTAMR/HPTALR cannot be programmed
HP Time Calibration Value	HPCALB_VAL Value	HPCALB_EN = 0 - HPCALB_VAL can be programmed HPCALB_EN = 1 - HPCALB_VAL cannot be programmed

Use the following step to program synchronized values:

1. Check the enable bit value. If set, clear it.
2. Verify that the enable bit is cleared.

   There are two reasons to verify the enable bit's setting:

   - Enable bit clearing does not happen immediately; it takes three IPclock cycles and two RTC clock cycles to change the enable bit's value.
   - If the enable bit is locked for programming, it cannot be cleared.
3. Program the desired value.
4. Set the enable bit; it takes three IP clock cycles and two RTC clock cycles for the bit to set.

NOTE

Incrementing the value programmed into RTC registers by two compensates for the two RTC clock cycle delay that is required to enable the counter.

46.6.2 RTC value read

There are two scenarios when software can read corrupted values from the RTC (HPRTCMR and HPRTCLR) registers:

- The RTC counters are incremented by the slow 32 kHz clock, which is asynchronous to the system clock. The counter value is synchronized to the system clock before software reads that. The synchronization register may capture the counter value in the middle of the counter update. In this case, it is not guaranteed that all bits are properly sampled by the synchronization register; the value read by software can be wrong.
- The RTC value is longer than the single bus read transaction of 32-bits. Therefore, software reads two registers, each holding a portion of the counter value. After reading one of these registers but before reading the second register, both registers may update their values. In this case, the value combined by software will be incorrect.

To avoid these issues, it is strongly recommended that software perform two consecutive reads of the RTC value:

- If two consecutive reads are similar, the value is correct.
- If two consecutive reads are different, perform two more reads.

The worst case scenario may require three sessions of two consecutive reads.

46.6.3 General initialization guidelines

Complete the following steps in order to properly initialize the module:

1. Enable interrupts in SNVScontrol and configuration registers.
2. Program SNVS general functions/configurations.
3. User Specific: Set lock bits.

46.7 SNVS Register Descriptions

This section contains detailed register descriptions for the SNVS registers. Each description includes a standard register diagram and register table. The register table provides detailed descriptions of the register bit and field functions, in bit order.

SNVS registers consist of two types:

- Privileged read/write accessible
- Non-privileged read/write accessible

Privileged read/write accessible registers can only be accessed for read/write by privileged software. Unauthorized write accesses are ignored, and unauthorized read accesses return zero. Non-privileged software can access privileged access registers when the non-privileged software access enable bit is set in the SNVS_HP Command Register.

In addition, all privileged access registers (except HPSVSR and LPSR) can be written to only if the system security monitor is in one of the three functional states:

- Non-Secure
- Trusted
- Secure

Certain fields in the SNVS_HP Command Register can be written in non-functional system security monitor states (init, check, soft fail, and hard fail) as follows:

- SW_LPSV, SSM_ST, SW_SV, and SW_FSV can be written in check or soft fail state.
- The HAC_STOP bit can only be set in soft fail state but can be cleared in either soft fail or a functional state.
- HAC_LOAD, HAC_CLEAR bits and HPSVSR, LPSR Registers can be accessed in soft fail or a functional state.

The system security monitor state does not restrict read access to SNVS registers.

Non-privileged read/write accessible registers are read/write accessible by any software.

The LP register values are set only on LP POR and are unaffected by System (HP) POR. The HP registers are set only on System POR and are unaffected by LP POR.

The following table shows the SNVS main memory map.

46.7.1 SNVS Memory Map

Offset	Register	Width (In bits)	Access	Reset value
0h	SNVS_HP Lock (HPLR)	32	RW	00000000h
4h	SNVS_HP Command (HPCOMR)	32	RW	00000000h
8h	SNVS_HP Control (HPCR)	32	RW	00000000h
Ch	SNVS_HP Security Interrupt Control (HPSICR)	32	RW	00000000h
10h	SNVS_HP Security Violation Control (HPSVCR)	32	RW	00000000h
14h	SNVS_HP Status (HPSR)	32	W1C	80000000h
18h	SNVS_HP Security Violation Status (HPSVSR)	32	W1C	00000000h
1Ch	SNVS_HP High Assurance Counter IV (HPHACIVR)	32	RW	00000000h
20h	SNVS_HP High Assurance Counter (HPHACR)	32	RO	00000000h
24h	SNVS_HP Real Time Counter MSB (HPRTCMR)	32	RW	00000000h
28h	SNVS_HP Real Time Counter LSB (HPRTCLR)	32	RW	00000000h
2Ch	SNVS_HP Time Alarm MSB (HPTAMR)	32	RW	00000000h
30h	SNVS_HP Time Alarm LSB (HPTALR)	32	RW	00000000h
34h	SNVS_LP Lock (LPLR)	32	RW	00000000h
38h	SNVS_LP Control (LPCR)	32	RW	00000000h
3Ch	SNVS_LP Master Key Control (LPMKCR)	32	RW	00000000h
40h	SNVS_LP Security Violation Control (LPSVCR)	32	RW	00000000h
44h	SNVS_LP Tamper Glitch Filters Configuration (LPTGFCR)	32	RW	00000000h
48h	SNVS_LP Tamper Detectors Configuration (LPTDCR)	32	RW	00000000h
4Ch	SNVS_LP Status (LPSR)	32	W1C	00000008h
50h	SNVS_LP Secure Real Time Counter MSB (LPSRTCMR)	32	RW	00000000h
54h	SNVS_LP Secure Real Time Counter LSB (LPSRTCLR)	32	RW	00000000h
58h	SNVS_LP Time Alarm (LPTAR)	32	RW	00000000h
5Ch	SNVS_LP Secure Monotonic Counter MSB (LPSMCMR)	32	RO	00000000h
60h	SNVS_LP Secure Monotonic Counter LSB (LPSMCLR)	32	RO	00000000h
64h	SNVS_LP Power Glitch Detector (LPPGDR)	32	RW	00000000h
68h	SNVS_LP General Purpose 0 (alias) (LPGPR0_alias)	32	RW	00000000h
6Ch - 8Bh	SNVS_LP Zeroizable Master Key (LPZMKR0 - LPZMKR31)	8	RW	00h
90h - 9Ch	SNVS_LP General Purposes 0 .. 3 (LPGPR0_30 - LPGPR0_33)	32	RW	00000000h
A0h	SNVS_LP Tamper Detectors Config 2 (LPTDC2R)	32	RW	00000000h
A4h	SNVS_LP Tamper Detectors Status (LPTDSR)	32	W1C	00000000h
A8h	SNVS_LP Tamper Glitch Filter 1 Configuration (LPTGF1CR)	32	RW	00000000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
ACh	SNVS_LP Tamper Glitch Filter 2 Configuration (LPTGF2CR)	32	RW	00000000h
C0h	SNVS_LP Active Tamper 1 Configuration (LPAT1CR)	32	WO	00000000h
C4h	SNVS_LP Active Tamper 2 Configuration (LPAT2CR)	32	WO	00000000h
C8h	SNVS_LP Active Tamper 3 Configuration (LPAT3CR)	32	WO	00000000h
CCh	SNVS_LP Active Tamper 4 Configuration (LPAT4CR)	32	WO	00000000h
D0h	SNVS_LP Active Tamper 5 Configuration (LPAT5CR)	32	WO	00000000h
E0h	SNVS_LP Active Tamper Control (LPATCTLR)	32	RW	00000000h
E4h	SNVS_LP Active Tamper Clock Control (LPATCLKR)	32	RW	00000000h
E8h	SNVS_LP Active Tamper Routing Control 1 (LPATRC1R)	32	RW	00000000h
ECh	SNVS_LP Active Tamper Routing Control 2 (LPATRC2R)	32	RW	00000000h
BF8h	SNVS_HP Version ID 1 (HPVIDR1)	32	RO	003E0103h
BFCh	SNVS_HP Version ID 2 (HPVIDR2)	32	RO	05000000h

46.7.2 SNVS_HP Lock (HPLR)

46.7.2.1 Address

Register	Offset
HPLR	0h

46.7.2.2 Function

The SNVS_HP Lock Register contains lock bits for the SNVS registers. This is a privileged write register.

46.7.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			AT5_SL	AT4_SL	AT3_SL	AT2_SL	AT1_SL	Reserved					HAC_L	HPSICR_L	HPSVCR_L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						MKS_SL	LPTDCR_SL	LPTGFCR_SL	LPSVCR_SL	GPR_SL	MC_SL	LPCALB_SL	SRTC_SL	ZMK_RSL	ZMK_WSL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.2.4 Fields

Field	Function
31-29 —	
28 AT5_SL	Active Tamper 5 Soft Lock When set, prevents any writes to the Active Tamper 5 registers. Once set, this bit can only be reset by the system reset 0b - 1b -
27 AT4_SL	Active Tamper 4 Soft Lock When set, prevents any writes to the Active Tamper 4 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
26 AT3_SL	Active Tamper 3 Soft Lock When set, prevents any writes to the Active Tamper 3 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
25 AT2_SL	Active Tamper 2 Soft Lock When set, prevents any writes to the Active Tamper 2 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
24	

Table continues on the next page...

Field	Function
AT1_SL	Active Tamper 1 Soft Lock When set, prevents any writes to the Active Tamper 1 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
23-19 —	
18 HAC_L	High Assurance Configuration Lock When set, prevents any writes to HPHACIVR, HPHACR, and HAC_EN bit of HPCOMR. Once set, this bit can only be reset by the system reset. 0b - 1b -
17 HPSICR_L	HP Security Interrupt Control Register Lock When set, prevents any writes to the HPSICR. Once set, this bit can only be reset by the system reset. 0b - 1b -
16 HPSVCR_L	HP Security Violation Control Register Lock When set, prevents any writes to the HPSVCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
15-10 —	
9 MKS_SL	Master Key Select Soft Lock When set, prevents any writes to the MASTER_KEY_SEL field of the LPMKCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
8 LPTDCR_SL	LP Tamper Detectors Configuration Register Soft Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
7 LPTGFCR_SL	LP Tamper Glitch Filter Configuration Register Soft Lock When set, prevents any writes to the LPTGFCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
6 LPSVCR_SL	LP Security Violation Control Register Soft Lock When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
5	General Purpose Register Soft Lock

Table continues on the next page...

Field	Function
GPR_SL	When set, prevents any writes to the GPR. Once set, this bit can only be reset by the system reset. 0b - 1b -
4 MC_SL	Monotonic Counter Soft Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the system reset. 0b - 1b -
3 LPCALB_SL	LP Calibration Soft Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the system reset. 0b - 1b -
2 SRTC_SL	Secure Real Time Counter Soft Lock When set, prevents any writes to the SRTC Registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the system reset. 0b - 1b -
1 ZMK_RSL	Zeroizable Master Key Read Soft Lock When set, prevents any software reads to the ZMK Registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), the ZMK and ZMK_ECC_VALUE cannot be read by software. Regardless of the bit setting, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the system reset. 0b - 1b -
0 ZMK_WSL	Zeroizable Master Key Write Soft Lock When set, prevents any writes (software and hardware) to the ZMK registers and MASTER_KEY_SEL, ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be cleared by system reset. 0b - 1b -

46.7.3 SNVS_HP Command (HPCOMR)

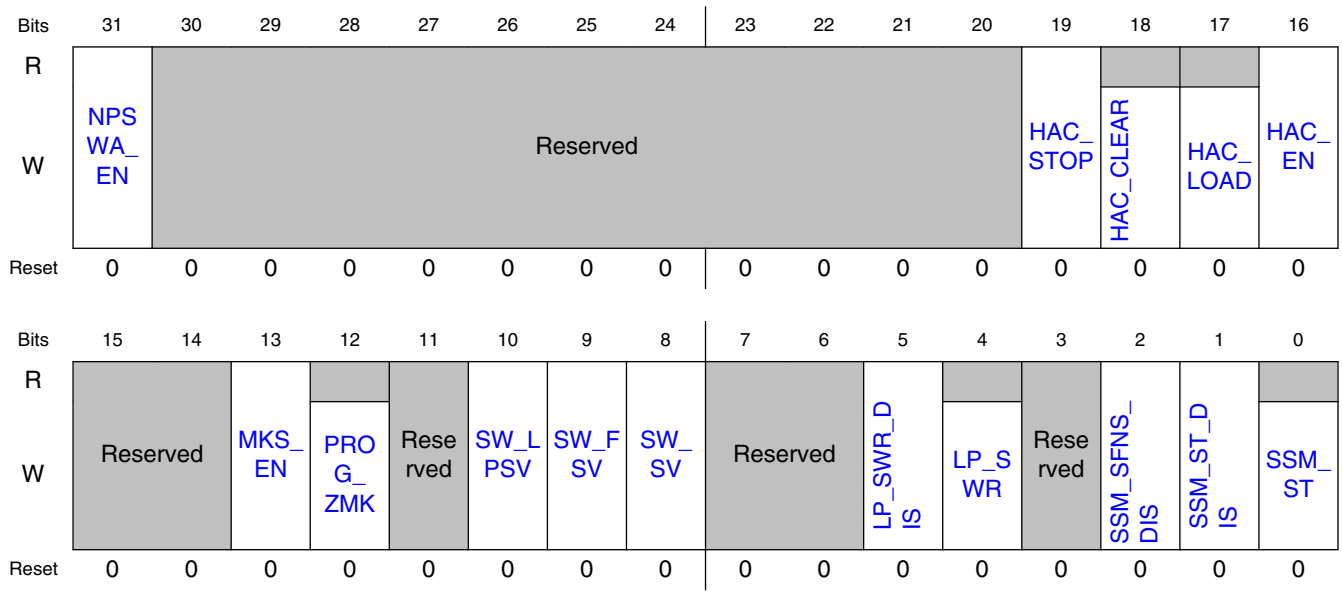
46.7.3.1 Address

Register	Offset
HPCOMR	4h

46.7.3.2 Function

The SNVS_HP Command Register contains the command, configuration, and control bits for the SNVS block. Some fields of this register can be written to in check and soft fail states in addition to the standard write access in functional states. This is a privileged write register.

46.7.3.3 Diagram



46.7.3.4 Fields

Field	Function
31 NPSWA_EN	Non-Privileged Software Access Enable When set, allows non-privileged software to access all SNVS registers, including those that are privileged software read/write access only. 0 Only privileged software can access privileged registers 1 Any software can access privileged registers
30-20 —	
19 HAC_STOP	High Assurance Counter Stop

Table continues on the next page...

Field	Function
	This bit can be set only when SSM is in soft fail state. When set, it stops the high assurance counter and prevents transition to the hard fail state. This bit can be cleared in a functional or soft fail state. If the bit is cleared in the soft fail state, the high assurance counter counts down from the place where it was stopped. 0 HAC counter can count down 1 HAC counter is stopped
18 HAC_CLEAR	High Assurance Counter Clear When set, it clears the High Assurance Counter Register. It can be cleared in a functional or soft fail state. If the HAC counter is cleared in the soft fail state, the SSM transitions to the hard fail state if high assurance configuration is enabled (HAC_EN is set). This self-clearing bit is always read as zero. 0b - 1b -
17 HAC_LOAD	High Assurance Counter Load When set, it loads the High Assurance Counter Register with the value of the High Assurance Counter Load Register. It can be done in a functional or soft fail state. This self-clearing bit is always read as zero. 0b - 1b -
16 HAC_EN	High Assurance Configuration Enable This bit controls the SSM transition from the soft fail to the hard fail state. When this bit is set and software fails to stop the HAC before it expires, the SSM transitions to the hard fail state. This bit cannot be changed once HAC_L bit is set. 0b - 1b -
15-14 —	
13 MKS_EN	Master Key Select Enable When not set, the one time programmable (OTP) master key is selected by default. When set, the master key is selected according to the setting of the master key select field (MASTER_KEY_SEL) of LPMKCR. Once set, this bit can only be reset by the system reset. 0b - OTP master key is selected as an SNVS master key 1b - SNVS master key is selected according to the setting of the MASTER_KEY_SEL field of LPMKCR
12 PROG_ZMK	Program Zeroizable Master Key This bit activates ZMK hardware programming mechanism. This mechanism is activated only if the ZMK is configured to the hardware programming mode and ZMK in not locked for writes. This self-clearing bit is always read as zero. 0b - 1b -
11 —	
10 SW_LPSV	LP Software Security Violation When set, SNVS_LP treats this bit as a security violation. The LP secure data is zeroized or invalidated according to the configuration. This security violation may result in a system security monitor transition if the LP Security Violation is enabled in the SNVS_HP Security Violation Control Register.

Table continues on the next page...

Field	Function
9 SW_FSV	Software Fatal Security Violation When set, the system security monitor treats this bit as a fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check State -> Soft Fail Non-Secure State -> Soft Fail Trusted State -> Soft Fail Secure State -> Soft Fail
8 SW_SV	Software Security Violation When set, the system security monitor treats this bit as a non-fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check -> Non-Secure Trusted -> Soft Fail Secure -> Soft Fail
7-6 —	
5 LP_SWR_DIS	LP Software Reset Disable When set, disables the LP software reset. Once set, this bit can only be reset by the system reset. 0b - 1b -
4 LP_SWR	LP Software Reset When set, it resets the SNVS_LP section. This bit cannot be set when the LP_SWR_DIS bit is set. This self-clearing bit is always read as zero. 0b - 1b -
3 —	
2 SSM_SFNS_DIS	SSM Soft Fail to Non-Secure State Transition Disable When set, it disables the SSM transition from soft fail to non-secure state. Once set after the reset this bit cannot be changed 0b - 1b -
1 SSM_ST_DIS	SSM Secure to Trusted State Transition Disable When set, disables the SSM transition from secure to trusted state. Once set after the reset, this bit cannot be changed. 0b - 1b -
0 SSM_ST	SSM State Transition Transition state of the system security monitor. This self-clearing bit is always read as zero. This command results only in the following transitions of the SSM:

Field	Function
	Check State -> Non-Secure (when Non-Secure Boot and not in Fab Configuration) Check State -> Trusted (when Secure Boot or in Fab Configuration) Trusted State -> Secure Secure State -> Trusted (if not disabled by SSM_ST_DIS bit) Soft Fail -> Non-Secure (if not disabled by SSM_SFNS_DIS bit)

46.7.4 SNVS_HP Control (HPCR)

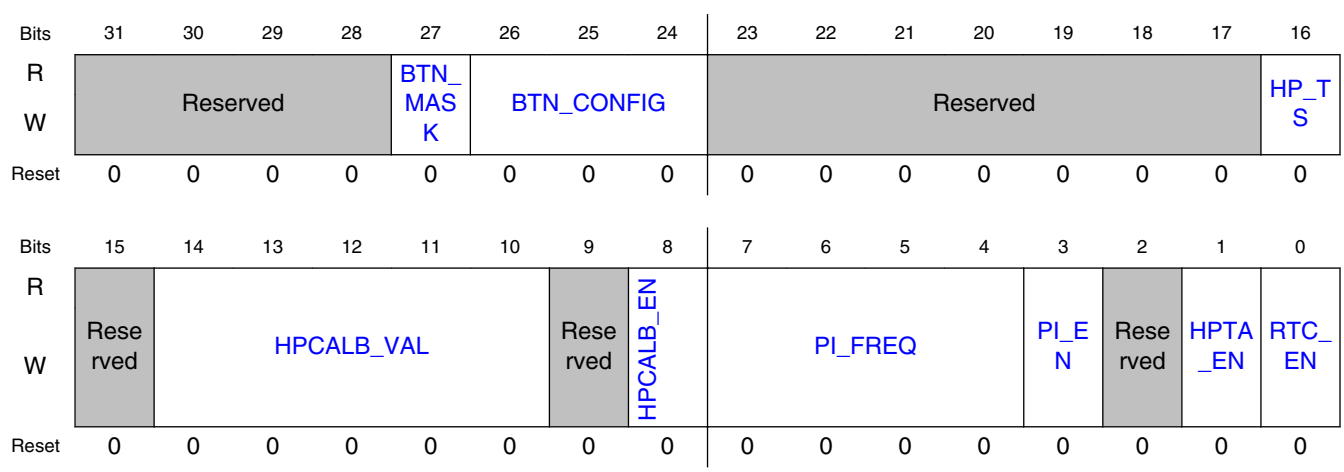
46.7.4.1 Address

Register	Offset
HPCR	8h

46.7.4.2 Function

The SNVS_HP Control Register contains various control bits of the HP section of SNVS.

46.7.4.3 Diagram



46.7.4.4 Fields

Field	Function
31-28 —	
27 BTN_MASK	Button interrupt mask. This bit is used to mask the ipi_snvs_btn_int_b (button) interrupt request. 0: Interrupt disabled 1: Interrupt enabled
26-24 BTN_CONFIG	Button Configuration. This field is used to configure which feature of the button (BTN) input signal constitutes "active". 000: Button signal is active low 001: Button signal is active high 010: Button signal is active on the rising edge 011: Button signal is active on the falling edge 100: Button signal is active on any edge All other patterns are reserved.
23-17 —	
16 HP_TS	HP Time Synchronize When set, this updates the HP Real Time Counter with the LP Real Time Counter value. This self-clearing bit is always read as zero. 0b - 1b -
15 —	
14-10 HPCALB_VAL	HP Calibration Value Defines signed calibration value for the HP Real Time Counter. This field can be programmed only when RTC Calibration is disabled (HPCALB_EN is not set). This is a 5-bit 2's complement value, hence the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter. 00000b - 00001b - 00010b - 01111b - 10000b - 10001b - 11110b - 11111b -
9 —	
8 HPCALB_EN	HP Real Time Counter Calibration Enabled

Table continues on the next page...

Field	Function
	Indicates that the time calibration mechanism is enabled. 0b - 1b -
7-4 PI_FREQ	Periodic Interrupt Frequency Defines frequency of the periodic interrupt. The interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the HP Real Time Counter and Real Time Counter and Periodic Interrupt are both enabled (RTC_EN and PI_EN are set). It is recommended to program this field when Periodic Interrupt is disabled (PI_EN is not set). The possible frequencies are: 0000b - 0001b - 0010b - 0011b - 0100b - 0101b - 0110b - 0111b - 1000b - 1001b - 1010b - 1011b - 1100b - 1101b - 1110b - 1111b -
3 PI_EN	HP Periodic Interrupt Enable The periodic interrupt can be generated only if the HP Real Time Counter is enabled. 0b - 1b -
2 —	
1 HPTA_EN	HP Time Alarm Enable When set, the time alarm interrupt is generated if the value in the HP Time Alarm Registers is equal to the value of the HP Real Time Counter. 0b - 1b -
0 RTC_EN	HP Real Time Counter Enable 0b - 1b -

46.7.5 SNVS_HP Security Interrupt Control (HPSICR)

46.7.5.1 Address

Register	Offset
HPSICR	Ch

46.7.5.2 Function

The HP Security Interrupt Control Register defines the SNVS security interrupt generation policy. This is a privileged write register.

46.7.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LPSVI_EN	Reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										SVI_EN5	SVI_EN4	SVI_EN3	SVI_EN2	SVI_EN1	SVI_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.5.4 Fields

Field	Function
31 LPSVI_EN	LP Security Violation Interrupt Enable This bit enables generating of the security interrupt to the host processor upon security violation signal from the LP section. 0b - 1b -
30-6 —	Reserved
5 SVI_EN5	Security Violation Interrupt 5 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 5. 0b -

Table continues on the next page...

Field	Function
	1b -
4 SVI_EN4	Security Violation Interrupt 4 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 4. 0b - 1b -
3 SVI_EN3	Security Violation Interrupt 3 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 3. 0b - 1b -
2 SVI_EN2	Security Violation Interrupt 2 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 2. 0b - 1b -
1 SVI_EN1	Security Violation Interrupt 1 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 1. 0b - 1b -
0 SVI_EN0	Security Violation Interrupt 0 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 0. 0b - 1b -

46.7.6 SNVS_HP Security Violation Control (HPSVCR)

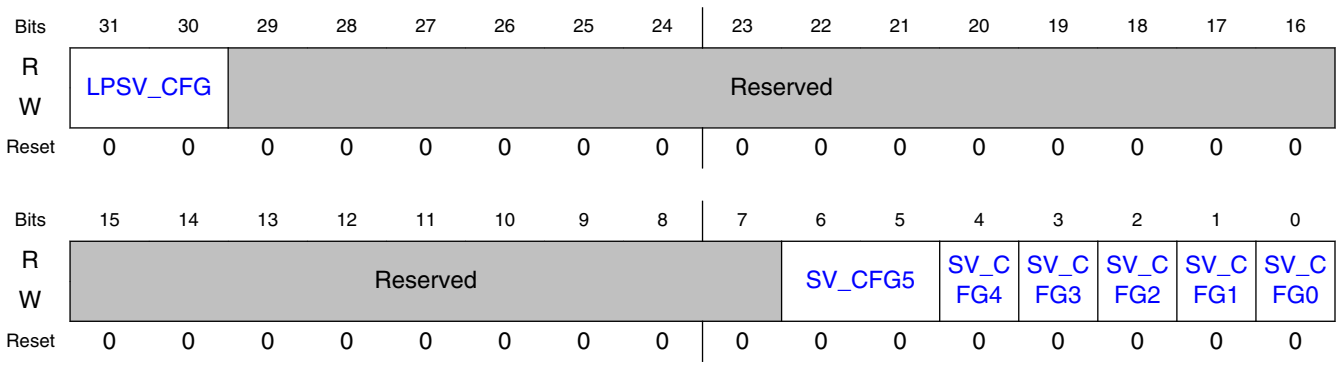
46.7.6.1 Address

Register	Offset
HPSVCR	10h

46.7.6.2 Function

The HP Security Violation Control Register defines types for each security violation input. This is a privileged write register.

46.7.6.3 Diagram



46.7.6.4 Fields

Field	Function
31-30 LPSV_CFG	LP Security Violation Configuration This field configures the LP security violation source. 00b - 01b - 1xb -
29-7 —	
6-5 SV_CFG5	Security Violation Input 5 Configuration This field configures the Security Violation Input 5. This setting instructs the SSM how to respond when a security violation on port 5 is detected. 00b - 01b - 1xb -
4 SV_CFG4	Security Violation Input 4 Configuration This field configures the security violation Input 4. This setting instructs the SSM how to respond when a security violation on port 4 is detected. 0b - 1b -
3	Security Violation Input 3 Configuration

Table continues on the next page...

Field	Function
SV_CFG3	This field configures the security violation input 3. This setting instructs the SSM how to respond when a security violation on port 3 is detected. 0b - 1b -
2 SV_CFG2	Security Violation Input 2 Configuration This field configures the security violation input 2. This setting instructs the SSM how to respond when a security violation on port 2 is detected. 0b - 1b -
1 SV_CFG1	Security Violation Input 1 Configuration This field configures the Security Violation Input 1. This setting instructs the SSM how to respond when a security violation on port 1 is detected. 0b - 1b -
0 SV_CFG0	Security Violation Input 0 Configuration This field configures the security violation input 0. This setting instructs the SSM how to respond when a security violation on port 0 is detected. 0b - 1b -

46.7.7 SNVS_HP Status (HPSR)

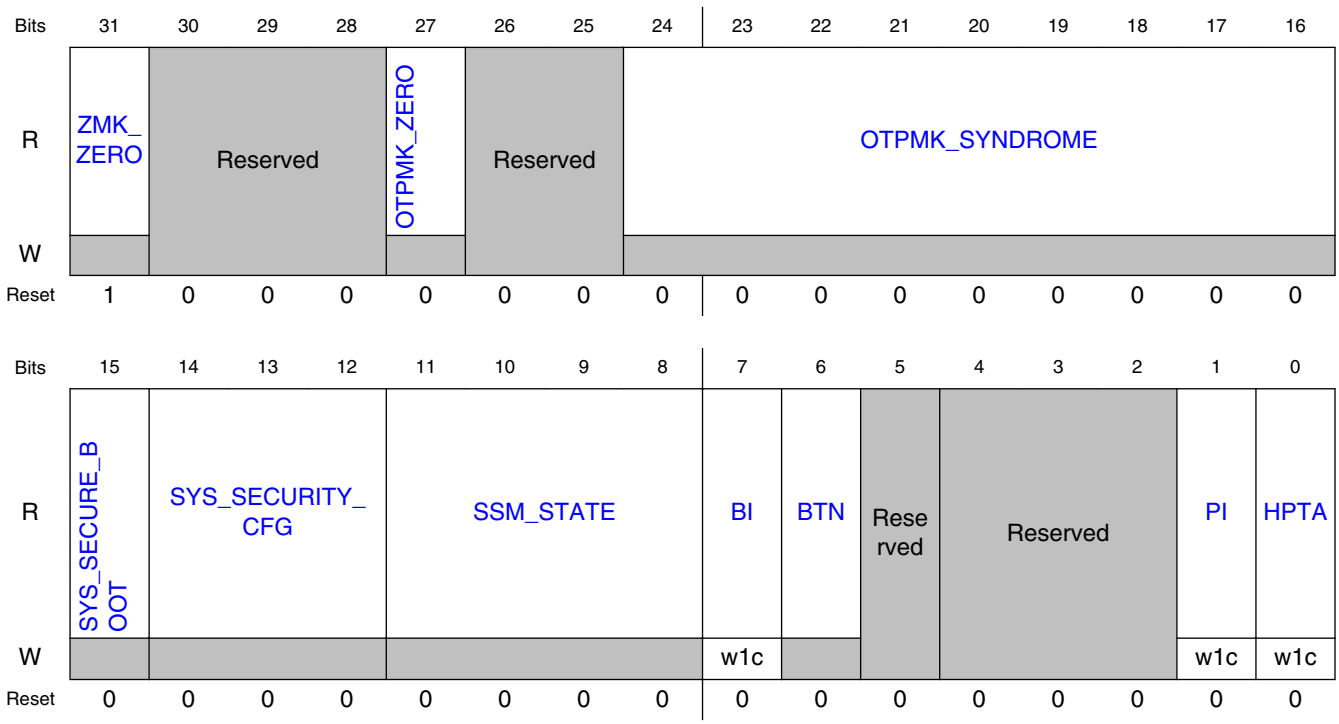
46.7.7.1 Address

Register	Offset
HPSR	14h

46.7.7.2 Function

The HP Status Register reflects the internal state of the SNVS.

46.7.7.3 Diagram



46.7.7.4 Fields

Field	Function
31 ZMK_ZERO	Zeroizable Master Key is Equal to Zero. When set, this bit triggers "bad key" violation if the ZMK is selected for use. 0b - 1b -
30-28 —	
27 OTPMK_ZERO	One Time Programmable Master Key is Equal to Zero. When set, this bit always triggers "bad key" violation. 0b - 1b -
26-25 —	
24-16 OTPMK_SYNDROME	One Time Programmable Master Key Syndrome In the case of a single-bit error, the eight lower bits of this value indicate the bit number of error location. For example, syndrome word 10010110 indicates that key bit 150 (lsb=0) has an error.

Table continues on the next page...

Field	Function
	The ninth bit of the syndrome word checks parity of the whole key value. This bit is 1 when an odd number of errors has occurred and it is 0 when the number of errors is even. For example, if one of the eight bits indicates a failure and the ninth bit is zero then the number of errors in the one time programmable master key is at least 2 and it cannot be corrected. When one of the syndrome bits is set, the bad key violation is always generated.
15 SYS_SECURE_BOOT	System Secure Boot This bit reflects the value of the sys_secure_boot input signal to SNVS. If this bit is 1, the chip boots from internal ROM.
14-12 SYS_SECURITY_CFG	System Security Configuration This field reflects the value of the sys_security_cfg input signal, which is defined as follows: 000b - 001b - 01xb - 1xxb -
11-8 SSM_STATE	System Security Monitor State This field contains the encoded state of the SSM's state machine. The encoding of the possible states are: 0000b - 0001b - 0011b - 1000b - 1001b - 1011b - 1101b - 1111b -
7 BI	Button Interrupt Signal ipi_snvs_btn_int_b was asserted.
6 BTN	Button Value of the BTN input. This is the external button used for PMIC control. 0: BTN not pressed 1: BTN pressed
5 —	
4-2 —	
1 PI	Periodic Interrupt Indicates that periodic interrupt has occurred since this bit was last cleared. 0b - 1b -
0 HPTA	HP Time Alarm Indicates that the HP Time Alarm has occurred since this bit was last cleared. 0b - 1b -

46.7.8 SNVS_HP Security Violation Status (HPSVSR)

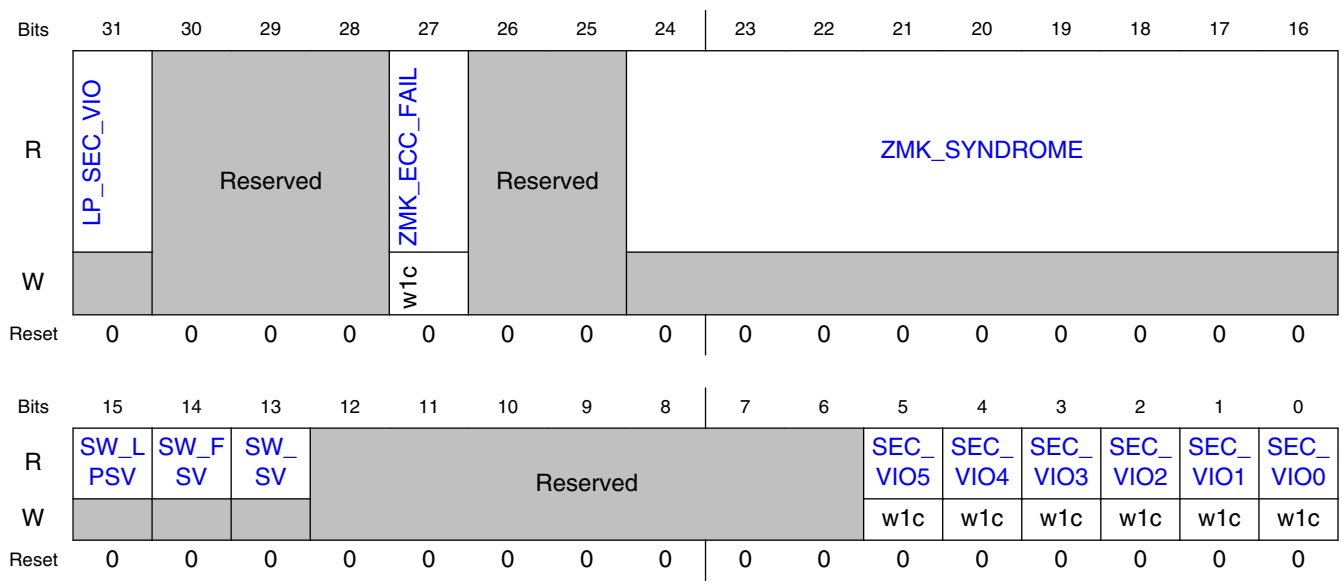
46.7.8.1 Address

Register	Offset
HPSVSR	18h

46.7.8.2 Function

The HP Security Violation Status Register reflects the HP domain security violation records. Write a 1 to SEC_VIO5-0 to clear the corresponding security violation detection flag. Note that this does not automatically clear the security violation signal that is connected to the input, so the security violation may immediately be detected again

46.7.8.3 Diagram



46.7.8.4 Fields

Field	Function
31 LP_SEC_VIO	LP Security Violation A security volation was detected in the SNVS low power section.
30-28 —	
27 ZMK_ECC_FAIL	Zeroizable Master Key Error Correcting Code Check Failure When set, this bit triggers a bad key violation to the SSM and a security violation to the SNVS_LP section, which clears security sensitive data. Writing a one to this bit clears the record of this failure and also clears this register's ZMK_SYNDROME field. 0b - 1b -
26-25 —	
24-16 ZMK_SYNDROME	Zeroizable Master Key Syndrome The ZMK syndrome indicates the single-bit error location and parity for the ZMK regsiter. It operates similar to the OTPMK syndrome. This value is set and locked when a ZMK ECC failure is detected. It is cleared by writing one into ZMK_ECC_FAIL bit.
15 SW_LPSV	LP Software Security Violation This bit is a read-only copy of the SW_LPSV bit in the HP Command Register.
14 SW_FSV	Software Fatal Security Violation This bit is a read-only copy of the SW_FSV bit in the HP Command Register.
13 SW_SV	Software Security Violation This bit is a read-only copy of the SW_SV bit in the HP Command Register.
12-6 —	
5 SEC_VIO5	Security volation on input 5 was detected. 0b - 1b -
4 SEC_VIO4	Security violation on input 4 was detected. 0b - 1b -
3 SEC_VIO3	Security violation on input 3 was detected. 0b - 1b -
2 SEC_VIO2	Security violation on input 2 was detected. 0b - 1b -
1 SEC_VIO1	Security violation on input 1 was detected. 0b -

Table continues on the next page...

Field	Function
	1b -
0 SEC_VIO0	Security violation on input 0 was detected. 0b - 1b -

46.7.9 SNVS_HP High Assurance Counter IV (HPHACIVR)

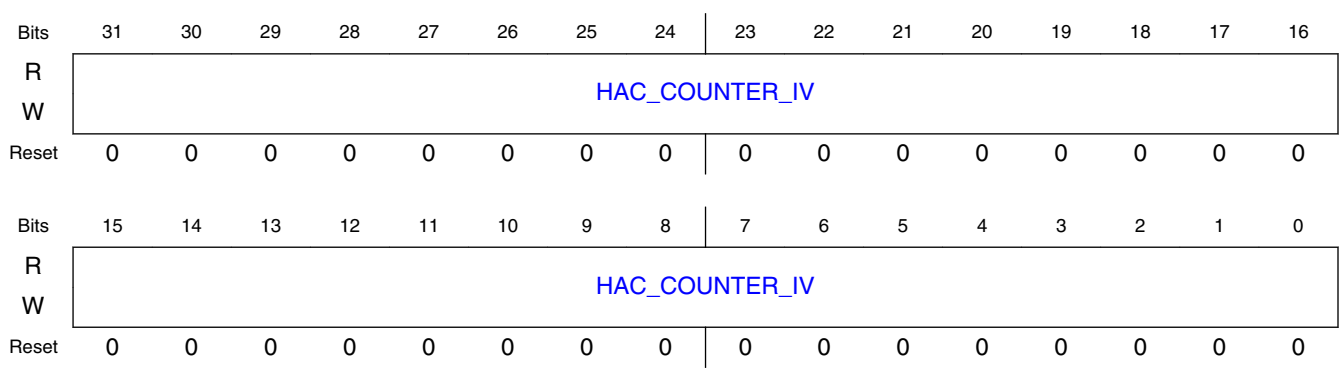
46.7.9.1 Address

Register	Offset
HPHACIVR	1Ch

46.7.9.2 Function

The SNVS_HP High Assurance Counter IV Register contains the initial value for the high assurance counter.

46.7.9.3 Diagram



46.7.9.4 Fields

Field	Function
31-0	

Field	Function
HAC_COUNTER_IV	High Assurance Counter Initial Value This register is used to set the starting count value to the high assurance counter. This register cannot be programmed when HAC_L bit is set.

46.7.10 SNVS_HP High Assurance Counter (HPHACR)

46.7.10.1 Address

Register	Offset
HPHACR	20h

46.7.10.2 Function

The SNVS_HP High Assurance Counter Register contains the value of the high assurance counter. The high assurance counter is a delay introduced before the system security monitor transitions from soft fail to hard fail state if this transition is enabled.

46.7.10.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HAC_COUNTER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.10.4 Fields

Field	Function
31-0	

Field	Function
HAC_COUNTER	High Assurance Counter When the HAC_EN bit is set and the SSM is in the soft fail state, this counter starts to count down with the system clock. When the counter reaches zero, the SSM transitions to the Hard Fail State. • When HAC_STOP bit is set, the HAC Counter is stopped. • When HAC_CLEAR bit is set, the HAC Counter is cleared. • When HAC_LOAD bit is set, the HAC Counter is loaded with the value of the HPHACIVR.

46.7.11 SNVS_HP Real Time Counter MSB (HPRTCMR)

46.7.11.1 Address

Register	Offset
HPRTCMR	24h

46.7.11.2 Function

The SNVS_HP Real Time Counter MSB register contains the 15 most-significant bits of the HP Real Time Counter.

46.7.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.11.4 Fields

Field	Function
31-15 —	Reserved
14-0 RTC	HP Real Time Counter The most-significant 15 bits of the RTC. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

46.7.12 SNVS_HP Real Time Counter LSB (HPRTCLR)

46.7.12.1 Address

Register	Offset
HPRTCLR	28h

46.7.12.2 Function

The SNVS_HP Real Time Counter LSB register contains the 32 least-significant bits of the HP real time counter.

46.7.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.12.4 Fields

Field	Function
31-0 RTC	HP Real Time Counter least-significant 32 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

46.7.13 SNVS_HP Time Alarm MSB (HPTAMR)

46.7.13.1 Address

Register	Offset
HPTAMR	2Ch

46.7.13.2 Function

The SNVS_HP Time Alarm MSB register contains the most-significant bits of the SNVS_HP Time Alarm value.

46.7.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.13.4 Fields

Field	Function
31-15 —	
14-0 HPTA_MS	HP Time Alarm, most-significant 15 bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

46.7.14 SNVS_HP Time Alarm LSB (HPTALR)

46.7.14.1 Address

Register	Offset
HPTALR	30h

46.7.14.2 Function

The SNVS_HP Time Alarm LSB register contains the 32 least-significant bits of the SNVS_HP Time Alarm value.

46.7.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HPTA_LS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.14.4 Fields

Field	Function
31-0 HPTA_LS	HP Time Alarm, 32 least-significant bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

46.7.15 SNVS_LP Lock (LPLR)

46.7.15.1 Address

Register	Offset
LPLR	34h

46.7.15.2 Function

The SNVS_LP Lock Register contains lock bits for the SNVS_LP registers.

46.7.15.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			AT5_HL	AT4_HL	AT3_HL	AT2_HL	AT1_HL	Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						MKS_HL	LPTDCR_HL	LPTGFCR_HL	LPSVCR_HL	GPR_HL	MC_HL	LPCALB_HL	SRTC_HL	ZMK_RHL	ZMK_WHL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.15.4 Fields

Field	Function
31-29 —	
28 AT5_HL	Active Tamper 5 Hard Lock When set, prevents any writes to the Active Tamper 5 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
27 AT4_HL	Active Tamper 4 Hard Lock When set, prevents any writes to the Active Tamper 4 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
26 AT3_HL	Active Tamper 3 Hard Lock When set, prevents any writes to the Active Tamper 3 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
25 AT2_HL	Active Tamper 2 Hard Lock When set, prevents any writes to the Active Tamper 2 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
24 AT1_HL	Active Tamper 1 Hard Lock When set, prevents any writes to the Active Tamper 1 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
23-10 —	
9 MKS_HL	Master Key Select Hard Lock When set, prevents any writes to the MASTER_KEY_SEL field of the LP Master Key Control Register. Once set, this bit can only be reset by the LP POR. 0b - 1b -
8 LPTDCR_HL	LP Tamper Detectors Configuration Register Hard Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -

Table continues on the next page...

Field	Function
7 LPTGFCR_HL	LP Tamper Glitch Filter Configuration Register Hard Lock When set, prevents any writes to the LPTGFCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -
6 LPSVCR_HL	LP Security Violation Control Register Hard Lock When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -
5 GPR_HL	General Purpose Register Hard Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the LP POR. 0b - 1b -
4 MC_HL	Monotonic Counter Hard Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the LP POR. 0b - 1b -
3 LPCALB_HL	LP Calibration Hard Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the LP POR. 0b - 1b -
2 SRTC_HL	Secure Real Time Counter Hard Lock When set, prevents any writes to the SRTC registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the LP POR. 0b - 1b -
1 ZMK_RHL	Zeroizable Master Key Read Hard Lock When set, prevents any software reads to the ZMK registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), software cannot read the ZMK or ZMK_ECC_VALUE. Regardless of the setting of this bit, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the LP POR. 0b - 1b -
0 ZMK_WHL	Zeroizable Master Key Write Hard Lock When set, prevents any writes (software and hardware) to the ZMK registers and ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -

46.7.16 SNVS_LP Control (LPCR)

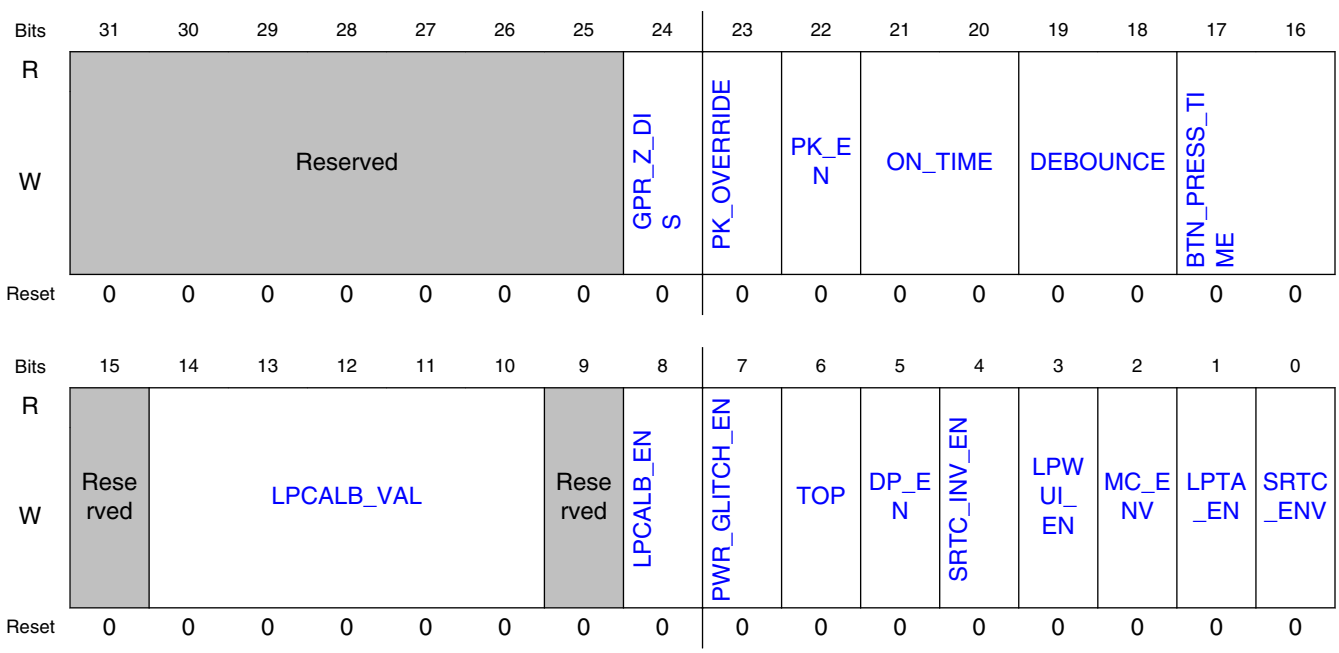
46.7.16.1 Address

Register	Offset
LPCR	38h

46.7.16.2 Function

The SNVS_LP Control Register contains various control bits of the LP section of SNVS.

46.7.16.3 Diagram



46.7.16.4 Fields

Field	Function
31-25	

Table continues on the next page...

Field	Function
—	
24 GPR_Z_DIS	General Purpose Registers Zeroization Disable. 1 = Disable zeroization of the GPR registers when a tamper event occurs. 0 = Zeroize the GPR registers when a tamper event occurs. (Default)
23 PK_OVERRIDE	PMIC On Request Override The value written to PK_OVERRIDE will be asserted on output signal snvs_lp_pk_override. That signal is used to override the IOMUX control for the PMIC I/O pad.
22 PK_EN	PMIC On Request Enable The value written to PK_EN will be asserted on output signal snvs_lp_pk_en. That signal is used to turn off the pullup/pulldown circuitry in the PMIC I/O pad.
21-20 ON_TIME	The ON_TIME field is used to configure the period of time after BTN is asserted before pmic_en_b is asserted to turn on the SoC power. 00: 500msec off->on transition time 01: 50msec off->on transition time 10: 100msec off->on transition time 11: 0msec off->on transition time
19-18 DEBOUNCE	This field configures the amount of debounce time for the BTN input signal. 00: 50msec debounce 01: 100msec debounce 10: 500msec debounce 11: 0msec debounce
17-16 BTN_PRESS_TIME	This field configures the button press time out values for the PMIC Logic. 00 : 5 secs 01 : 10 secs 10 : 15 secs 11 : long press disabled (pmic_en_b will not be asserted regardlessof how long BTN is asserted)
15 —	
14-10 LPCALB_VAL	LP Calibration Value Defines signed calibration value for SRTC. This field can be programmed only when SRTC calibration is disabled and not locked, i.e. when LPCALB_EN, LPCALB_SL, and LPCALB_HL bits are not set. This is a 5-bit 2's complement value. Hence, the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter clock 00000b - 00001b - 00010b - 01111b - 10000b - 10001b - 11110b - 11111b -

Table continues on the next page...

Field	Function
9 —	
8 LPCALB_EN	LP Calibration Enable When set, enables the SRTC calibration mechanism. This bit cannot be changed once LPCALB_SL or LPCALB_HL bit is set. 0b - 1b -
7 PWR_GLITCH_EN	Power Glitch Enable By default the detection of a power glitch does not cause the pmic_en_b signal to be asserted. Setting the Power Glitch Enable bit to 1 enables the power glitch event for the PMIC. 0 - disabled 1 - enabled
6 TOP	Turn off System Power Asserting this bit causes a signal to be sent to the Power Management IC to turn off the system power. This bit will clear once power is off. This bit is only valid when the Dumb PMIC is enabled. 0b - 1b -
5 DP_EN	Dumb PMIC Enabled When set, software can control the system power. When cleared, the system requires a Smart PMIC to automatically turn power off. 0b - 1b -
4 SRTC_INV_EN	If this bit is 1, in the case of a security violation the SRTC stops counting and the SRTC is invalidated (SRTC_ENV bit is cleared). This is intended to allow software to read the time at which the security violation occurred. This field cannot be changed once SRTC_SL or SRTC_HL bit is set. 0b - 1b -
3 LPWUI_EN	LP Wake-Up Interrupt Enable This interrupt line should be connected to the external pin and is intended to inform the external chip about an SNVS_LP event (tamper event, MC rollover,SRTC rollover, or time alarm). This wake-up signal can be asserted only when the chip (HP section) is powered down, and the LP section is isolated. 0 LP wake-up interrupt is disabled. 1 LP wake-up interrupt is enabled.
2 MC_ENV	Monotonic Counter Enabled and Valid When set, the MC can be incremented (by write transaction to the LPSMCMR or LPSMCLR). This bit cannot be changed once MC_SL or MC_HL bit is set. 0b - 1b -
1 LPTA_EN	LP Time Alarm Enable When set, the SNVS functional interrupt is asserted if the LP Time Alarm Register is equal to the 32 MSBs of the secure real time counter.

Table continues on the next page...

Field	Function
	0b - 1b -
0 SRTC_ENV	Secure Real Time Counter Enabled and Valid When set, the SRTC becomes operational. This bit cannot be changed once SRTC_SL or SRTC_HL bit is set. 0b - 1b -

46.7.17 SNVS_LP Master Key Control (LPMKCR)

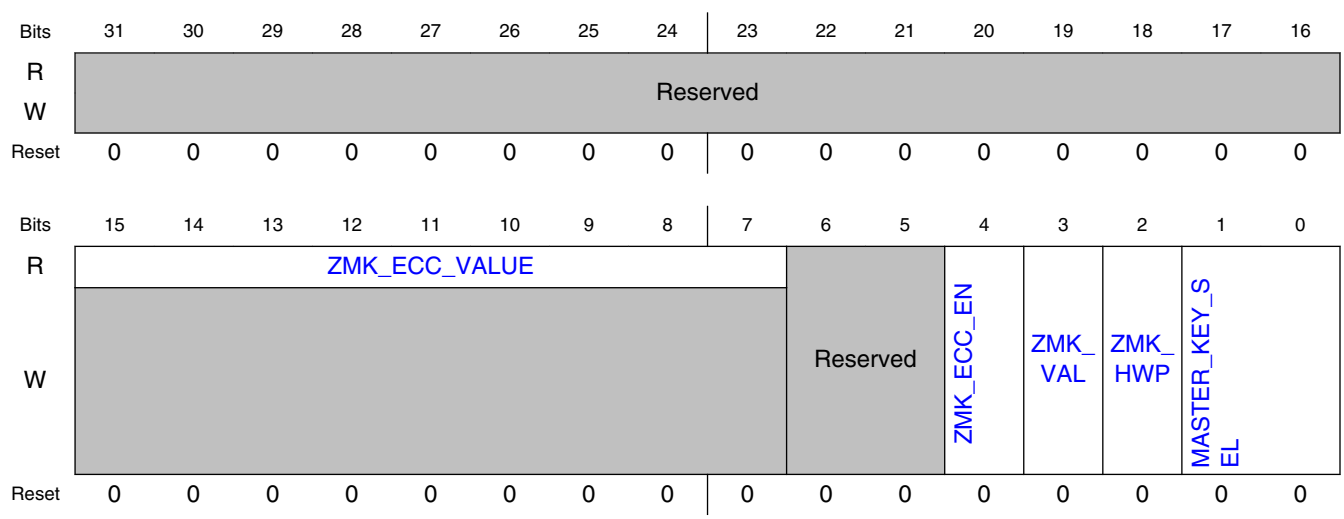
46.7.17.1 Address

Register	Offset
LPMKCR	3Ch

46.7.17.2 Function

The SNVS_LP Master Key Control Register contains the master keys configuration.

46.7.17.3 Diagram



46.7.17.4 Fields

Field	Function
31-16 —	
15-7 ZMK_ECC_VALUE	Zeroizable Master Key Error Correcting Code Value This field is automatically calculated and set when one is written into ZMK_ECC_EN bit of this register. This field cannot be programmed by software. It keeps the ECC value of the zeroizable master key, which allows checking that ZMK has not been corrupted/altered with time. Note that this ZMK ECC code is equivalent to the ECC bits encoded into the OTPMK value but for the ZMK, the ECC value is kept separate from the ZMK value. See Error code for the ZMK for details. Read restrictions similar to the ZMK Registers are applied to this field (see).
6-5 —	
4 ZMK_ECC_EN	Zeroizable Master Key Error Correcting Code Check Enable Writing one to this field automatically calculates and sets the ZMK ECC value in the ZMK_ECC_VALUE field of this register. When both ZMK value is valid (ZMK_VAL is set) and ZMK ECC check is enabled (ZMK_ECC_EN is set), the ZMK value is continuously checked for the valid ECC word. If the ZMK ECC word calculated every clock cycle does not match the one recorded in this register, the ZMK ECC Check Fail Violation is generated. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0b - 1b -
3 ZMK_VAL	Zeroizable Master Key Valid When set, the ZMK value can be selected by the master key control block for use by cryptographic modules. In hardware programming mode, hardware sets this bit when the ZMK provisioning is complete. In software programming mode, software should set this bit. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0b - 1b -
2 ZMK_HWP	Zeroizable Master Key hardware Programming mode When set, only the hardware key programming mechanism can set the ZMK and software cannot read it. When not set, the ZMK can be programmed only by software. See for details. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0b - 1b -
1-0 MASTER_KEY_SEL	Master Key Select These bits select the SNVS Master Key output when Master Key Select bits are enabled by MKS_EN bit in the HPCOMR . When MKS_EN bit is not set, the one time programmable master key is selected by default. This field cannot be programmed when MKS_SL(or the hard lock) bit is set. 0xb - 10b - Select zeroizable master key when MKS_EN bit is set . 11b - Select combined master key when MKS_EN bit is set .

46.7.18 SNVS_LP Security Violation Control (LPSVCR)

46.7.18.1 Address

Register	Offset
LPSVCR	40h

46.7.18.2 Function

The LP Security Violation Control Register configures security violation inputs. This register cannot be programmed when the LPSVCR Lock bit is set. Note that configurations of the security violation inputs in the HP section (HPSVCR) and LP section (LPSVCR Register) are independent and have different functionality.

46.7.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										SV_EN5	SV_EN4	SV_EN3	SV_EN2	SV_EN1	SV_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.18.4 Fields

Field	Function
31-6 —	
5 SV_EN5	Security Violation 5 Enable

Table continues on the next page...

Field	Function
	This bit enables security violation input 5. When set, a security violation 5 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
4 SV_EN4	Security Violation 4 Enable This bit enables security violation input 4. When set, a security violation 4 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
3 SV_EN3	Security Violation 3 Enable This bit enables security violation input 3. When set, a security violation 3 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
2 SV_EN2	Security Violation 2 Enable This bit enables security violation input 2. When set, a security violation 2 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
1 SV_EN1	Security Violation 1 Enable This bit enables security violation input 1. When set, a security violation 1 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
0 SV_EN0	Security Violation 0 Enable This bit enables security violation input 0. When set, a security violation 0 causes an LP security violation, which clears LP sensitive data. 0b - 1b -

46.7.19 SNVS_LP Tamper Glitch Filters Configuration (LPTGFCR)

46.7.19.1 Address

Register	Offset
LPTGFCR	44h

46.7.19.2 Function

The SNVS_LP Tamper Glitch Filters Configuration Register is used to configure the glitch filters for the SNVS_LP tamper inputs. This register cannot be programmed when the LPTGFCR_SL or LPTGFCR_HL bit is set.

46.7.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ETGF2_EN	ETGF2							ETGF1_EN	ETGF1						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								WMTGF_EN	Reserved		WMTGF				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.19.4 Fields

Field	Function
31 ETGF2_EN	External Tamper Glitch Filter 2 Enable When set, enables the external tamper glitch filter 2. 0b - 1b -
30-24 ETGF2	External Tamper Glitch Filter 2 Configures the length of the digital glitch filter for the external tamper 2 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 2 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF2 x 256), where ETGF2 = 0, ... , 127
23 ETGF1_EN	External Tamper Glitch Filter 1 Enable When set, enables the external tamper glitch filter 1. 0b - 1b -
22-16 ETGF1	External Tamper Glitch Filter 1

Table continues on the next page...

Field	Function
	Configures the length of the digital glitch filter for the external tamper 1 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 1 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF1 x 256), where ETGF1 = 0, ... , 127
15-8 —	
7 WMTGF_EN	Wire-Mesh Tamper Glitch Filter Enable When set, enables the wire-mesh tamper glitch filter. Note that two wire-mesh tamper inputs (1 and 2) share this glitch filter. 0b - 1b -
6-5 —	
4-0 WMTGF	Wire-Mesh Tamper Glitch Filter Configures the length of the digital glitch filter for the wire-mesh tamper 1 and 2 pins between 1 and 63 SRTC clock cycles. Two wire-mesh tamper inputs share this glitch filter. Any assertion on wire-mesh tamper 1 or 2 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 1 + (WMTGF x 2), where WMTGF = 0, ... , 31

46.7.20 SNVS_LP Tamper Detectors Configuration (LPTDCR)

46.7.20.1 Address

Register	Offset
LPTDCR	48h

46.7.20.2 Function

The SNVS_LP Tamper Detectors Configuration Register is used to configure analog and digital tamper detector sources. This register cannot be programmed when LPTDCR is locked for write.

46.7.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			OSCB	Reserved	VRC			Reserved	HTDC			Reserved	LTDC		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POR_OBSERV	PFD_OBSERV	Reserved	ET2P	ET1P	ET2_EN	ET1_EN	WMT2_EN	WMT1_EN	VT_EN	TT_EN	CT_EN	Reserved	MCR_EN	SRTCR_EN	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.20.4 Fields

Field	Function
31-29 —	
28 OSCB	Oscillator Bypass When OSCB=1 the osc_bypass signal is asserted. That signal asks SoC logic external to SNVS to bypass the SoC's normal SRTC clock source and drive the SRTC clock from an alternate source. 0b - 1b -
27 —	
26-24 VRC	Voltage Reference Configuration These configuration bits are wired as an output of the module.
23 —	
22-20 HTDC	High Temperature Detect Configuration These configuration bits are wired as an output of the module.
19 —	
18-16 LTDC	Low Temp Detect Configuration These configuration bits are wired as an output of the module.
15 POR_OBSERV	Power On Reset (POR) Observability Flop

Table continues on the next page...

Field	Function
	The asynchronous reset input of this flop is connected directly to the output of the POR analog circuitry (external to the SNVS. This flop can be used to detect brown-out voltage of the POR circuitry.
14 PFD_OBSERV	System Power Fail Detector (PFD) Observability Flop The asynchronous reset input of this flop is connected directly to the inverted output of the PFD analog circuitry (external to the SNVS block). This flop can be used to detect brown-out voltage of the PFD circuitry.
13 —	
12 ET2P	External Tampering 2 Polarity This bit is used to determine the polarity of external tamper 2. 0b - 1b -
11 ET1P	External Tampering 1 Polarity This bit is used to determine the polarity of external tamper 1. 0b - 1b -
10 ET2_EN	External Tampering 2 Enable When set, external tampering 2 detection generates an LP security violation. 0b - 1b -
9 ET1_EN	External Tampering 1 Enable When set, external tampering 1 detection generates an LP security violation. 0b - 1b -
8 WMT2_EN	Wire-Mesh Tampering 2 Enable When set, wire-mesh tampering 2 detection generates an LP security violation. 0b - 1b -
7 WMT1_EN	Wire-Mesh Tampering 1 Enable When set, wire-mesh tampering 1 detection generates an LP security violation. 0b - 1b -
6 VT_EN	Voltage Tamper Enable **NOTE:** Voltage Tamper Enable should be enabled 500 us after setting SCSC_SOSC_CTR [VOLT_TEMP_TAMPER_EN]. After enabling voltage tamper enable, clear SNVSLPSR [VTD] if set. When set, a voltage monitor tamper generates an LP security violation. 0b - 1b -
5	Temperature Tamper Enable

Table continues on the next page...

Field	Function
TT_EN	When set, a temperature monitor tamper generates an LP security violation. **NOTE:** Temperature Tamper Enable should be enabled 500 us after setting SCSC_SOSC_CTR [VOLT_TEMP_TAMPER_EN]. After enabling temperature tamper enable, clear SNVS_LPSR [TTD] if set. 0b - 1b -
4 CT_EN	Clock Tamper Enable When set, a clock monitor tamper generates an LP security violation. 0b - 1b -
3 —	Reserved
2 MCR_EN	MC Rollover Enable When set, an MC Rollover event generates an LP security violation. 0b - 1b -
1 SRTCR_EN	SRTC Rollover Enable When set, an SRTC rollover event generates an LP security violation. 0b - 1b -
0 —	

46.7.21 SNVS_LP Status (LPSR)

46.7.21.1 Address

Register	Offset
LPSR	4Ch

46.7.21.2 Function

The SNVS_LP Status Register reflects the internal state and behavior of the SNVS_LP.

46.7.21.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LPS	LPNS	Reserved									SED	Reserved	SPO	EO	ESVD
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					ET2D	ET1D	WMT2D	WMT1D	VTD	TTD	CTD	PGD	MCR	SRTCR	LPTA
W						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

46.7.21.4 Fields

Field	Function
31 LPS	LP Section is Secured Indicates that the LP section is provisioned/programmed in the secure or trusted state. The first write to the LP registers in secure or trusted state sets this bit. This bit can never be set together with the LPNS bit. When set the SNVS_LP section cannot be programmed and ZMK cannot be read in the non-secure state of the SSM. 0b - 1b -
30 LPNS	LP Section is Non-Secured Indicates that LP section was provisioned/programmed in the non-secure state. The first successful write to the LP Registers in non-secure state sets this bit. This bit can never be set together with the LPS bit. When set, the entire SNVS_LP section (all LP registers) are cleared upon an SSM transition from check to trusted state. 0b - 1b -
29-21 —	
20 SED	Scan Exit Detected 0b - 1b -
19 —	
18 SPO	Set Power Off The SPO bit is set when the set_pwr_off_irq interrupt is triggered, which happens when software writes a 1 to the TOP bit in the LPCR or when the power button is pressed longer than the configured debounce time. Writing to the SPO bit will clear the set_pwr_off_irq interrupt.

Table continues on the next page...

Field	Function
	0b - 1b -
17 EO	Emergency Off This bit is set when a power off is requested. 0b - 1b -
16 ESVD	External Security Violation Detected Indicates that a security violation is detected on one of the HP security violation ports. The record of the port on which the violation has occurred can be found in the HP Security Violation Status Register. 0b - 1b -
15-11 —	
10 ET2D	0b - 1b -
9 ET1D	External Tampering 1 Detected 0b - 1b -
8 WMT2D	Wire-Mesh Tampering 2 Detected 0b - 1b -
7 WMT1D	Wire-Mesh Tampering 1 Detected 0b - 1b -
6 VTD	Voltage Tampering Detected 0b - 1b -
5 TTD	Temperature Tamper Detected 0b - 1b -
4 CTD	Clock Tampering Detected 0b - 1b -
3 PGD	Power Supply Glitch Detected 0 No power supply glitch. 1 Power supply glitch is detected.
2 MCR	Monotonic Counter Rollover 0b - 1b -
1 SRTCR	Secure Real Time Counter Rollover 0b - 1b -

Table continues on the next page...

Field	Function
0 LPTA	LP Time Alarm 0b - 1b -

46.7.22 SNVS_LP Secure Real Time Counter MSB (LPSRTCMR)

46.7.22.1 Address

Register	Offset
LPSRTCMR	50h

46.7.22.2 Function

The SNVS_LP Secure Real Time Counter MSB register contains the 15 most-significant bits of the LP Secure Real Time Counter.

46.7.22.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.22.4 Fields

Field	Function
31-15	

Table continues on the next page...

Field	Function
—	
14-0 SRTC	LP Secure Real Time Counter The most-significant 15 bits of the SRTC. This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

46.7.23 SNVS_LP Secure Real Time Counter LSB (LPSRTCLR)

46.7.23.1 Address

Register	Offset
LPSRTCLR	54h

46.7.23.2 Function

The SNVS_LP Secure Real Time Counter LSB register contains the 32 least-significant bits of the secure real time counter.

46.7.23.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SRTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.23.4 Fields

Field	Function
31-0 SRTC	LP Secure Real Time Counter least-significant 32 bits This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

46.7.24 SNVS_LP Time Alarm (LPTAR)

46.7.24.1 Address

Register	Offset
LPTAR	58h

46.7.24.2 Function

The SNVS_LP Time Alarm register contains the 32-bit LP Time Alarm value.

46.7.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LPTA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.24.4 Fields

Field	Function
31-0 LPTA	LP Time Alarm This register can be programmed only when the LP time alarm is disabled (LPTA_EN bit is not set).

46.7.25 SNVS_LP Secure Monotonic Counter MSB (LPSMCMR)

46.7.25.1 Address

Register	Offset
LPSMCMR	5Ch

46.7.25.2 Function

The SNVS_LP Secure Monotonic Counter MSB Register contains the monotonic counter era bits and the most-significant 16 bits of the monotonic counter. The monotonic counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

46.7.25.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MC_ERA_BITS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.25.4 Fields

Field	Function
31-16 MC_ERA_BITS	Monotonic Counter Era Bits These bits are inputs to the module and typically connect to fuses.
15-0 MON_COUNTER	Monotonic Counter most-significant 16 Bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

46.7.26 SNVS_LP Secure Monotonic Counter LSB (LPSMCLR)

46.7.26.1 Address

Register	Offset
LPSMCLR	60h

46.7.26.2 Function

The SNVS_LP Secure Monotonic Counter LSB Register contains the 32 least-significant bits of the monotonic counter. The MC is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

46.7.26.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MON_COUNTER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.26.4 Fields

Field	Function
31-0 MON_COUNTER	Monotonic Counter bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR Register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

46.7.27 SNVS_LP Power Glitch Detector (LPPGDR)

46.7.27.1 Address

Register	Offset
LPPGDR	64h

46.7.27.2 Function

The SNVS_LP Power Glitch Detector Register is a 32-bit read/write register that is used for storing the power glitch detector value, as described in .

46.7.27.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PGD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.27.4 Fields

Field	Function
31-0 PGD	Power Glitch Detector Value

46.7.28 SNVS_LP General Purpose 0 (alias) (LPGPR0_alias)

46.7.28.1 Address

Register	Offset
LPGPR0_alias	68h

46.7.28.2 Function

The SNVS_LP General Purpose Register is a 128-bit read/write register located in the low power domain, which can be used by any application for retaining data during an SoC power-down mode. The register is accessed as four 32-bit registers located in successive word addresses starting at offset 90h. For backward compatibility with earlier versions of SNVS LPGPR0 is aliased at its original offset of 68h.The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

46.7.28.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.28.4 Fields

Field	Function
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

46.7.29 SNVS_LP Zeroizable Master Key (LPZMKRa)

46.7.29.1 Address

For a = 0 to 31:

Register	Offset
LPZMKRa	6Ch + (a × 1h)

46.7.29.2 Function

The SNVS_LP Zeroizable Master Key Registers contain the 256-bit zeroizable master key value. These registers are programmable as follows:

- When ZMK write lock bit is set, they cannot be programmed.
- When ZMK_HWP is not set, they are in software programming mode and can be programmed only by software.
- When ZMK_HWP is set, they are in hardware programming mode and can be programmed only by hardware.

These registers cannot be read by software when the ZMK_HWP or ZMK read lock bit is set.

46.7.29.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	ZMK							
Reset	0	0	0	0	0	0	0	0

46.7.29.4 Fields

Field	Function
7-0 ZMK	Zeroizable Master Key Each of these registers contains part of the 256-bit ZMK value. When accessed via the register bus this should be treated as a byte array with the first byte in offset 6Ch (although the register must be accessed as eight 32-bit words).

46.7.30 SNVS_LP General Purposes 0 .. 3 (LPGPR0_3a)

46.7.30.1 Address

Register	Offset
LPGPR0_30	90h
LPGPR0_31	94h
LPGPR0_32	98h
LPGPR0_33	9Ch

46.7.30.2 Function

The SNVS_LP General Purpose Register is a 128-bit read/write register located in the low power domain, which can be used by any application for retaining data during an SoC power-down mode. The register is accessed as four 32-bit registers located in successive word addresses starting at offset 90h. For backward compatibility with earlier versions of SNVS LPGPR0 is aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

46.7.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.30.4 Fields

Field	Function
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

46.7.31 SNVS_LP Tamper Detectors Config 2 (LPTDC2R)

46.7.31.1 Address

Register	Offset
LPTDC2R	A0h

46.7.31.2 Function

The SNVS_LP Tamper Detectors Configuration 2 Register is used to configure digital external tamper sources. This register cannot be programmed when LPTDCR is locked for write.

46.7.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ET10P	ET9P	ET8P	ET7P	ET6P	ET5P	ET4P	ET3P
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								ET10_EN	ET9_EN	ET8_EN	ET7_EN	ET6_EN	ET5_EN	ET4_EN	ET3_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.31.4 Fields

Field	Function
31-24 —	
23 ET10P	External Tampering 10 Polarity This bit is used to determine the polarity of external tamper 10. 0b - 1b -
22 ET9P	External Tampering 9 Polarity This bit is used to determine the polarity of external tamper 9. 0b - 1b -
21 ET8P	External Tampering 8 Polarity This bit is used to determine the polarity of external tamper 8. 0b - 1b -
20 ET7P	External Tampering 7 Polarity This bit is used to determine the polarity of external tamper 7. 0b - 1b -
19 ET6P	External Tampering 6 Polarity This bit is used to determine the polarity of external tamper 6. 0b - 1b -
18	External Tampering 5 Polarity

Table continues on the next page...

Field	Function
ET5P	This bit is used to determine the polarity of external tamper 5. 0b - 1b -
17 ET4P	External Tampering 4 Polarity This bit is used to determine the polarity of external tamper 4. 0b - 1b -
16 ET3P	External Tampering 3 Polarity This bit is used to determine the polarity of external tamper 3. 0b - 1b -
15-8 —	
7 ET10_EN	External Tampering 10 Enable When set, external tampering 10 detection generates an LP security violation. 0b - 1b -
6 ET9_EN	External Tampering 9 Enable When set, external tampering 9 detection generates an LP security violation. 0b - 1b -
5 ET8_EN	External Tampering 8 Enable When set, external tampering 8 detection generates an LP security violation. 0b - 1b -
4 ET7_EN	External Tampering 7 Enable When set, external tampering 7 detection generates an LP security violation. 0b - 1b -
3 ET6_EN	External Tampering 6 Enable When set, external tampering 6 detection generates an LP security violation. 0b - 1b -
2 ET5_EN	External Tampering 5 Enable When set, external tampering 5 detection generates an LP security violation. 0b - 1b -
1 ET4_EN	External Tampering 4 Enable When set, external tampering 4 detection generates an LP security violation.

Table continues on the next page...

Field	Function
	0b - 1b -
0 ET3_EN	External Tampering 3 Enable When set, external tampering 3 detection generates an LP security violation. 0b - 1b -

46.7.32 SNVS_LP Tamper Detectors Status (LPTDSR)

46.7.32.1 Address

Register	Offset
LPTDSR	A4h

46.7.32.2 Function

The SNVS_LP Tamper Detectors Status Register reflects the status of the SNVS_LP external tampers 3-10.

46.7.32.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								ET10D	ET9D	ET8D	ET7D	ET6D	ET5D	ET4D	ET3D
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.32.4 Fields

Field	Function
31-8 —	
7 ET10D	External Tampering 10 Detected 0b - 1b -
6 ET9D	External Tampering 9 Enable When set, external tampering 9 detection generates an LP security violation. 0b - 1b -
5 ET8D	External Tampering 8 Detected 0b - 1b -
4 ET7D	External Tampering 7 Detected 0b - 1b -
3 ET6D	External Tampering 6 Detected 0b - 1b -
2 ET5D	External Tampering 5 Detected 0b - 1b -
1 ET4D	External Tampering 4 Detected 0b - 1b -
0 ET3D	External Tampering 3 Detected 0b - 1b -

46.7.33 SNVS_LP Tamper Glitch Filter 1 Configuration (LPTGF1CR)

46.7.33.1 Address

Register	Offset
LPTGF1CR	A8h

46.7.33.2 Function

The SNVS_LP Tamper Glitch Filters Configuration Register is used to configure the glitch filters for the SNVS_LP external tamper inputs 3-6. This register cannot be programmed when the LPTGFCR_SL or LPTGFCR_HL bit is set.

46.7.33.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ETGF6_EN	ETGF6							ETGF5_EN	ETGF5						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ETGF4_EN	ETGF4							ETGF3_EN	ETGF3						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.33.4 Fields

Field	Function
31 ETGF6_EN	External Tamper Glitch Filter 6 Enable When set, enables the external tamper glitch filter 6. 0b - 1b -
30-24 ETGF6	External Tamper Glitch Filter 6 Configures the length of the digital glitch filter for the external tamper 6 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 6 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF6 x 256), where ETGF6 = 0, ... , 127
23 ETGF5_EN	External Tamper Glitch Filter 5 Enable When set, enables the external tamper glitch filter 5. 0b - 1b -
22-16 ETGF5	External Tamper Glitch Filter 5

Table continues on the next page...

Field	Function
	Configures the length of the digital glitch filter for the external tamper 5 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 5 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF5 x 256), where ETGF5 = 0, ... , 127
15 ETGF4_EN	External Tamper Glitch Filter 4 Enable When set, enables the external tamper glitch filter 4. 0b - 1b -
14-8 ETGF4	External Tamper Glitch Filter 4 Configures the length of the digital glitch filter for the external tamper 4 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 4 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF4 x 256), where ETGF4 = 0, ... , 127
7 ETGF3_EN	External Tamper Glitch Filter 3 Enable When set, enables the external tamper glitch filter 3. 0b - 1b -
6-0 ETGF3	External Tamper Glitch Filter 3 Configures the length of the digital glitch filter for the external tamper 3 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 3 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF3 x 256), where ETGF3 = 0, ... , 127

46.7.34 SNVS_LP Tamper Glitch Filter 2 Configuration (LPTGF2CR)

46.7.34.1 Address

Register	Offset
LPTGF2CR	ACh

46.7.34.2 Function

The SNVS_LP Tamper Glitch Filters Configuration Register is used to configure the glitch filters for the SNVS_LP> external tamper inputs 7-10. This register cannot be programmed when the LPTGFCR_SL or LPTGFCR_HL bit is set.

46.7.34.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ETGF10_EN	ETGF10							ETGF9_EN	ETGF9						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ETGF8_EN	ETGF8							ETGF7_EN	ETGF7						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.34.4 Fields

Field	Function
31 ETGF10_EN	External Tamper Glitch Filter 10 Enable When set, enables the external tamper glitch filter 10. 0b - 1b -
30-24 ETGF10	External Tamper Glitch Filter 10 Configures the length of the digital glitch filter for the external tamper 10 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 6 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF10 x 256), where ETGF10 = 0, ... , 127
23 ETGF9_EN	External Tamper Glitch Filter 9 Enable When set, enables the external tamper glitch filter 9. 0b - 1b -
22-16	External Tamper Glitch Filter 9

Table continues on the next page...

Field	Function
ETGF9	Configures the length of the digital glitch filter for the external tamper 9 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 5 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF9 x 256), where ETGF9 = 0, ... , 127
15 ETGF8_EN	External Tamper Glitch Filter 8 Enable When set, enables the external tamper glitch filter 8. 0b - 1b -
14-8 ETGF8	External Tamper Glitch Filter 8 Configures the length of the digital glitch filter for the external tamper 8 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 8 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF8 x 256), where ETGF8 = 0, ... , 127
7 ETGF7_EN	External Tamper Glitch Filter 7 Enable When set, enables the external tamper glitch filter 7. 0b - 1b -
6-0 ETGF7	External Tamper Glitch Filter 7 Configures the length of the digital glitch filter for the external tamper 7 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 7 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF7 x 256), where ETGF7 = 0, ... , 127

46.7.35 SNVS_LP Active Tamper 1 Configuration (LPAT1CR)

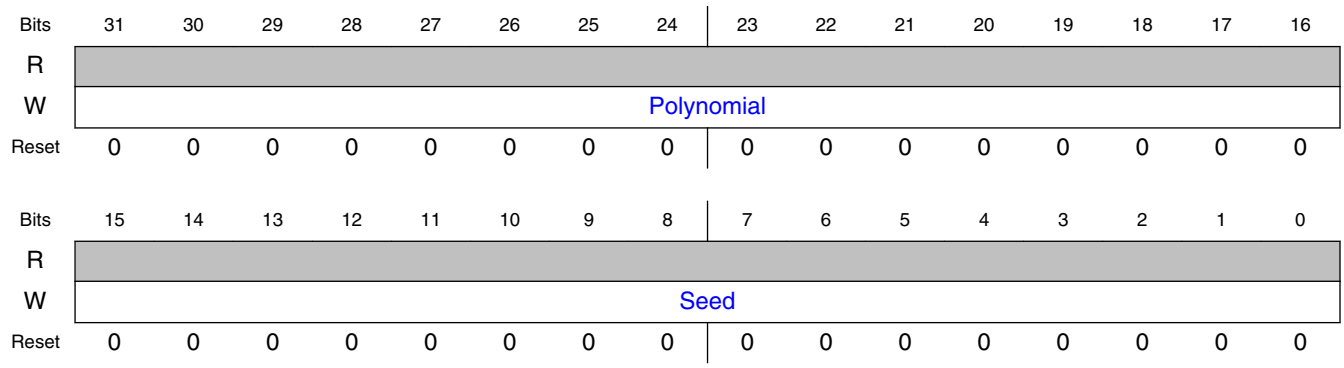
46.7.35.1 Address

Register	Offset
LPAT1CR	C0h

46.7.35.2 Function

The SNVS_LP Active Tamper 1 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT1EN bit is set.

46.7.35.3 Diagram



46.7.35.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 1 Polynomial Default Polynomial is 8400h.
15-0 Seed	Active Tamper 1 Initial Seed Default Seed is 1111h.

46.7.36 SNVS_LP Active Tamper 2 Configuration (LPAT2CR)

46.7.36.1 Address

Register	Offset
LPAT2CR	C4h

46.7.36.2 Function

The SNVS_LP Active Tamper 2 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT2EN bit is set.

46.7.36.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Polynomial															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.36.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 2 Polynomial Default Polynomial is 9C00h.
15-0 Seed	Active Tamper 2 Initial Seed Default Seed is 2222h.

46.7.37 SNVS_LP Active Tamper 3 Configuration (LPAT3CR)

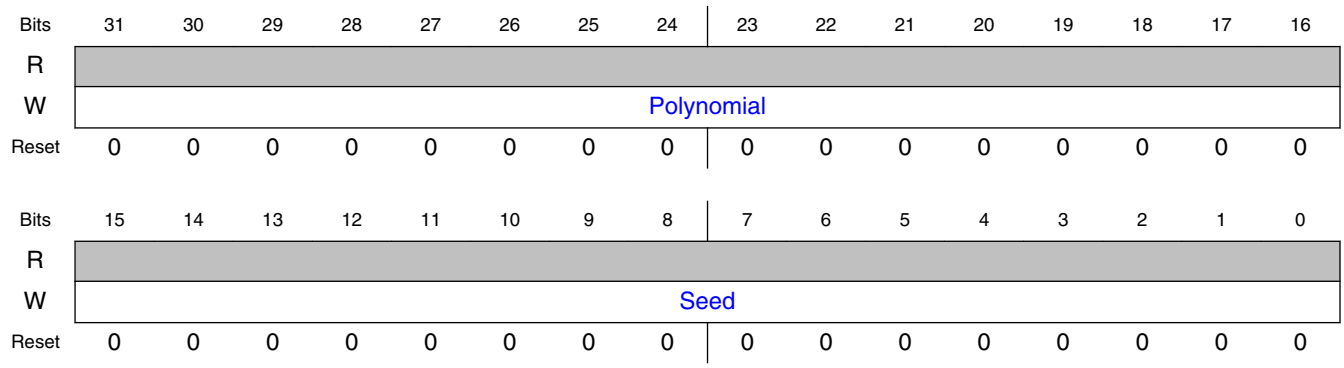
46.7.37.1 Address

Register	Offset
LPAT3CR	C8h

46.7.37.2 Function

The SNVS_LP Active Tamper 3 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT3EN bit is set.

46.7.37.3 Diagram



46.7.37.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 3 Polynomial Default Polynomial is CA00h.
15-0 Seed	Active Tamper 3 Initial Seed Default Seed is 3333h.

46.7.38 SNVS_LP Active Tamper 4 Configuration (LPAT4CR)

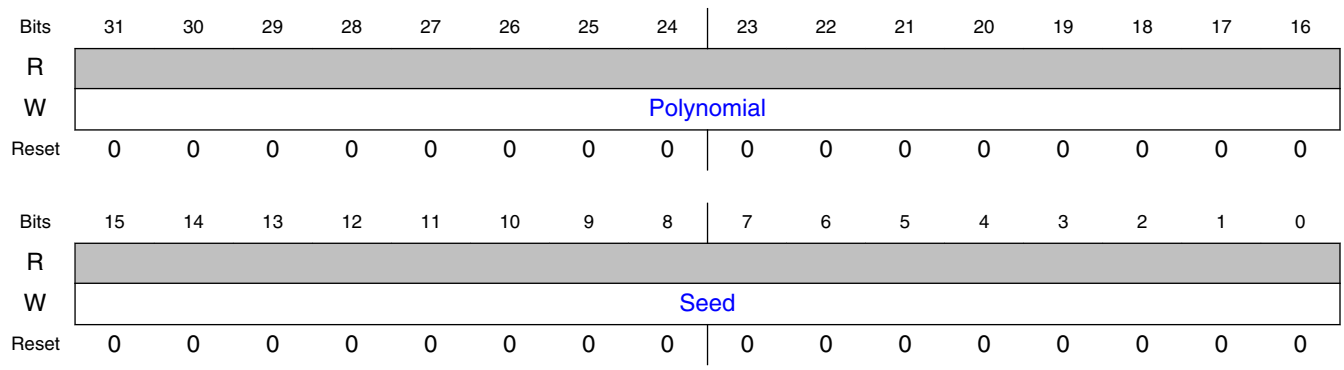
46.7.38.1 Address

Register	Offset
LPAT4CR	CCh

46.7.38.2 Function

The SNVS_LP Active Tamper 4 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT4EN bit is set.

46.7.38.3 Diagram



46.7.38.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 4 Polynomial Default Polynomial is 8580h.
15-0 Seed	Active Tamper 4 Initial Seed Default Seed is 4444h.

46.7.39 SNVS_LP Active Tamper 5 Configuration (LPAT5CR)

46.7.39.1 Address

Register	Offset
LPAT5CR	D0h

46.7.39.2 Function

The SNVS_LP Active Tamper 5 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT5EN bit is set.

46.7.39.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Polynomial															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

46.7.39.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 5 Polynomial Default Polynomial is A840h.
15-0 Seed	Active Tamper 5 Initial Seed Default Seed is 5555h.

46.7.40 SNVS_LP Active Tamper Control (LPATCTLR)

46.7.40.1 Address

Register	Offset
LPATCTLR	E0h

46.7.40.2 Function

The SNVS_LP Active Tamper Control Register is used to enable the LFSRs which is used for the SNVS_LP active tamper outputs. It is also used to control external pads to enable for input or output.

46.7.40.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved											AT5_PAD_EN	AT4_PAD_EN	AT3_PAD_EN	AT2_PAD_EN	AT1_PAD_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											AT5_EN	AT4_EN	AT3_EN	AT2_EN	AT1_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.40.4 Fields

Field	Function
31-21 —	
20 AT5_PAD_EN	Active Tamper 5 Pad Out Enable When set, enables the Active Tamper 5 external pad. 0b - 1b -
19 AT4_PAD_EN	Active Tamper 4 Pad Out Enable When set, enables the Active Tamper 4 external pad. 0b - 1b -
18 AT3_PAD_EN	Active Tamper 3 Pad Out Enable When set, enables the Active Tamper 3 external pad. 0b - 1b -
17	

Table continues on the next page...

Field	Function
AT2_PAD_EN	Active Tamper 2 Pad Out Enable When set, enables the Active Tamper 2 external pad. 0b - 1b -
16 AT1_PAD_EN	Active Tamper 1 Pad Out Enable When set, enables the Active Tamper 1 external pad. 0b - 1b -
15-5 —	
4 AT5_EN	Active Tamper 5 Enable When set, enables the Active Tamper 5 LFSR. 0b - 1b -
3 AT4_EN	Active Tamper 4 Enable When set, enables the Active Tamper 4 LFSR. 0b - 1b -
2 AT3_EN	Active Tamper 3 Enable When set, enables the Active Tamper 3 LFSR. 0b - 1b -
1 AT2_EN	Active Tamper 2 Enable When set, enables the Active Tamper 2 LFSR. 0b - 1b -
0 AT1_EN	Active Tamper 1 Enable When set, enables the Active Tamper 1 LFSR. 0b - 1b -

46.7.41 SNVS_LP Active Tamper Clock Control (LPATCLKR)

46.7.41.1 Address

Register	Offset
LPATCLKR	E4h

46.7.41.2 Function

The SNVS_LP Active Tamper Clock Control Register is used to define what frequency the LFSRs are run at for the SNVS_LP active tamper outputs. The Active Tamper clocks are based on the SRTC. The SRTC must be enabled for Active Tamper to work. The clock control fields are not writeable once an LFSR is enabled.

46.7.41.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														AT5_CLK_CTL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		AT4_CLK_CTL		Reserved		AT3_CLK_CTL		Reserved		AT2_CLK_CTL		Reserved		AT1_CLK_CTL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.41.4 Fields

Field	Function
31-18 —	
17-16 AT5_CLK_CTL	Active Tamper 5 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz
15-14 —	
13-12 AT4_CLK_CTL	Active Tamper 4 Clock Control 00: 16hz 01: 8hz 10: 4hz

Table continues on the next page...

Field	Function
	11: 2hz
11-10 —	
9-8 AT3_CLK_CTL	Active Tamper 3 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz
7-6 —	
5-4 AT2_CLK_CTL	Active Tamper 2 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz
3-2 —	
1-0 AT1_CLK_CTL	Active Tamper 1 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz

46.7.42 SNVS_LP Active Tamper Routing Control 1 (LPATRC1R)

46.7.42.1 Address

Register	Offset
LPATRC1R	E8h

46.7.42.2 Function

The SNVS_LP Active Tamper Routing Control Register routes an active tamper compare value to the proper external tamper detector.

46.7.42.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	ET8RCTL			Reserved	ET7RCTL			Reserved	ET6RCTL			Reserved	ET5RCTL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	ET4RCTL			Reserved	ET3RCTL			Reserved	ET2RCTL			Reserved	ET1RCTL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.42.4 Fields

Field	Function
31 —	
30-28 ET8RCTL	External Tamper 8 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
27 —	
26-24 ET7RCTL	External Tamper 7 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
23 —	

Table continues on the next page...

Field	Function
22-20 ET6RCTL	External Tamper 6 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
19 —	
18-16 ET5RCTL	External Tamper 5 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
15 —	
14-12 ET4RCTL	External Tamper 4 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
11 —	
10-8 ET3RCTL	External Tamper 3 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4

Table continues on the next page...

Field	Function
	101: Active Tamper 5
7 —	
6-4 ET2RCTL	External Tamper 2 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
3 —	
2-0 ET1RCTL	External Tamper 1 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5

46.7.43 SNVS_LP Active Tamper Routing Control 2 (LPATRC2R)

46.7.43.1 Address

Register	Offset
LPATRC2R	ECh

46.7.43.2 Function

The SNVS_LP Active Tamper Routing Control Register routes an active tamper compare value to the proper external tamper detector.

46.7.43.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									ET10RCTL			Reserved	ET9RCTL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.7.43.4 Fields

Field	Function
31-7 —	
6-4 ET10RCTL	External Tamper 10 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
3 —	
2-0 ET9RCTL	External Tamper 9 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5

46.7.44 SNVS_HP Version ID 1 (HPVIDR1)

46.7.44.1 Address

Register	Offset
HPVIDR1	BF8h

46.7.44.2 Function

The SNVS_HP Version ID Register 1 is a read-only register that contains the current version of the SNVS. The version consists of a module ID, a major version number, and a minor version number.

46.7.44.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IP_ID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0

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

46.7.44.4 Fields

Field	Function
31-16 IP_ID	SNVS block ID
15-8 MAJOR_REV	SNVS block major version number
7-0 MINOR_REV	SNVS block minor version number

46.7.45 SNVS_HP Version ID 2 (HPVIDR2)

46.7.45.1 Address

Register	Offset
HPVIDR2	BFCh

46.7.45.2 Function

The SNVS_HP Version ID Register 2 is a read-only register that indicates the current version of the SNVS. Version ID register 2 consists of the following fields: integration options, ECO revision, and configuration options.

46.7.45.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IP_ERA								INTG_OPT							
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

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

46.7.45.4 Fields

Field	Function
31-24 IP_ERA	IP Era 00h - Era 1 or 2 03h - Era 3 04h - Era 4

Table continues on the next page...

Field	Function
	05h - Era 5
23-16 INTG_OPT	SNVS Integration Options
15-8 ECO_REV	SNVS ECO Revision
7-0 CONFIG_OPT	SNVS Configuration Options

Chapter 47
Shared Peripheral Bus Arbiter (SPBA)

47.1 Overview

The Shared Peripheral Bus Arbiter (SPBA) is a three-to-one IP Bus interface arbiter. Three masters arbitrate for shared peripheral access through the SPBA.

The SPBA has three primary functions:

- The IP Bus Line switches a master to one peripheral
- The Masters arbiter arbitrates between the three masters to solve concurrent access or restricted access to peripherals
- The Control Registers and Ownership Control includes a set of registers which are reachable through software and permit the access scheme to be defined for each peripheral (Resource Ownership and Access Control). It generates signals for the external steering logic of interrupts and DMA signals.

The figure below shows the SPBA block diagram

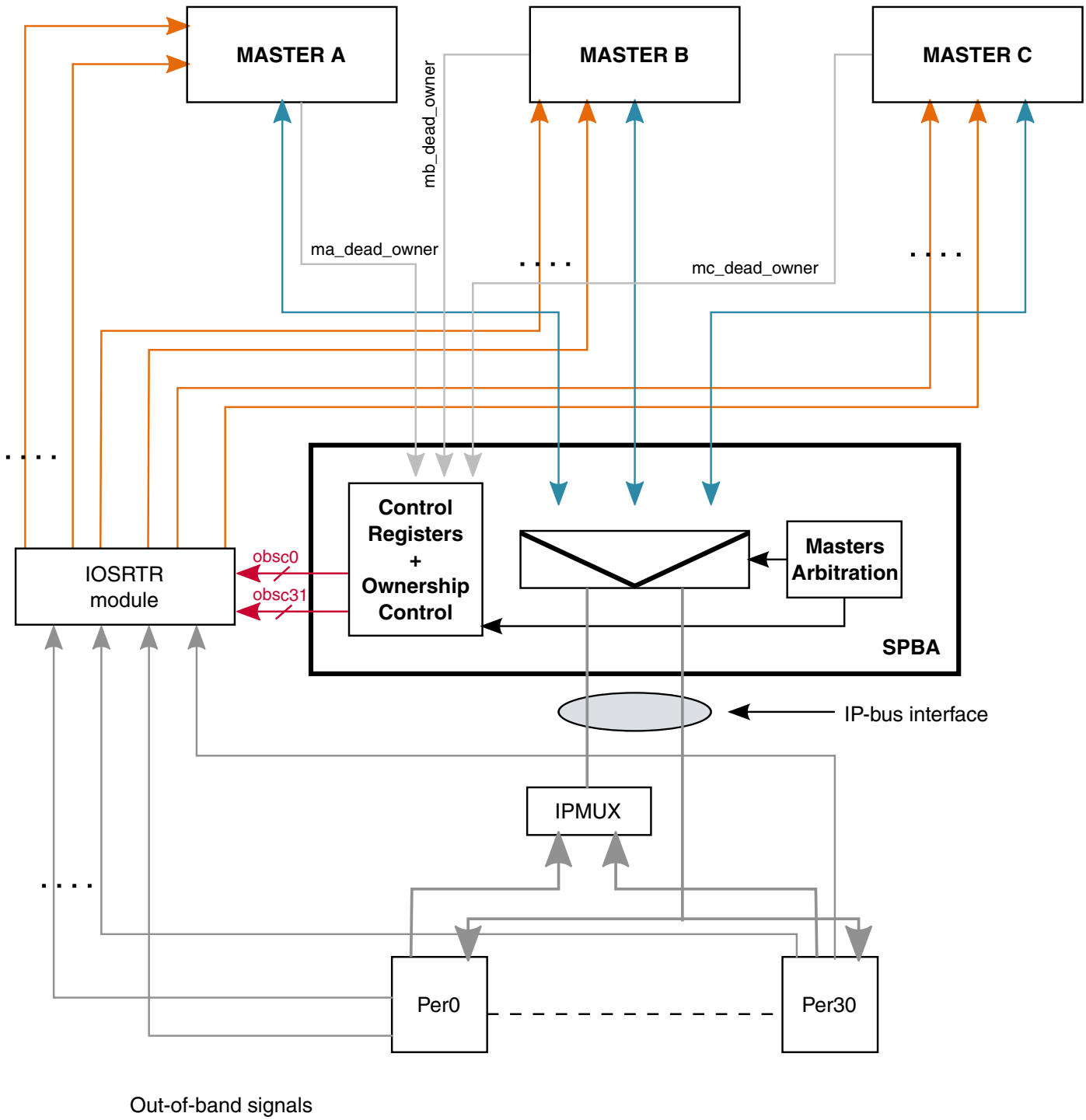


Figure 47-1. SPBA Block Diagram

47.1.1 Features

The SPBA includes the following features:

- Three IP Bus masters arbitration: Master A, B and C
- Support for DMA masters
- 32-bit data
- Supports up to 31 shared peripherals, each consuming 16 kilobytes of address space
- SPBA can be considered the 32nd peripheral, used for resource ownership and access control of the 31 peripherals
- Provides 31 sets of out of band steering control (OBSC) signals to the off-block steering logic
- Operating frequency up to 67 MHz
- Clocks: ipg_clk, ipg_clk_s

47.1.2 Modes of operation

SPBA behavior is transparent when accessing a peripheral, though it has these distinct modes of operation.

Reset/Abort

The SPBA has a hardware reset which initializes all registers, arbitration and peripherals rights registers (PRRs).

An abort signal input is provided allowing each master to abort its current access and release ownership (in case of master reset sequence).

Functional

Once a master request is granted, its IP Bus signals are steered to the requested peripheral.

Standby

No clock needed. The SPBA needs clocks only during access to the PRRs, arbitration, and abort phases. It generates two clock enable signals indicating when the clocks must be provided.

Configuration

During this phase, a master accesses the SPBA PRRs. The SPBA memory-mapped registers are seen as a shared peripheral.

47.2 Clocks

The table found here describes the clock sources for SPBA.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 47-1. SPBA Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

47.3 Functional description

47.3.1 Masters arbitration

The arbitration mechanism determines which port will control the master port, based on a simple round-robin arbitration scheme.

There are several use cases to consider.

- Only one master request per access. The master is switched to the shared peripheral bus, without arbitration. Figure 47-2 shows the MB request on the global module enable signal, served without wait state.
- If two masters simultaneously access SPBA, the last granted master is held off using the <master>_ips_xfr_wait output signal (default value is high). When the master is granted sips_xfr_wait, shared IP Bus peripheral is connected to <master>_ips_xfr_wait outputs.
- If three masters simultaneously access SPBA, then the last two granted masters are held off using <master>_ips_xfr_wait. Figure 47-3 shows a case in which the last two accesses granted are MA and MB. The requests are used even if they are in the same cycle.
- If after reset, at the first multiple access, no master has been granted, the priority is static: Master A (MA), Master B (MB) and last Master C (MC) port.
- No master request. No master switch to shared peripherals.

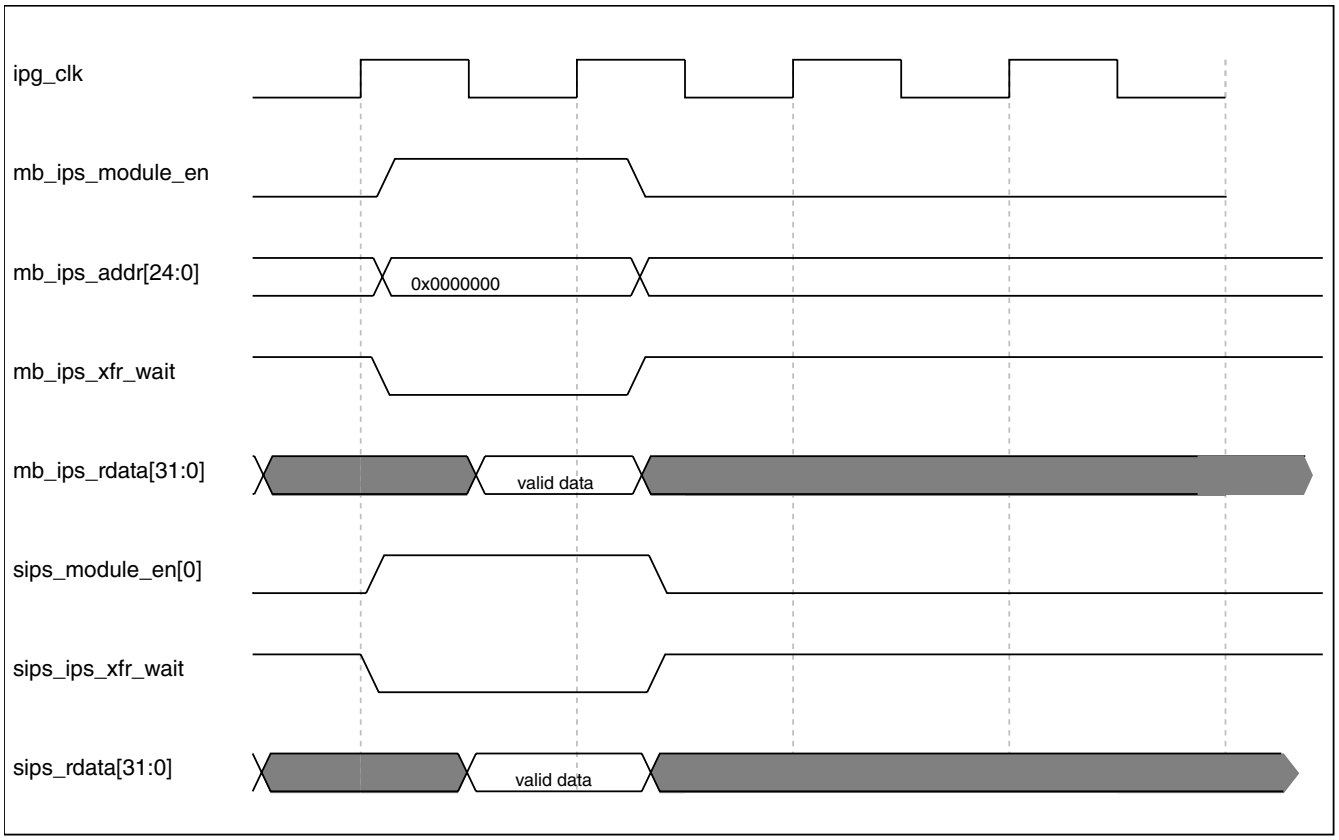


Figure 47-2. Example of one master request, no SPBA arbitration

The following figure assumes MA and MB have been the last two masters granted in the previous transfers (MA then MB).

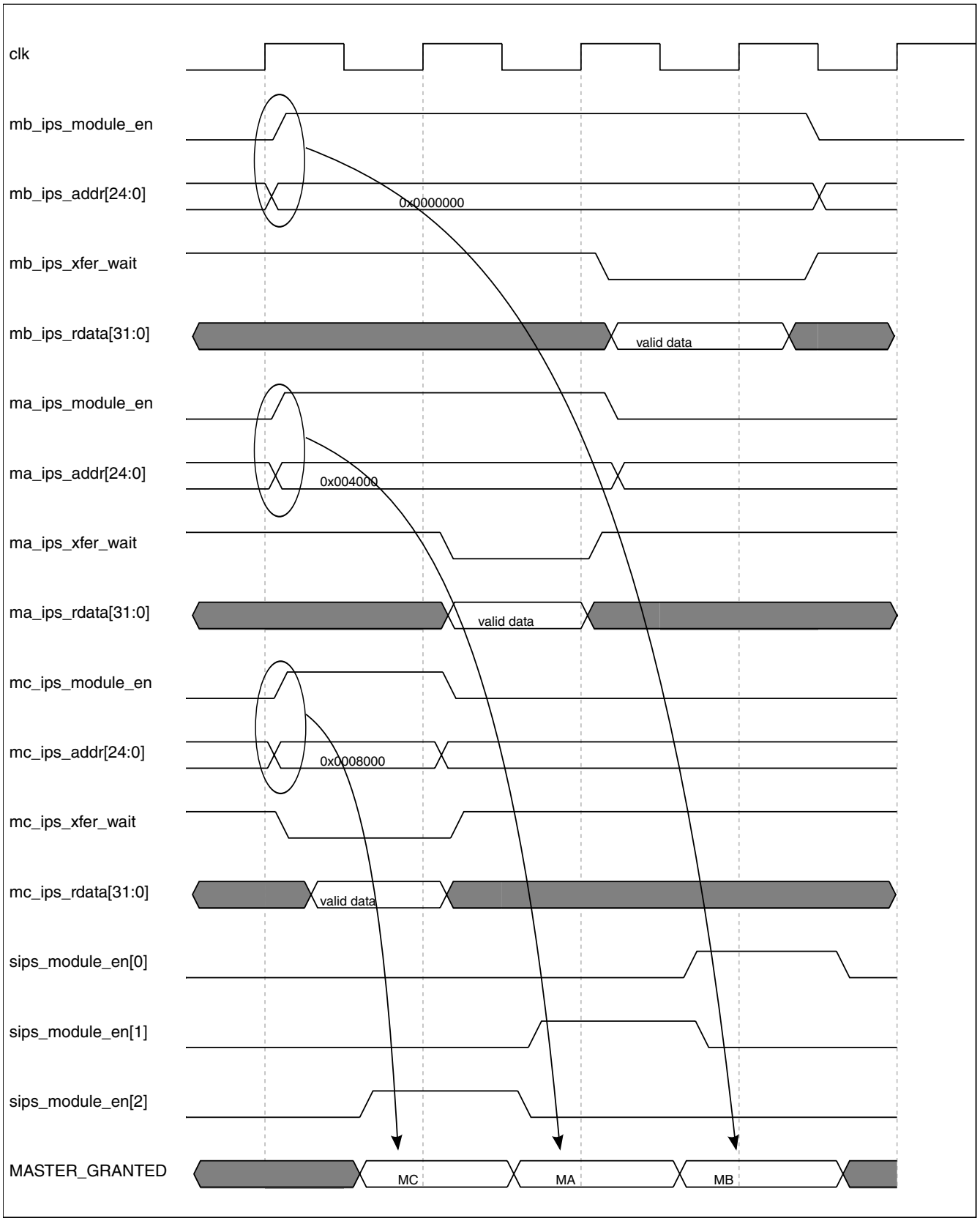


Figure 47-3. Example of three master requests: Masters already granted are "waited";

47.4 Resource ownership control

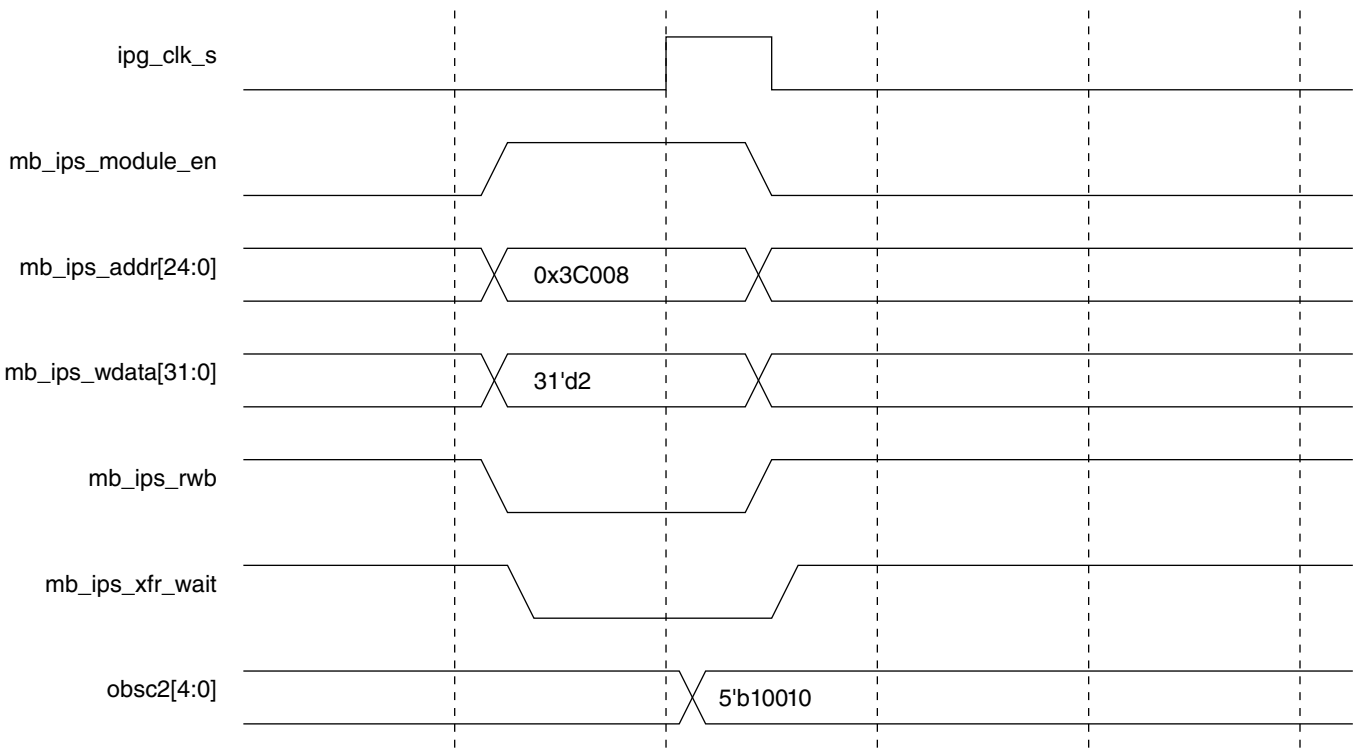
The resource ownership control regulates access to the shared peripherals and determines the steering of out-of-band signals.

47.4.1 Access control

47.4.1.1 Peripheral access

The peripheral access (resource access) of the requesting master is given by the corresponding RAR bit of the Peripheral Right Register. It determines if the master has access privilege to the resource.

Any attempt at access made by a requesting master whose access privilege bit is not set (in the PRR) is terminated with a bus error (**<master>_ips_xfr_err** is asserted by SPBA logic). The master that owns the resource can lock the peripheral for itself and/or grant other masters access to the peripheral by setting the appropriate bit(s) in the RAR field.



Master B is taking ownership of peripheral 2 by writing 3'b010 in the SPBA peripheral 2 right register (rarfield)
This ownership can be checked on obsc2 output as roi2[1:0] = 2'b10 and rar2[2:0] = 3'b010
(obsc[4:0] = {roi2[1], roi2[0], rar2[2], rar2[1], rar2[0]})

Figure 47-4. Example of one master B gaining ownership of peripheral 2

47.4.1.2 Peripheral Right Register access

The ROI bits of the Peripheral Right Register (PRR) determine which master is allowed to make write access to PRR. The identification of the requesting master is compared to the ROI bits of the PRR to determine if the master has ownership of the corresponding register.

Any attempted write access to a PRR already owned by another master will be ignored.

47.4.2 Owner election

When the peripheral is not owned by any master (ROI="00", after coming out of reset for instance), the first master to perform successfully a write to the RAR bits of the PRR is granted ownership of the peripheral and its associated PRR.

After writing to the PRR (RAR bit(s)), the master must read it back to make sure that it was granted ownership. If the RMO field is 2'b11, then the ownership claim is successful. If RMO is 2'b10, another master claimed ownership before this master was able to complete its write. This resolves the case in which two or more masters attempt to write the PRR at the same time; only the first master will be granted ownership. However all masters must read the PRR to determine if this case occurred, and if so, whether they were the first master which was granted ownership.

NOTE

A master that has been granted ownership of the PRR does not automatically have the right access to the peripheral; it must still set its own RAR bits in the PRR to access the peripheral.

47.4.3 Ending ownership

Ownership may be voluntarily ended by the owning master, or automatically upon assertion of a master-specific dead_owner signal.

The former is appropriate for software-controlled yielding of ownership. The latter is appropriate for automatic yielding of ownership when the owner has gone into reset.

When a master is reset, it clears the ROI bits of the PRRs owned by the corresponding master. When the owner is dead (in reset), all peripherals previously owned by that master must be changed to the un-owned state.

NOTE

It is the programmer's responsibility to make sure the peripherals are placed in an appropriate state before ending ownership.

47.4.3.1 Software Controlled Ownership Ending

The ROI bits will be automatically cleared when the master that owns the PRR access right clears (write) the RAR bits (Table 2).

It will then end the ownership of the PRR.

47.4.4 The Un-owned State

During the time when the peripheral is un-owned (i.e the ROI field contains all 0's), all masters have full access to it (RAR bits can then be modified by a master if ROI[1:0] = 2'b0).

In such cases it is necessary for software to ensure any necessary coherency in the resource, there is no hardware protection.

47.5 SPBA Memory Map/Register Definition

The SPBA control registers (Peripheral Right Registers) are mapped as a virtual shared peripheral.

SPBA can support up to 31 shared peripherals. Each of them has its own Peripheral Right Register (PRR) accessible within the SPBA memory-mapped registers, and consists of the Requesting Master Owner, the Resource Owner ID and the Resource Access Right fields.

SPBA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
203_C000	Peripheral Rights Register (SPBA_PRR0)	32	R/W	0000_0007h	47.5.1/3020
203_C004	Peripheral Rights Register (SPBA_PRR1)	32	R/W	0000_0007h	47.5.1/3020
203_C008	Peripheral Rights Register (SPBA_PRR2)	32	R/W	0000_0007h	47.5.1/3020
203_C00C	Peripheral Rights Register (SPBA_PRR3)	32	R/W	0000_0007h	47.5.1/3020
203_C010	Peripheral Rights Register (SPBA_PRR4)	32	R/W	0000_0007h	47.5.1/3020
203_C014	Peripheral Rights Register (SPBA_PRR5)	32	R/W	0000_0007h	47.5.1/3020
203_C018	Peripheral Rights Register (SPBA_PRR6)	32	R/W	0000_0007h	47.5.1/3020
203_C01C	Peripheral Rights Register (SPBA_PRR7)	32	R/W	0000_0007h	47.5.1/3020
203_C020	Peripheral Rights Register (SPBA_PRR8)	32	R/W	0000_0007h	47.5.1/3020

Table continues on the next page...

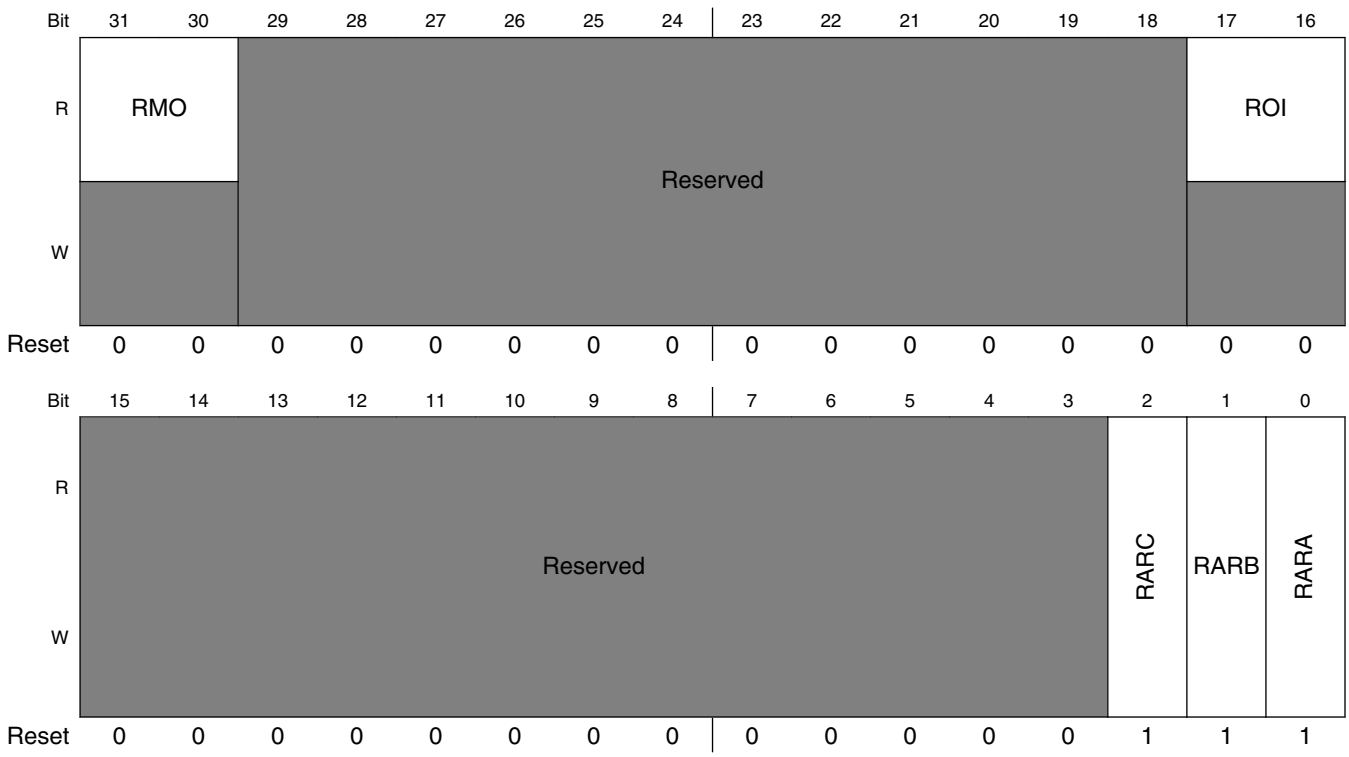
SPBA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
203_C024	Peripheral Rights Register (SPBA_PRR9)	32	R/W	0000_0007h	47.5.1/3020
203_C028	Peripheral Rights Register (SPBA_PRR10)	32	R/W	0000_0007h	47.5.1/3020
203_C02C	Peripheral Rights Register (SPBA_PRR11)	32	R/W	0000_0007h	47.5.1/3020
203_C030	Peripheral Rights Register (SPBA_PRR12)	32	R/W	0000_0007h	47.5.1/3020
203_C034	Peripheral Rights Register (SPBA_PRR13)	32	R/W	0000_0007h	47.5.1/3020
203_C038	Peripheral Rights Register (SPBA_PRR14)	32	R/W	0000_0007h	47.5.1/3020
203_C03C	Peripheral Rights Register (SPBA_PRR15)	32	R/W	0000_0007h	47.5.1/3020
203_C040	Peripheral Rights Register (SPBA_PRR16)	32	R/W	0000_0007h	47.5.1/3020
203_C044	Peripheral Rights Register (SPBA_PRR17)	32	R/W	0000_0007h	47.5.1/3020
203_C048	Peripheral Rights Register (SPBA_PRR18)	32	R/W	0000_0007h	47.5.1/3020
203_C04C	Peripheral Rights Register (SPBA_PRR19)	32	R/W	0000_0007h	47.5.1/3020
203_C050	Peripheral Rights Register (SPBA_PRR20)	32	R/W	0000_0007h	47.5.1/3020
203_C054	Peripheral Rights Register (SPBA_PRR21)	32	R/W	0000_0007h	47.5.1/3020
203_C058	Peripheral Rights Register (SPBA_PRR22)	32	R/W	0000_0007h	47.5.1/3020
203_C05C	Peripheral Rights Register (SPBA_PRR23)	32	R/W	0000_0007h	47.5.1/3020
203_C060	Peripheral Rights Register (SPBA_PRR24)	32	R/W	0000_0007h	47.5.1/3020
203_C064	Peripheral Rights Register (SPBA_PRR25)	32	R/W	0000_0007h	47.5.1/3020
203_C068	Peripheral Rights Register (SPBA_PRR26)	32	R/W	0000_0007h	47.5.1/3020
203_C06C	Peripheral Rights Register (SPBA_PRR27)	32	R/W	0000_0007h	47.5.1/3020
203_C070	Peripheral Rights Register (SPBA_PRR28)	32	R/W	0000_0007h	47.5.1/3020
203_C074	Peripheral Rights Register (SPBA_PRR29)	32	R/W	0000_0007h	47.5.1/3020
203_C078	Peripheral Rights Register (SPBA_PRR30)	32	R/W	0000_0007h	47.5.1/3020
203_C07C	Peripheral Rights Register (SPBA_PRR31)	32	R/W	0000_0007h	47.5.1/3020

47.5.1 Peripheral Rights Register (SPBA_PRR*n*)

This register controls master ownership and access for a peripheral.

Address: 203_C000h base + 0h offset + (4d × i), where i=0d to 31d



SPBA_PRR*n* field descriptions

Field	Description
31–30 RMO	Requesting Master Owner. This 2-bit register field indicates if the corresponding resource is owned by the requesting master or not. This register is reset to 2'b0 if ROI = 2'b0. 00 **UNOWNED** — The resource is unowned. 01 Reserved. 10 **ANOTHER_MASTER** — The resource is owned by another master. 11 **REQUESTING_MASTER** — The resource is owned by the requesting master.
29–18 -	This field is reserved. Reserved
17–16 ROI	Resource Owner ID. This field indicates which master (one at a time) can access to the PRR for rights modification. This is a read-only register. After reset, ROI bits are cleared ("00" -> un-owned resource).

Table continues on the next page...

SPBA_PRR*n* field descriptions (continued)

Field	Description
	A master performing a write access to the an un-owned PRR will get its ID automatically written into ROI, while modifying RARx bits. It can then read back the RMO, RAR, ROI bits to make sure RMO returns the right value, ROI bits contain its ID and RARx bits are correctly asserted. Then no other master (whom ID is different from the one stored in ROI) will be able to modify RAR fields. Owner master of a peripheral can assert its dead_owner signal, or write 1'b0 in the RARx to release the ownership (ROI[1:0] reset to 2'b0). 00 **UNOWNED** — Unowned resource. 01 **MASTER_A** — The resource is owned by master A port. 10 **MASTER_B** — The resource is owned by master B port. 11 **MASTER_C** — The resource is owned by master C port.
15–3 -	This field is reserved. Reserved
2 RARC	Resource Access Right. Control and Status bit for master C. This field indicates whether master C can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.
1 RARB	Resource Access Right. Control and Status bit for master B. This field indicates whether master B can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.
0 RARA	Resource Access Right. Control and Status bit for master A. This field indicates whether master A can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.

Chapter 48
Sony/Philips Digital Interface (SPDIF)

48.1 Introduction

The Sony/Philips Digital Interface (SPDIF) audio block is a stereo transceiver that allows the processor to receive and transmit digital audio.

The SPDIF transceiver allows the handling of both SPDIF channel status (CS) and User (U) data and includes a frequency measurement block that allows the precise measurement of an incoming sampling frequency.

A recovered clock is provided to drive both internal and external components in the system such as SAI ports, as well as external A/Ds or D/As, with clocking control provided via related registers.

As the SPDIF internal data width is 24-bit, the eight most-significant bits of all registers return zeros.

The figure below shows a block diagram of the SPDIF transceiver data paths (receiver and transmitter) and its interface.

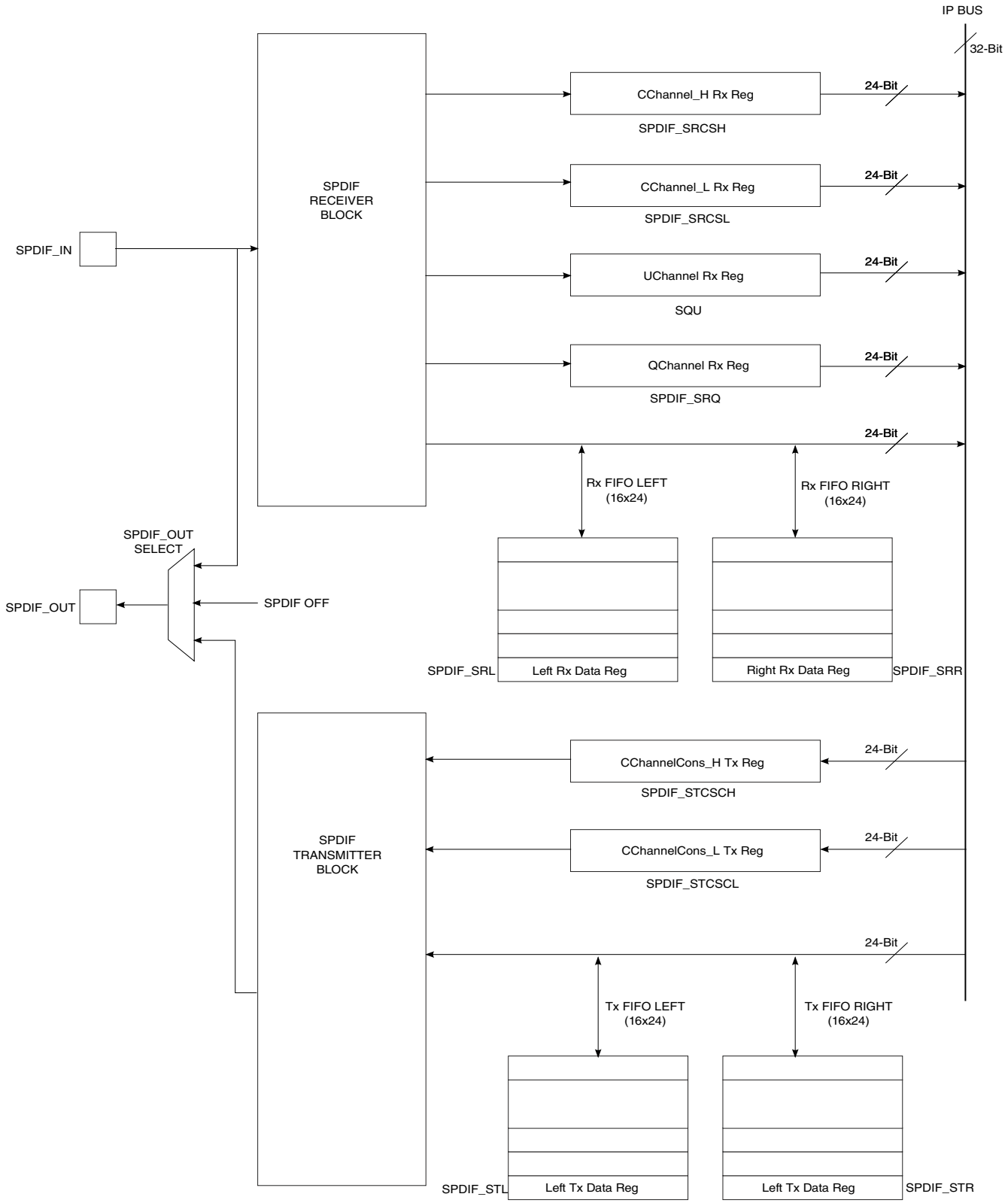


Figure 48-1. SPDIF Transceiver Data Interface Block Diagram

48.1.1 Overview

The SPDIF is composed of two parts: SPDIF Receiver and SPDIF Transmitter.

The SPDIF receiver extracts the audio data from each SPDIF frame and places the data in the SPDIF Rx left and right FIFOs. The Channel Status and User Bits are also extracted from each frame and placed in the corresponding registers. The SPDIF receiver also provides a bypass option for direct transfer of the SPDIF input signal to the SPDIF transmitter.

For the SPDIF transmitter, the audio data is provided by the processor via the SPDIFTxLeft and SPDIFTxRight registers. The Channel Status bits are also provided via the corresponding registers. The SPDIF transmitter generates a SPDIF output bitstream in the biphase mark format (IEC60958), which consists of audio data, channel status and user bits.

In the SPDIF transmitter, the IEC60958 biphase bit stream is generated on both edges of the SPDIF Transmit clock. The SPDIF Transmit clock is generated by the SPDIF internal clock generate block and the sources are from outside of the SPDIF block. For the SPDIF receiver, it can recover the SPDIF Rx clock. shows the clock structure of the SPDIF transceiver.

48.2 External Signals

The following table describes the external signals of SPDIF:

Table 48-1. SPDIF External Signals

Signal	Description	Pad	Mode	Direction
SPDIF_EXT_CLK	External clock signal	LCD_DATA07	ALT4	I
		NAND_DQS	ALT1	
SPDIF_IN	Input line	GPIO1_IO09	ALT2	I
		LCD_DATA08	ALT4	
		SD1_CLK	ALT1	
		UART1_RX_DATA	ALT6	
SPDIF_LOCK	Lock signal	LCD_DATA06	ALT4	O
SPDIF_OUT	Output line signal	GPIO1_IO08	ALT2	O
		JTAG_MOD	ALT4	
		LCD_DATA05	ALT1	
		SD1_CMD	ALT6	
		UART1_TX_DATA	ALT6	

Table continues on the next page...

Table 48-1. SPDIF External Signals (continued)

Signal	Description	Pad	Mode	Direction
SPDIF_SR_CLK	SR clock signal	LCD_DATA04	ALT4	O

48.3 Clocks

The table found here describes the clock sources for SPDIF.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 48-2. SPDIF Clocks

Clock name	Clock Root	Description
gclkw_t0	ipg_clk_root	Global clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
tx_clk	spdif0_clk_root	Module Tx clock

48.4 Functional Description

48.4.1 SPDIF Receiver

The SPDIF receiver extracts the audio data from each SPDIF frame and places the data in Rx left and right FIFOs.

The Tx left and right FIFOs are 16-deep and 24-bit-wide (equal to the audio data width). The Channel Status and User Bits are also extracted from each frame and placed in corresponding registers. The SPDIF receiver also provides a bypass option for direct transfer of the SPDIF input signal to the SPDIF transmitter.

The SPDIF receiver handles the main data audio stream and recovers the bit clock from the SPDIF input signal. The sample rate can be determined from the frequency measuring block. Additionally, the receiver supports the SPDIF C and U channels. The SPDIF C and U channel data is interfaced directly to memory-mapped registers.

All the data registers are controlled by the Interrupt Control Block and transferred to the memory-mapped IP bus.

The following functions are performed by the SPDIF receiver:

- Audio Data Reception see Audio Data Reception
- Channel Status bits Reception see Channel Status Reception
- U Channel bits Reception see User Bit Reception
- Validity Flag Reception see Validity Flag Reception
- SPDIF Receiver Exception support see SPDIF Receiver
- SPDIF Lock Detection

48.4.1.1 Audio Data Reception

The SPDIF Receiver block extracts the audio data from the IEC60958 stream, and outputs this via Rx left and right FIFOs to the memory-mapped registers SPDIFRxLeft and SPDIFRxRight.

Data from the SPDIF receiver is buffered in receive FIFO, and can be read by the processor from the memory-mapped registers.

- **SPDIF receiver data registers - Behavior on overrun, underrun**

The SPDIF Data Receive registers (SPDIFRxLeft and SPDIFRxRight) have individual FIFOs for left and right channel. As a result, there is always the possibility that left and right FIFOs may go out of sync due to FIFO underruns and FIFO overruns that affect only one part (left or right) of any FIFOs. To prevent this from happening, hardware has been added to the device. Two mechanisms to prevent mismatch between the FIFOs are available.

If a SPDIF Data Rx FIFO overrun occurs on e.g. the right half of the FIFO, the sample that caused the overrun is not written to the right half (due to overrun). Special hardware will make sure the next sample is not written to the left half of the FIFO. If the overrun occurs on the left half of the FIFO, the next sample is not written to the right half of the FIFO.

- **SPDIF receiver data registers - Automatic resynchronization of FIFOs**

An automatic FIFO resynchronization feature is available. It can be enabled and disabled separately for every FIFO. If it is enabled, the hardware will check to see if the left and right FIFOs are in sync. If that is not the case, it will set the filling pointer of the right FIFO to be equal to the filling pointer of the left FIFO.

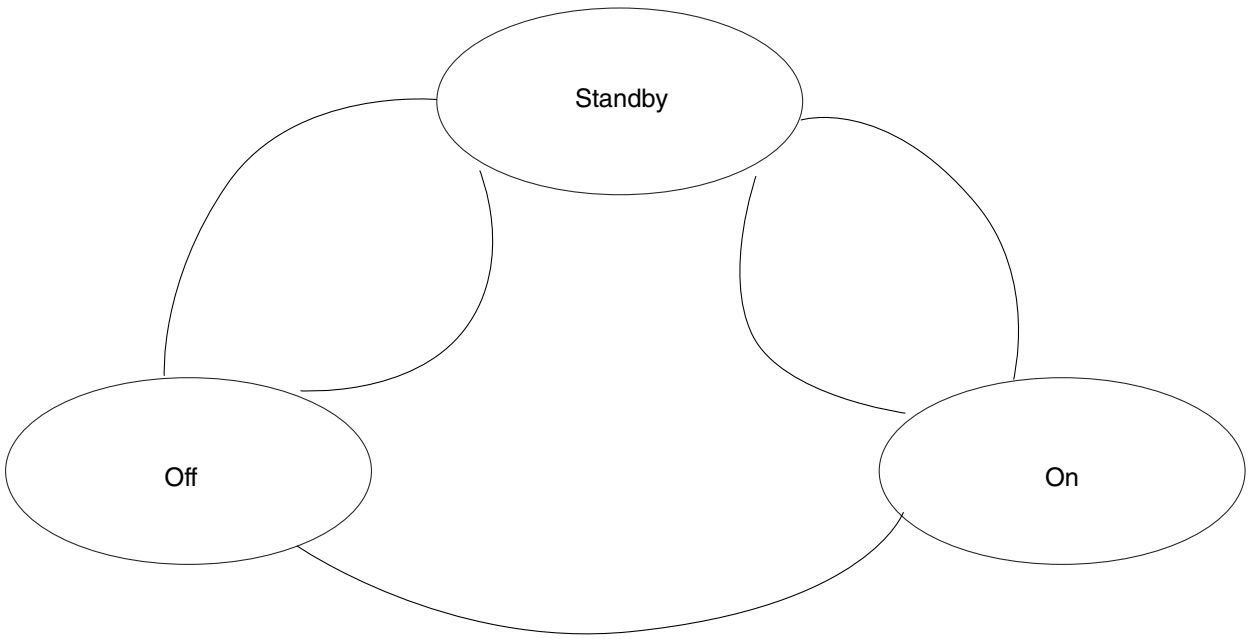


Figure 48-2. FIFO Auto-resync Controller State Machine

The operation is explained from the state diagram shown above. Every FIFO auto-resync controller has a state machine with 3 states: Off, StandBy and On. In the On state, the filling of the left FIFO is compared with the filling of right, and if they are not equal, right is made equal to left, and an interrupt is generated.

The controller will stay in Off state when the feature is disabled. When not disabled, the state machine will go to Off state on any processor read or write to the FIFO. It will go from On or Off to Standby on any left sample read from SPDIF Tx FIFOs, or on any left sample write to SPDIF Rx FIFOs. The controller will go from Standby to On on any right sample read from SPDIF Tx FIFO, or on any right sample write to SPDIF Rx FIFO. There is a control bit in the SPDIFConfig register to enable/disable the feature for the SPDIF Rx FIFO and SPDIF Tx FIFO.

48.4.1.1.1 Application Note

The automatic FIFO resynchronization can be switched on, and will avoid all mismatches between left and right FIFOs, if the software obeys the following rules: 1. When the left data is read or written to the left FIFO, in the same place of the program, data must be read or written to the right FIFO. Maximum time difference between left and right is 1/2 sample clock. (E.g. if sample frequency is 44 Khz, approximately 10 micro-seconds. For 88 Khz, approximately 5 micro-seconds.) 2. Write/read data to FIFO s at least 2 samples

at the time. If there is a mismatch Left-Right, the resync logic may go on only 1 sample clock after last data is read/written to the FIFO. Also acceptable is polling the FIFO, if at least part of the time 2 samples will be read/written to it.

- **SPDIF receiver - Additional features**

There are three exceptions associated with the SPDIF Receivers FIFOs

- full
- under/overrun
- resync

When the "full" condition is set for processor data input registers, the processor should read data from the FIFO, before overrun occurs. When "full" is set, and the FIFO contains e.g. 6 samples, it is acceptable for the software to read first 6 samples from the LEFT address, followed by 6 samples from the RIGHT address, or 6 samples from the RIGHT address, followed by 6 samples from the LEFT address, or 1 sample LEFT, followed by 1 sample RIGHT repeated 6 times. There is no order specified.

The implementation for SPDIF Rx is a double FIFO, one for left and one for right. "full" is set when both FIFOs are full. "underrun, overrun" are set when one of the FIFOs do underrun or do overrun. The resync interrupt means hardware took special action to resynchronize left and right FIFOs.

The FIFO level at which the "full" interrupt is generated, is programmable via the Full Select field in the SPDIFConfigReg register.

Rx FIFO on and Rx FIFO reset.

Two additional control fields of the SPDIF Rx FIFO are the on/off select and FIFO reset fields.

If on/off select is set to off, all-zero will be read from the FIFO, irrespective of the data received over the SPDIF interface.

If FIFO reset is set, the FIFO is blocked at "1 sample in FIFO". In this, the full interrupt will be on if FullSelect is set to "00". If FullSelect is set to any other value, interrupt will be off. The other interrupts are always off.

48.4.1.2 Channel Status Reception

A total of 48 channel status bits are received in two registers. No interpretation is performed by the SPDIF receiver block.

Channel Status Bits are ordered first bit left. CS-channel MSB bit "0" is located in bit position 23 in the memory-mapped register SPDIFRxCChannel_h. CS-channel bit "23" is considered the LSB bit 0 in the register. C-channel bit 24 to 47 is seen as [23:0] bits of register SPDIFRxCChannel_l.

48.4.1.2.1 Channel Status Interrupt

When the value of a new SPDIF "CS" channel status frame is loaded in the register, an interrupt is generated. The interrupt is cleared when the processor writes the corresponding bit in the InterruptStat register.

48.4.1.3 User Bit Reception

There are two modes for U Channel reception, CD and non-CD. As is decided by USyncMode (bit 1 of CDText_Control register).

- **Behavior of U Channel receive interface on incoming CD U Channel Sub-code in SPDIF receiver**.

This mode is selected if UsyncMode, bit 1 in register CD Text control is set "1".

The CD sub-code stream embedded into the SPDIF U channel consists of a sequence of packets. Every packet is made up 98 "symbols". The first two symbols of every packet are "sync symbols", the other 96 symbols are "data symbols".

Any sequence found in the SPDIF U channel stream starting with a leading one, followed by 7 information bits, is recognized as a "data symbol". Subsequent data symbols are separated by "pauses". During the "pause", "zero bits" are seen on the SPDIF U channel.

Data symbols are coming in MSB first. The MSB is the leading one.

When a "long pause" is seen between 2 subsequent "data symbols", the SPDIF receiver will assume the reception of one or more "sync symbols". Table below gives details.

Table 48-3. Sync Control Bits

Number of U Channel zero bits	Corresponding number of sync symbols
0-1	Unpredictable, not allowed
2-10	0
11-22	1
23-34	2
35-46	3
>45	Unpredictable, not allowed

The recognition of the number of sync symbols derives from the fact that the U channel transmitter in the CD channel decoder will transmit one symbol on average every 12 SPDIF channel bits. On this average rate, there is a maximum tolerance of 5%.

The SPDIF receiver is tolerant of symbol errors. Due to the physical nature of the transmission of the data over the CD disc, not more than 1 out of any 5 consecutive user channel symbols may be in error. The error may cause a change in data value, which is not detected by this interface, or it may cause a data symbol to be seen as a sync symbol, or a sync symbol to be seen as a data symbol. However, not more than 1 out of any 5 consecutive user channel symbols should be affected in this way.

The SPDIF U channel circuitry recognizes the 98-symbol packet structure, and sends the 96 symbol payload to the processor application. The 96 symbol payload is transmitted to the processor via 2 registers:

- The SPDIFRxUChannel register. In this register, data is presented 3 symbols at the time to the processor. Every time 3 new valid symbols, received on the SPDIF U Channel are present, the UChannelRxFull interrupt is asserted. For one 98-symbol packet, 96 symbols are carried across SPDIFRxUChannel. To transfer all this data, 32 UChannelRxFull interrupts are generated.
- The QChannelReceive register. In this register, only the Q bit of the packet is accumulated.Operation is similar to UChannelReceive. Because only Q-bit is transferred, only 96 Q-bits are transferred for any 98-symbol packet. To transfer this data, 4 QChannelRxFull interrupts are generated. When QChannelRxFull occurs, it is coincident with UChannelRxFull. There is only one QChannelRxFull for every 8 UChannelRxFull. The convention is that most significant data is transmitted first, and is left-aligned in the registers.
- Timing regarding packet boundary is extracted by hardware. The last UChannelRxFull corresponding to a given packet should be coincident with the last QChannelRxFull. In this last U, Q channel interrupt, symbols 95-98 are received, Q channel bits 67-98. The interrupts are coincident with UQSyncFound, flagging last symbols of the current frame.
- When the start of the new packet is found before the current packet is complete (less than 98 symbols in the packet), the UQFrameError interrupt is set. The application software should read out UChannelReceive and QchannelReceive registers, discard the value, and assume the start of a new packet.
- As already said, packet sync extraction is tolerant for single-symbol errors. Packet sync detection is based on the recognition of the sequence data-sync-sync-data in the symbol stream, because this is the only syncing sequence that is not affected by single errors. If the sync symbols are not found 98 symbols after the previous

occurrence, it is assumed to be destroyed by channel error, and a new sync symbols is interpolated.
- Normally, only data bytes are passed to the application software. Every databyte will have its most significant bit set. If sync symbols are passed to the application software, they are seen as all-zero symbols. Sync symbols can only end up in the data stream due to channel error.

- **Behavior of U Channel receive interface on incoming non-CD data**.

This mode is selected if UsyncMode, bit 1 in register CD Text control is set'0'.

In non-CD mode, the SPDIF U channel stream is recognized as a sequence of "data symbols". No packet recognition is done.

Any sequence found in the SPDIF U channel stream starting with a leading one, followed by 7 information bits, is recognized as a "data symbol". Subsequent data symbols are separated by "pauses". During the "pause", "zero bits" are seen on the SPDIF U channel.

3 consecutive data symbols seen in the SPDIF U Channel stream are grouped together into the SPDIFRxUChannel register. First symbol is left, last symbol is right aligned. When SPDIFRxUChannel contains 3 new data symbols, UChannelRxFull is asserted.

In this mode, the operation of QchannelRx and associated interrupt QchannelRxFull is reserved, undefined. And the operation of UQFrameError and UQSyncFound is also reserved, undefined.

The U channel is extracted, and output by the SPDIF Rx on SPDIFRxUChannel-Stream.

When incoming SPDIF data parity error or bit error is detected, and if the next SPDIF word for that channel is error-free, the SPDIF word in error is replaced with the average of the previous word and next word. When incoming SPDIF data parity error or bit error is detected, and the next SPDIF word is in error, the previous SPDIF word is repeated.

48.4.1.4 Validity Flag Reception

An interrupt is associated with the Validity flag. (interrupt 16 - SPDIFValNoGood). This interrupt is set every time a frame is seen on the SPDIF interface with the validity bit set to "invalid".

48.4.1.5 SPDIF Receiver Interrupt Exception Definition

Several SPDIF exceptions can trigger an interrupt.

They are:

- Control Status channel change. Set when SPDIFRxCChannel_l register is updated. The register is updated for every new C-Channel received. The exception is reset on write to InterruptClear register.
- SPDIF Illegal Symbol. Set on reception of illegal symbol during SPDIF receive. Reset by writing register InterruptClear.[1]
- SPDIF bit error. Set on reception of bit error. (Parity bit does not match). Reset on write to InterruptClear register.
- Receive data FIFO full. Set when SPDIF receive data FIFO is full.
- Receive data FIFO underrun/overrun. Set when there is a underrun/overrun on the SPDIF receive data FIFO.
- Receive data FIFO resynchronization. Set when a resynchronization event occurs on the SPDIF receive data FIFO.
- Receive U Channel buffer full. Set when next 24 bits of U channel code are available.
- Receive Q Channel buffer overrun. Set when Q channel buffer overrun.
- Receive U Channel buffer overrun. Set on U channel buffer overrun.
- Receive Q Channel buffer full. Set when next 24 bits of Q channel code are available.
- Receive UQ sync found. Set when UQ channel sync found.
- Receive UQ frame error. Set when UQ frame error found.

48.4.1.6 Standards Compliance

The SPDIF interface is compatible with the Tech 3250-E standard of the European Broadcasting Union, except clause 6.3.3 and the IEC60958-3 Ed2 for relevant topics.

Supported input frequency range is 12 Khz up to 96 Khz. (fully compliant) and 96 Khz up to 176 Khz (Can interface with compliant SPDIF transmitter within same cabinet, making reasonable assumptions on jitter added due to interconnecting wire.)

Tolerated jitter on SPDIF input signals are 0.25 bit peak-peak for high frequencies. There is no jitter limit for low frequencies. The user channel extraction in CD mode is capable of coping with single-symbol errors, and still retrieve U channel frames on correct boundaries. This capability is required for reliable reception of CD-Text from some Philips CD channel decoders. This capability was deemed more important than compliance with the IEC60958 annex A.3 standard, and for this reason user channel

1. The SPDIF input is a biphase/mark modulated signal. The time between any two successive transitions of the SPDIF signal is always 1, 2 or 3 SPDIF symbol periods long. The SPDIF receiver will parse the stream, and split it in so-called symbols. It recognizes s1, s2 and s3 symbols, depending on the length of the symbols. Not all sequences of these symbols are allowed. To give an example, a sequence s2-s1-s1-s1-s2 cannot occur in a no-error SPDIF signal. If the receiver finds such an illegal sequence, the illegal symbol interrupt is set. No corrective action is undertaken. When the interrupt occurs, this means that(a) The SPDIF signal is destroyed by noise (b) The SPDIF frequency changed.

reception is not compliant with IEC60958 annex A.3. However, the interface is capable to receive U channel inserted by a typical CD channel decoder. Also, in this case, it is more robust and tolerant for channel error than what is required by IEC60958 annex A.3.

48.4.1.7 SPDIF PLOCK Detection and Rxclk Output

Using the high speed system clock, the internal DPLL can extract the bit clock (advanced pulse) from the input bitstream. When this internal DPLL is locked, the LOCK bit of PhaseConfig Register will be set, and the SPDIF Lock output pin SPDIF_LOCK will be asserted.

After DPLL has locked, the pulses are generated, and the average pulse rate is 128 x the sampling frequency.(For a 44.1 Khz input sampling frequency, the average pulse rate = 128 x 44.1 Khz.) The pulse signal is used in the FreqMeas circuit to generate the frequency measurement result.

48.4.1.8 Measuring Frequency of SPDIF_RxClk

The internal DPLL can extract the bit clock (advanced plus) from the input bitstream. To do that, it is necessary to measure the frequency of the incoming signal in relationship with the system clock (BUS_CLK).

Associated with it are two registers, PhaseConfig and FreqMeas. The circuit will measure the frequency of the incoming clock as a function of the BUS_CLK. The circuit is a second-order filter. The output is a value represented by an unsigned number stored in the 24-bit FreqMeas register, giving the frequency of the source as a function of the BUS_CLK.

FreqMeas[23:0] = FreqMeas_CLK / BUS_CLK * 2^{10} * GAIN.

For example, if the GAIN is selected as 8*(2**10) (PhaseConfig[5:3] = 3'b011), the actual result

FreqMeas_CLK / BUS_CLK is equal to FreqMeas[23:0] / 2^{23}.

48.4.2 SPDIF Transmitter

Audio data for the SPDIF transmitter is provided by processor via the SPDIFTxLeft and SPDIFTxRight registers.

Clocking for SPDIF transmitter is selected through a multiplexer from several clock sources (see TxClk_Source for clock source inputs). The SPDIF transmitter clock source can be divided down as needed using Txclk_DF. The SPDIF transmitter output can be chosen from either the SPDIF transmitter block, directly from the SPDIF receiver (via the output multiplexer), or disabled.

The SPDIF transmitter generates a SPDIF output bitstream in IEC60958 biphase mark format, consisting of audio data, channel status.

48.4.2.1 Audio Data Transmission

Audio data for the SPDIF transmitter is provided by the processor via SPDIFTxLeft and SPDIFTxRight registers. They send audio data to Tx left and right FIFOs. The Tx left and right FIFOs are also 16-deep and 24-width (equal to the audio data width).

- **SPDIF transmitter data registers - Behavior on overrun, underrun**

The SPDIF Data Transmit registers (SPDIFTxLeft and SPDIFTxRight) have individual FIFOs for left and right channel. As a result, there is always the possibility that left and right FIFOs may go out of sync due to FIFO underruns and FIFO overruns that affect only one part (left or right) of any FIFO. To prevent this from happening, hardware has been added on the device. Two mechanisms to prevent mismatch between the FIFOs are available.

If SPDIF Tx FIFO underruns on the right half of the FIFO, no sample leaves that FIFO (because it was already empty). Special hardware will make sure that the next sample read from the left FIFO will not leave the FIFO (no read strobe is generated). If the underrun occurs on the left half of the FIFO, next read strobe to the right FIFO is blocked.

- **SPDIF transmitter data registers - Automatic resynchronization of FIFOs**

See Audio Data Reception.

- **SPDIFTxLeft, SPDIFTxRight details**

With SPDIF Tx FIFOs three exceptions are associated.

- empty
- under/overrun
- resync

When the empty condition is set for processor data output registers, the processor should write data to the FIFO, before underrun occurs. When empty is set and, for instance, 6 samples need to be written, it is acceptable for the software to write first 6 samples from

the LEFT address, followed by 6 samples from the RIGHT address, or 1 sample LEFT, followed by 1 sample RIGHT repeated 6 times. Left should be written before right. The implementation of all data out FIFOs is a double FIFO, one for left and one for right. Empty is set when both FIFOs are empty. Underrun, overrun are set when one of the FIFOs do underrun or do overrun. Resync is set when the hardware resynchronizes left and right FIFOs.

On receiving underrun, overrun interrupt, synchronization between Left and Right words in the FIFOs may be lost. Synchronization will not be lost when the underrun or overrun comes from the IEC60958 side of the FIFO. If the processor reads or writes more data from, for example, left than from right, synchronization will be lost. If automatic resynchronization is enabled, and if the software obeys the rules to let this work, resynchronization will be automatic.

48.4.2.2 Channel Status Transmission

A total of 48 Consumer channel status bits are transmitted from two registers. Channel Status Bits are ordered first bit left.

CS-channel MSB bit "0" is located in bit position 23 in the memory-mapped register SPDIFTxCChannelCons_h. CS-channel bit "23" is considered bit 0 in the register. C-channel bits 24-47 are seen as MSB-LSB bits of register SPDIFTxCChannelCons_l.

48.4.2.3 Validity Flag Transmission

The validity bit setting is performed via bit 5 of the SPDIF_SCR register.

48.5 SPDIF Memory Map/Register Definition

SPDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200_4000	SPDIF Configuration Register (SPDIF_SCR)	32	R/W	0000_0400h	48.5.1/3038
200_4004	CDText Control Register (SPDIF_SRCD)	32	R/W	0000_0000h	48.5.2/3040
200_4008	PhaseConfig Register (SPDIF_SRPC)	32	R/W	0000_0000h	48.5.3/3041
200_400C	InterruptEn Register (SPDIF_SIE)	32	R/W	0000_0000h	48.5.4/3042
200_4010	InterruptStat Register (SPDIF_SIS)	32	R	0000_0002h	48.5.5/3044
200_4010	InterruptClear Register (SPDIF_SIC)	32	W	0000_0000h	48.5.6/3046

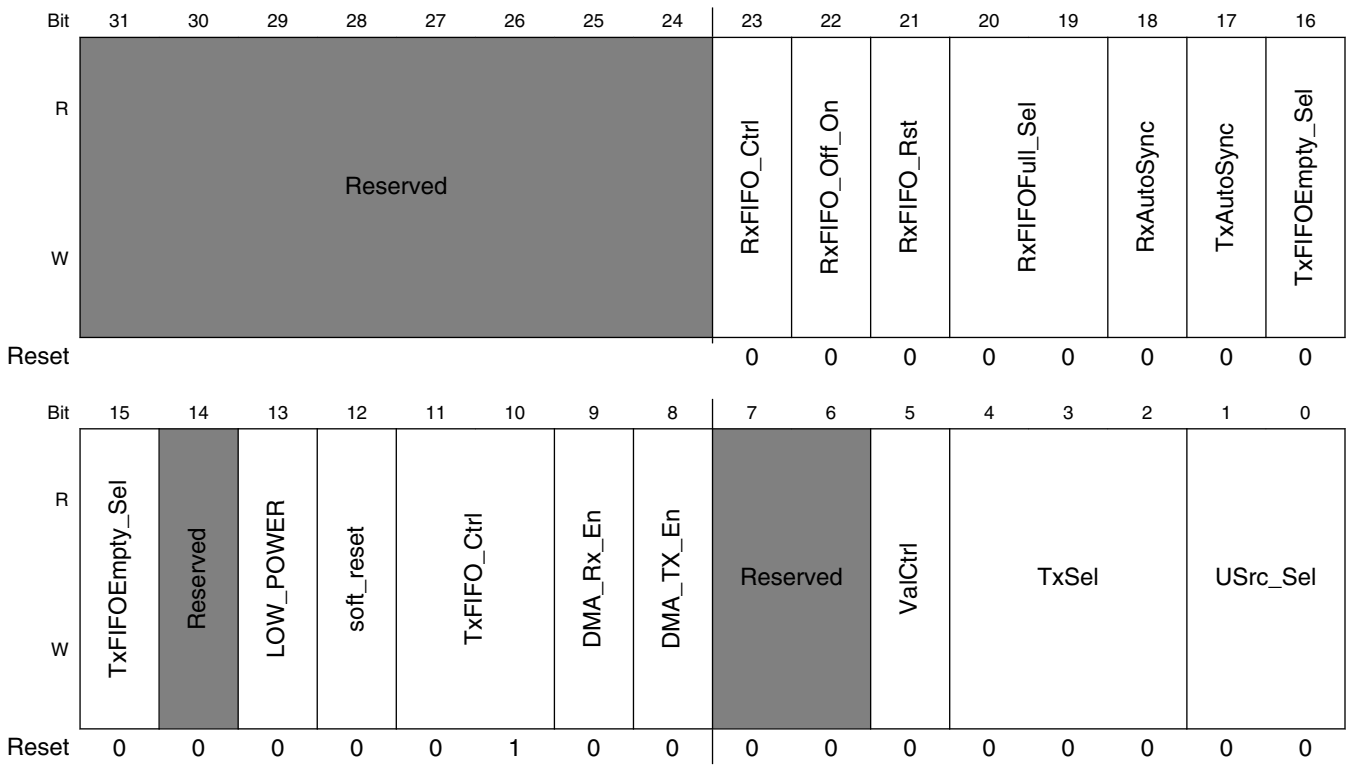
Table continues on the next page...

SPDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200_4014	SPDIFRxLeft Register (SPDIF_SRL)	32	R	0000_0000h	48.5.7/3047
200_4018	SPDIFRxRight Register (SPDIF_SRR)	32	R	0000_0000h	48.5.8/3048
200_401C	SPDIFRxCChannel_h Register (SPDIF_SRCSH)	32	R	0000_0000h	48.5.9/3048
200_4020	SPDIFRxCChannel_l Register (SPDIF_SRCSL)	32	R	0000_0000h	48.5.10/ 3049
200_4024	UchannelRx Register (SPDIF_SRU)	32	R	0000_0000h	48.5.11/ 3049
200_4028	QchannelRx Register (SPDIF_SRQ)	32	R	0000_0000h	48.5.12/ 3050
200_402C	SPDIFTxLeft Register (SPDIF_STL)	32	W	0000_0000h	48.5.13/ 3050
200_4030	SPDIFTxRight Register (SPDIF_STR)	32	W	0000_0000h	48.5.14/ 3051
200_4034	SPDIFTxCChannelCons_h Register (SPDIF_STCSCH)	32	R/W	0000_0000h	48.5.15/ 3051
200_4038	SPDIFTxCChannelCons_l Register (SPDIF_STCSCL)	32	R/W	0000_0000h	48.5.16/ 3052
200_4044	FreqMeas Register (SPDIF_SRFM)	32	R	0000_0000h	48.5.17/ 3052
200_4050	SPDIFTxClk Register (SPDIF_STC)	32	R/W	0002_0F00h	48.5.18/ 3053

48.5.1 SPDIF Configuration Register (SPDIF_SCR)

Address: 200_4000h base + 0h offset = 200_4000h



SPDIF_SCR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 RxFIFO_Ctrl	0 Normal operation 1 Always read zero from Rx data register
22 RxFIFO_Off_On	0 SPDIF Rx FIFO is on 1 SPDIF Rx FIFO is off. Does not accept data from interface
21 RxFIFO_Rst	0 Normal operation 1 Reset register to 1 sample remaining
20–19 RxFIFOFull_Sel	00 Full interrupt if at least 1 sample in Rx left and right FIFOs 01 Full interrupt if at least 4 sample in Rx left and right FIFOs 10 Full interrupt if at least 8 sample in Rx left and right FIFOs 11 Full interrupt if at least 16 sample in Rx left and right FIFO
18 RxAutoSync	0 Rx FIFO auto sync off 1 RxFIFO auto sync on
17 TxAutoSync	0 Tx FIFO auto sync off 1 Tx FIFO auto sync on

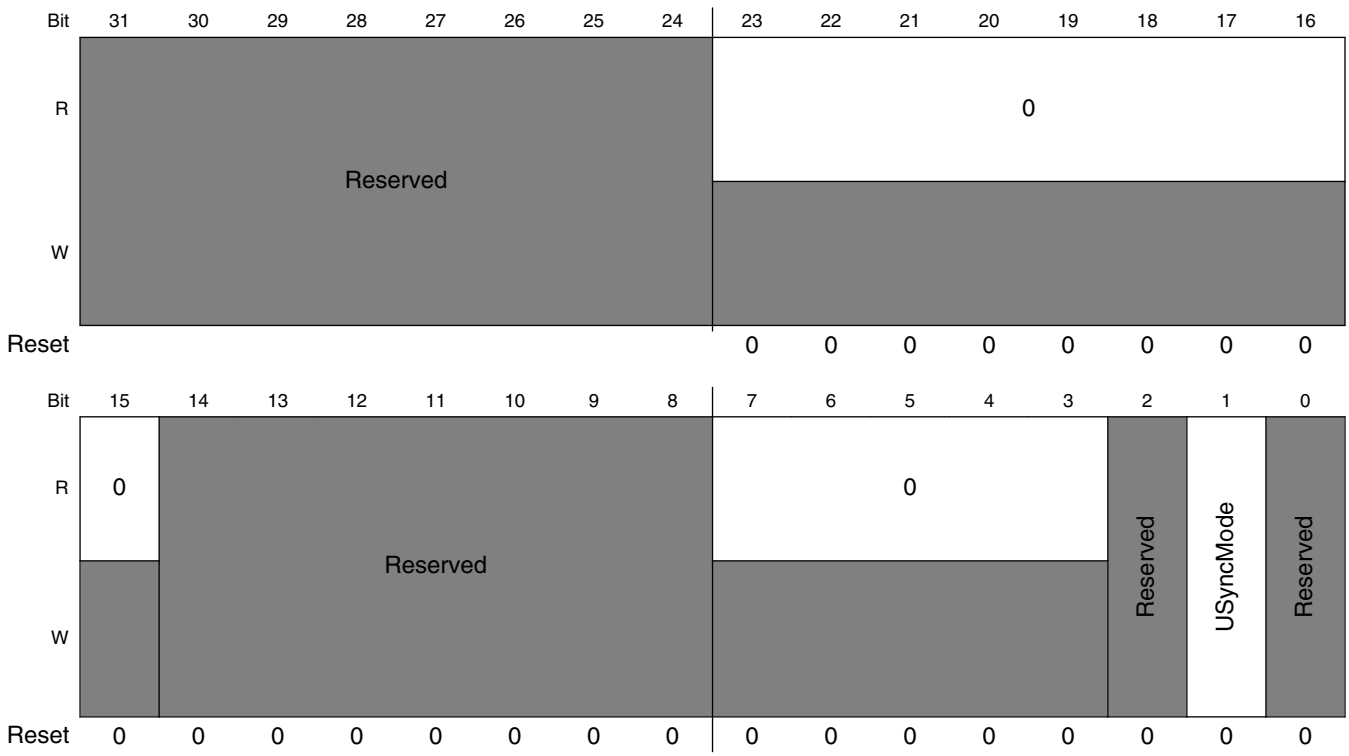
Table continues on the next page...

SPDIF_SCR field descriptions (continued)

Field	Description
16–15 TxFIFOEmpty_Sel	00 Empty interrupt if 0 sample in Tx left and right FIFOs 01 Empty interrupt if at most 4 sample in Tx left and right FIFOs 10 Empty interrupt if at most 8 sample in Tx left and right FIFOs 11 Empty interrupt if at most 12 sample in Tx left and right FIFOs
14 -	This field is reserved. Reserved
13 LOW_POWER	When write 1 to this bit, it will cause SPDIF enter low-power mode. return 1 when SPDIF in Low-Power mode.
12 soft_reset	When write 1 to this bit, it will cause SPDIF software reset. The software reset will last 8 cycles. When in the reset process, return 1 when read. else return 0 when read.
11–10 TxFIFO_Ctrl	00 Send out digital zero on SPDIF Tx 01 Tx Normal operation 10 Reset to 1 sample remaining 11 Reserved
9 DMA_Rx_En	DMA Receive Request Enable (RX FIFO full)
8 DMA_TX_En	DMA Transmit Request Enable (Tx FIFO empty)
7–6 -	This field is reserved. Reserved
5 ValCtrl	0 Outgoing Validity always set 1 Outgoing Validity always clear
4–2 TxSel	000 Off and output 0 001 Feed-through SPDIFIN 101 Tx Normal operation Others Reserved
USrc_Sel	00 No embedded U channel 01 U channel from SPDIF receive block (CD mode) 10 Reserved 11 U channel from on chip transmitter

48.5.2 CDText Control Register (SPDIF_SRCD)

Address: 200_4000h base + 4h offset = 200_4004h

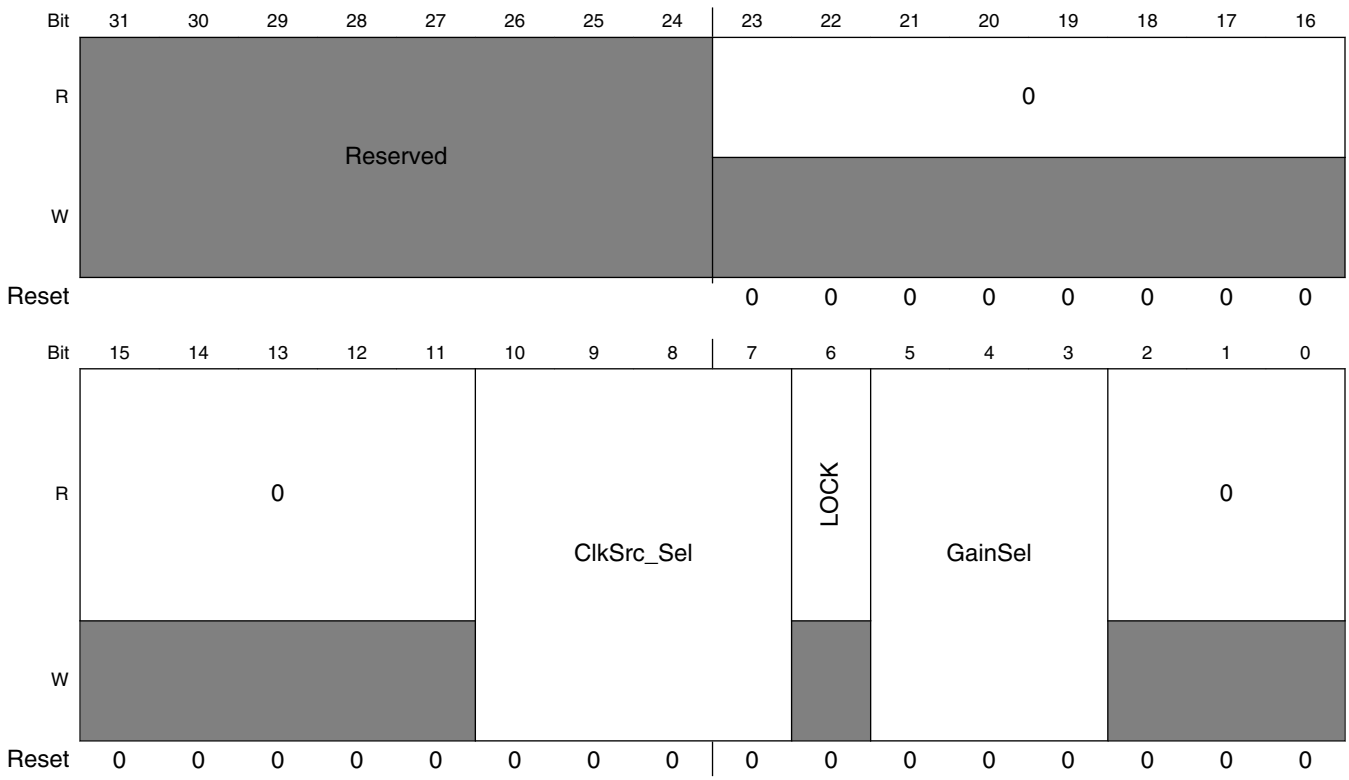


SPDIF_SRCD field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–15 Reserved	This read-only field is reserved and always has the value 0.
14–8 -	This field is reserved. Reserved. set to zero.
7–3 Reserved	This read-only field is reserved and always has the value 0.
2 -	This field is reserved. Reserved.
1 USyncMode	0 Non-CD data 1 CD user channel subcode
0 -	This field is reserved. Reserved.

48.5.3 PhaseConfig Register (SPDIF_SRPC)

Address: 200_4000h base + 8h offset = 200_4008h



SPDIF_SRPC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–11 Reserved	This read-only field is reserved and always has the value 0.
10–7 ClkSrc_Sel	Clock source selection, all other settings not shown are reserved: 0000 if (DPLL Locked) SPDIF_RxClk else REF_CLK_32K (XTALOSC) 0001 if (DPLL Locked) SPDIF_RxClk else tx_clk (SPDIF0_CLK_ROOT) 0011 if (DPLL Locked) SPDIF_RxClk else SPDIF_EXT_CLK 0101 REF_CLK_32K (XTALOSC) 0110 tx_clk (SPDIF0_CLK_ROOT) 1000 SPDIF_EXT_CLK
6 LOCK	LOCK bit to show that the internal DPLL is locked, read only
5–3 GainSel	Gain selection: 000 24*(2**10) 001 16*(2**10)

Table continues on the next page...

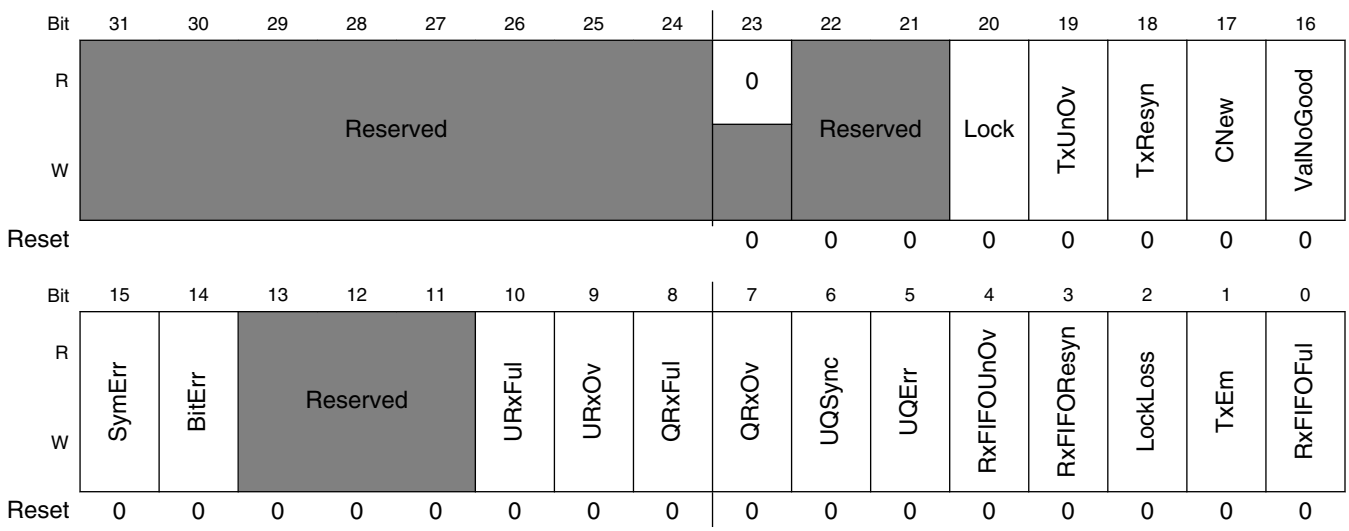
SPDIF_SRPC field descriptions (continued)

Field	Description
	010 12*(2**10) 011 8*(2**10) 100 6*(2**10) 101 4*(2**10) 110 3*(2**10)
Reserved	This read-only field is reserved and always has the value 0.

48.5.4 InterruptEn Register (SPDIF_SIE)

The InterruptEn register (SPDIF_SIE) provides control over the enabling of interrupts.

Address: 200_4000h base + Ch offset = 200_400Ch



SPDIF_SIE field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 Reserved	This read-only field is reserved and always has the value 0.
22–21 -	This field is reserved. Reserved. set to zero.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun

Table continues on the next page...

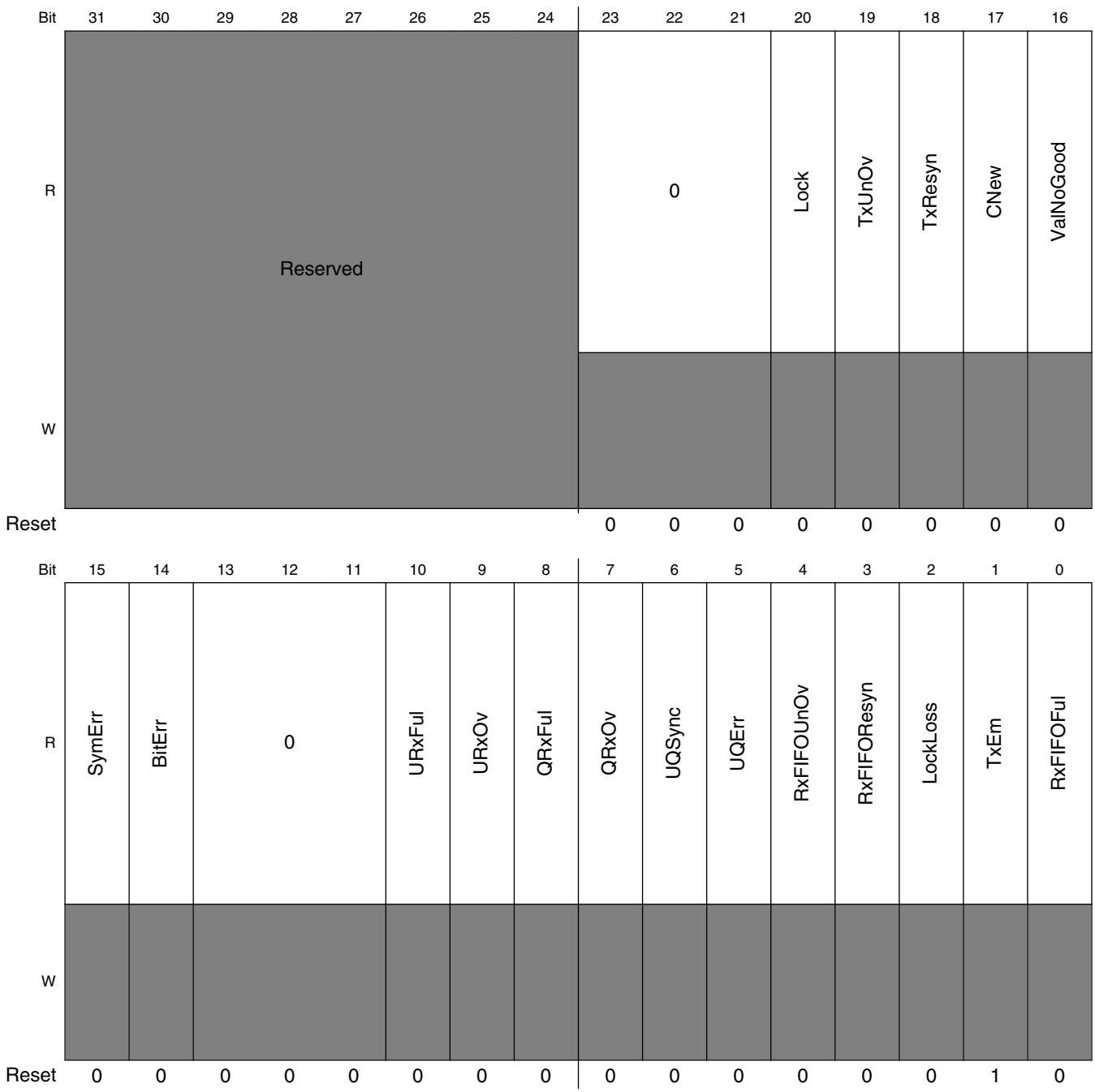
SPDIF_SIE field descriptions (continued)

Field	Description
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–11 -	This field is reserved. Reserved. set to zero.
10 URxFul	U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	U Channel receive register overrun
8 QRxFul	Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
1 TxEm	SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

48.5.5 InterruptStat Register (SPDIF_SIS)

The InterruptStat (SPDIF_SIS) register is a read only register that provides the status on interrupt operations.

Address: 200_4000h base + 10h offset = 200_4010h



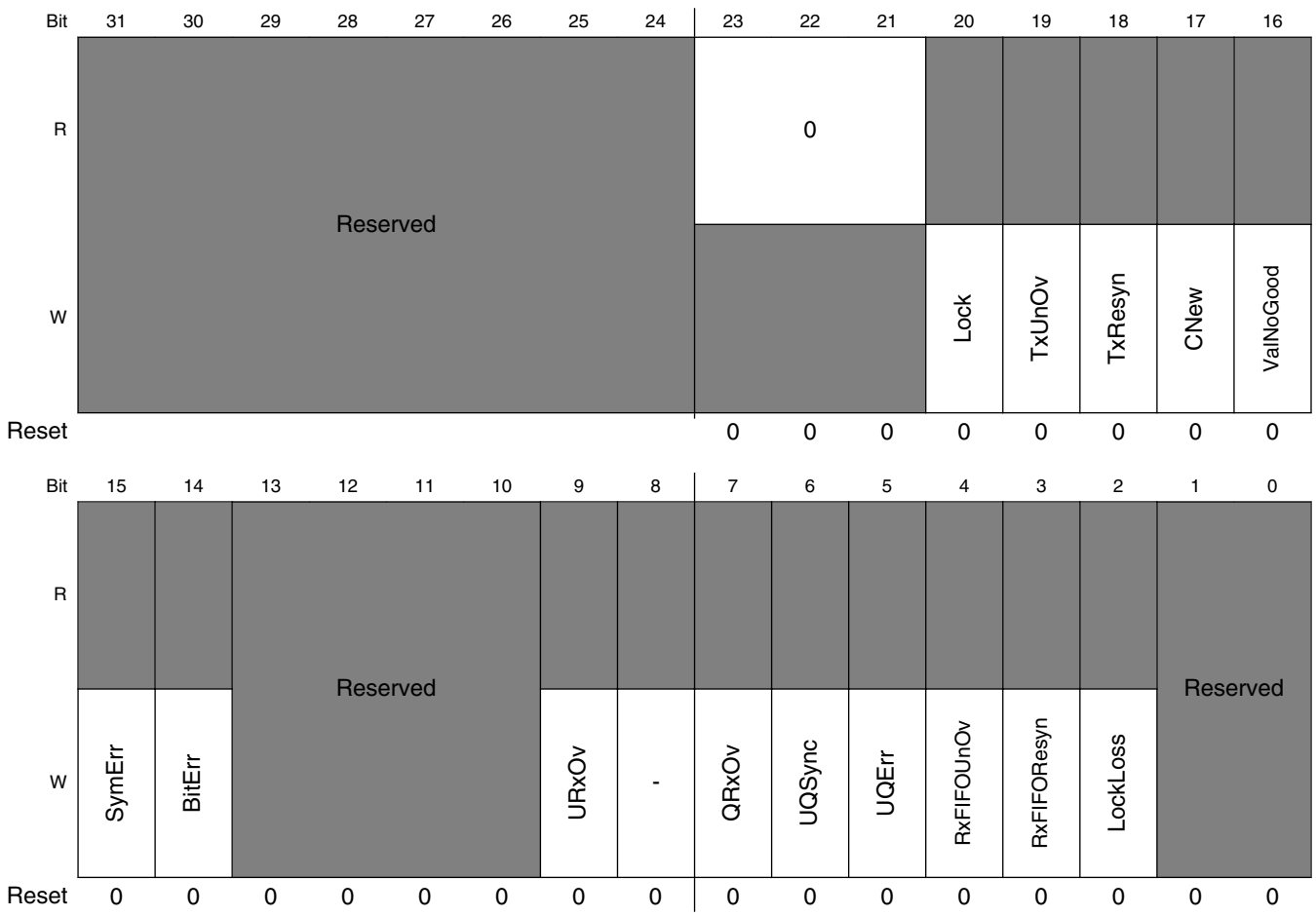
SPDIF_SIS field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–21 Reserved	This read-only field is reserved and always has the value 0.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–11 Reserved	This read-only field is reserved and always has the value 0.
10 URxFul	U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	U Channel receive register overrun
8 QRxFul	Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
1 TxEm	SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

48.5.6 InterruptClear Register (SPDIF_SIC)

The InterruptClear (SPDIF_SIC) register is a write only register and is used to clear interrupts.

Address: 200_4000h base + 10h offset = 200_4010h



SPDIF_SIC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–21 Reserved	This read-only field is reserved and always has the value 0.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun

Table continues on the next page...

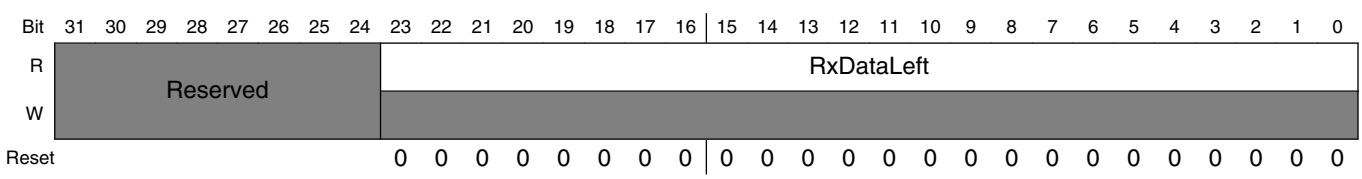
SPDIF_SIC field descriptions (continued)

Field	Description
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–10 -	This field is reserved. Reserved.
9 URxOv	U Channel receive register overrun
8 -	Reserved
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
-	This field is reserved. Reserved.

48.5.7 SPDIFRxLeft Register (SPDIF_SRL)

SPDIFRxLeft register is an audio data reception register.

Address: 200_4000h base + 14h offset = 200_4014h



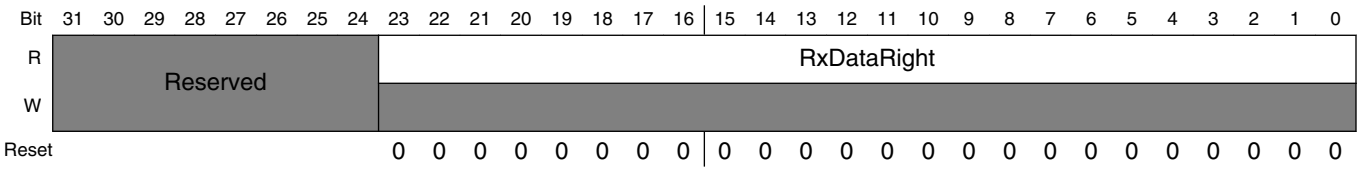
SPDIF_SRL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxDataLeft	Processor receive SPDIF data left

48.5.8 SPDIFRxRight Register (SPDIF_SRR)

SPDIFRxRight register is an audio data reception register.

Address: 200_4000h base + 18h offset = 200_4018h



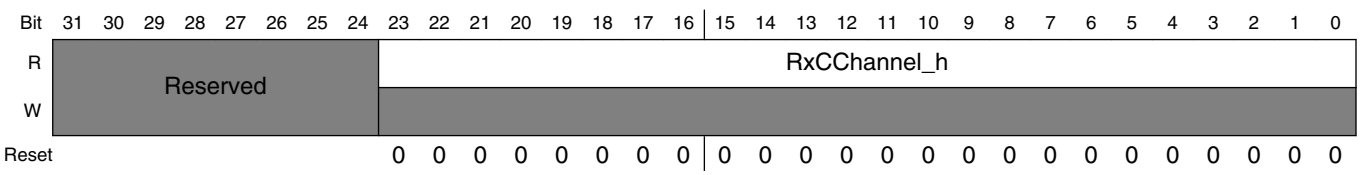
SPDIF_SRR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxDataRight	Processor receive SPDIF data right

48.5.9 SPDIFRxCChannel_h Register (SPDIF_SRCSH)

SPDIFRxCChannel_h register is a channel status reception register.

Address: 200_4000h base + 1Ch offset = 200_401Ch



SPDIF_SRCSH field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxCChannel_h	SPDIF receive C channel register, contains first 24 bits of C channel without interpretation

48.5.10 SPDIFRxCChannel_l Register (SPDIF_SRCSL)

SPDIFRxCChannel_l register is a channel status reception register.

Address: 200_4000h base + 20h offset = 200_4020h

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

SPDIF_SRCSL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxCChannel_l	SPDIF receive C channel register, contains next 24 bits of C channel without interpretation

48.5.11 UchannelRx Register (SPDIF_SRU)

UChannelRx register is a user bits reception register.

Address: 200_4000h base + 24h offset = 200_4024h

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

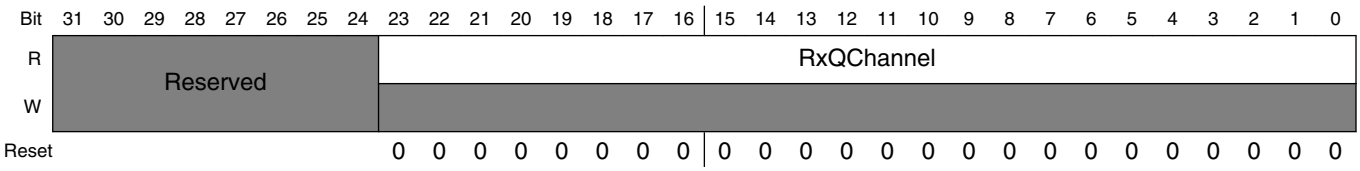
SPDIF_SRU field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
RxUChannel	SPDIF receive U channel register, contains next 3 U channel bytes

48.5.12 QchannelRx Register (SPDIF_SRQ)

QChannelRx register is a user bits reception register.

Address: 200_4000h base + 28h offset = 200_4028h



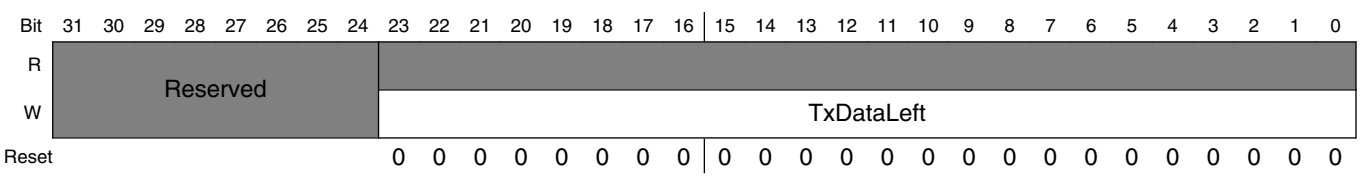
SPDIF_SRQ field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxQChannel	SPDIF receive Q channel register, contains next 3 Q channel bytes

48.5.13 SPDIFTxLeft Register (SPDIF_STL)

SPDIFTxLeft register is an audio data transmission register.

Address: 200_4000h base + 2Ch offset = 200_402Ch



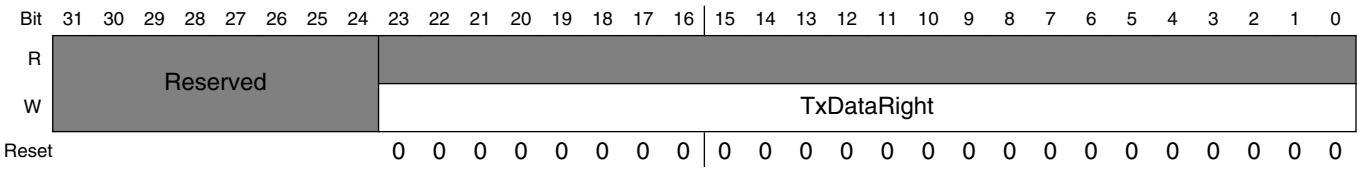
SPDIF_STL field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
TxDataLeft	SPDIF transmit left channel data. It is write-only, and always returns zeros when read

48.5.14 SPDIFTxRight Register (SPDIF_STR)

SPDIFTxRight register is an audio data transmission register.

Address: 200_4000h base + 30h offset = 200_4030h



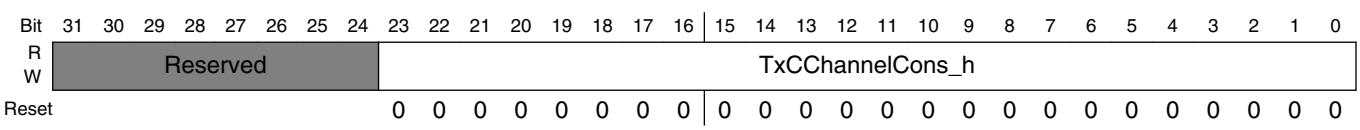
SPDIF_STR field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
TxDataRight	SPDIF transmit right channel data. It is write-only, and always returns zeros when read

48.5.15 SPDIFTxCChannelCons_h Register (SPDIF_STCSCH)

SPDIFTxCChannelCons_h register is a channel status transmission register.

Address: 200_4000h base + 34h offset = 200_4034h



SPDIF_STCSCH field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.

Table continues on the next page...

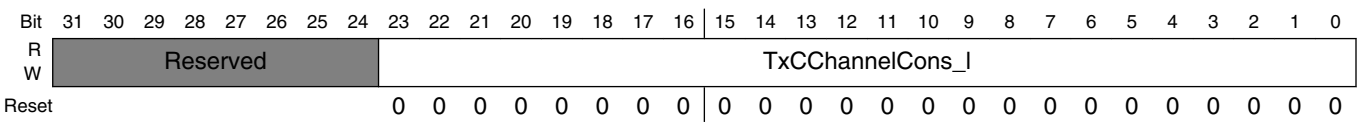
SPDIF_STCSCH field descriptions (continued)

Field	Description
TxCChannelCons_h	SPDIF transmit Cons. C channel data, contains first 24 bits without interpretation. When read, it returns the latest data written by the processor

48.5.16 SPDIFTxCChannelCons_l Register (SPDIF_STCSCL)

SPDIFTxCChannelCons_l register is a channel status transmission register.

Address: 200_4000h base + 38h offset = 200_4038h

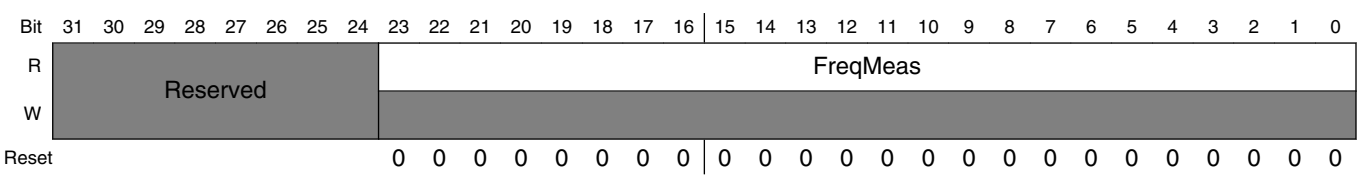


SPDIF_STCSCL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
TxCChannelCons_l	SPDIF transmit Cons. C channel data, contains next 24 bits without interpretation. When read, it returns the latest data written by the processor

48.5.17 FreqMeas Register (SPDIF_SRFM)

Address: 200_4000h base + 44h offset = 200_4044h



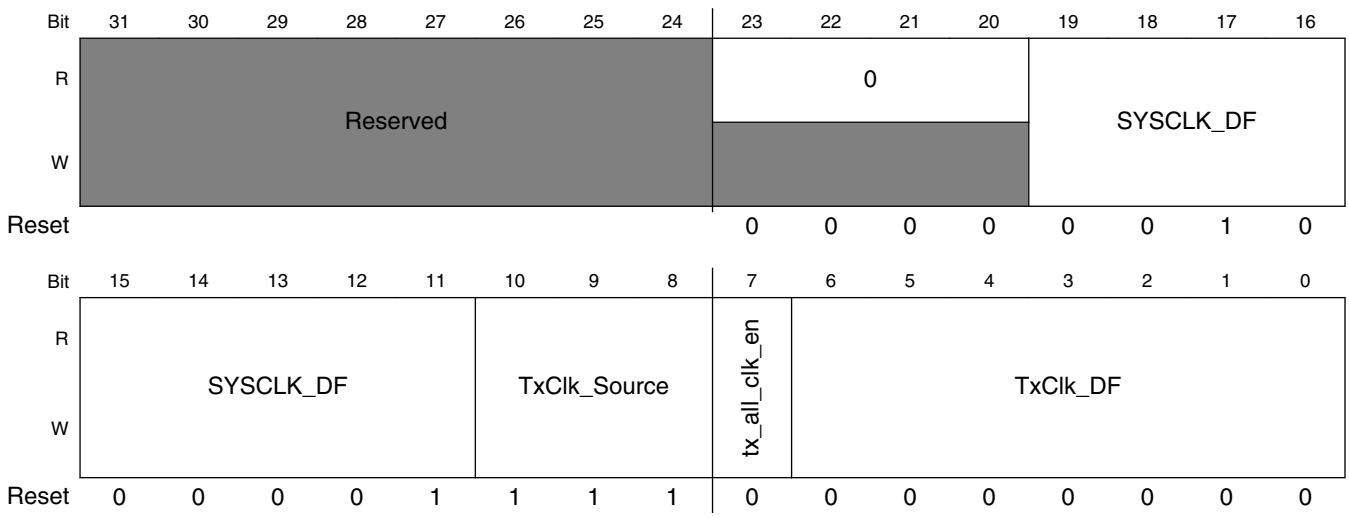
SPDIF_SRFM field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
FreqMeas	Frequency measurement data

48.5.18 SPDIFTxClk Register (SPDIF_STC)

The SPDIFTxClk Control register includes the means to select the transmit clock and frequency division.

Address: 200_4000h base + 50h offset = 200_4050h



SPDIF_STC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–20 Reserved	This read-only field is reserved and always has the value 0.
19–11 SYSCLK_DF	system clock divider factor, 2~512. 0 no clock signal 1 divider factor is 2 ... ... 511 divider factor is 512
10–8 TxClk_Source	000 REF_CLK_32K input (XTALOSC 32 kHz clock) 001 tx_clk input (from SPDIF0_CLK_ROOT. See CCM.) 011 SPDIF_EXT_CLK, from pads 101 ipg_clk input (frequency divided)
7 tx_all_clk_en	Spdif transfer clock enable.When data is going to be transfered, this bit should be set to1. 0 disable transfer clock. 1 enable transfer clock.
TxClk_DF	Divider factor (1-128)

Table continues on the next page...

SPDIF_STC field descriptions (continued)

Field	Description
	0 divider factor is 1 1 divider factor is 2 ... ... 127 divider factor is 128

Chapter 49
System Reset Controller (SRC)

49.1 SRC Overview

The System Reset Controller (SRC) controls the reset and boot operation of the SoC.

It is responsible for the generation of all reset signals and boot decoding.

The reset controller determines the source and the type of reset, such as POR, WARM, COLD, and performs the necessary reset qualification and stretching sequences. Based on the type of reset, the reset logic generates the reset sequence for the entire IC. Whenever the chip is powered on, the reset is issued through SRC_ONOFF signal and the entire chip is reset.

49.1.1 Features

The SRC includes the following features.

- Receives and handles the resets from all the reset sources
- Resets the appropriate domains based upon the resets sources and the nature of the reset
- Latches the SRC_BOOT_MODE pins and common configuration signals from the internal fuse

49.2 External Signals

The following table describes the external signals of SRC.

Table 49-1. SRC External Signals

Signal	Description	Pad	Mode	Direction
SRC_BT_CFG0	Boot configuration signals	LCD_DATA00	ALT6	I
SRC_BT_CFG1		LCD_DATA01	ALT6	I
SRC_BT_CFG2		LCD_DATA02	ALT6	I
SRC_BT_CFG3		LCD_DATA03	ALT6	I
SRC_BT_CFG4		LCD_DATA04	ALT6	I
SRC_BT_CFG5		LCD_DATA05	ALT6	I
SRC_BT_CFG6		LCD_DATA06	ALT6	I
SRC_BT_CFG7		LCD_DATA07	ALT6	I
SRC_BT_CFG8		LCD_DATA08	ALT6	I
SRC_BT_CFG9		LCD_DATA09	ALT6	I
SRC_BT_CFG10		LCD_DATA10	ALT6	I
SRC_BT_CFG11		LCD_DATA11	ALT6	I
SRC_BT_CFG12		LCD_DATA12	ALT6	I
SRC_BT_CFG13		LCD_DATA13	ALT6	I
SRC_BT_CFG14		LCD_DATA14	ALT6	I
SRC_BT_CFG15		LCD_DATA15	ALT6	I
SRC_BT_CFG24		LCD_DATA16	ALT6	I
SRC_BT_CFG25		LCD_DATA17	ALT6	I
SRC_BT_CFG26		LCD_DATA18	ALT6	I
SRC_BT_CFG27		LCD_DATA19	ALT6	I
SRC_BT_CFG28		LCD_DATA20	ALT6	I
SRC_BT_CFG29		LCD_DATA21	ALT6	I
SRC_BT_CFG30		LCD_DATA22	ALT6	I
SRC_BT_CFG31		LCD_DATA23	ALT6	I
SRC_POR_B	Power on reset signal	POR_B	No Muxing (ALT0)	I
SRC_ONOFF	ON/OFF signal	ONOFF	No Muxing (ALT0)	I
SRC_BOOT_MODE0	Boot mode signals	BOOT_MODE0	No Muxing	I
SRC_BOOT_MODE1		BOOT_MODE1	No Muxing	I

49.3 Clocks

The table found here describes the clock sources for SRC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 49-2. SRC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

49.4 Top-level resets, power-up sequence and external supply integration

Information found here defines chip resets, power-up sequence, and external supply integration.

49.4.1 Reset and Power-up Flow

The chip presumes the following reset and power-up flow:

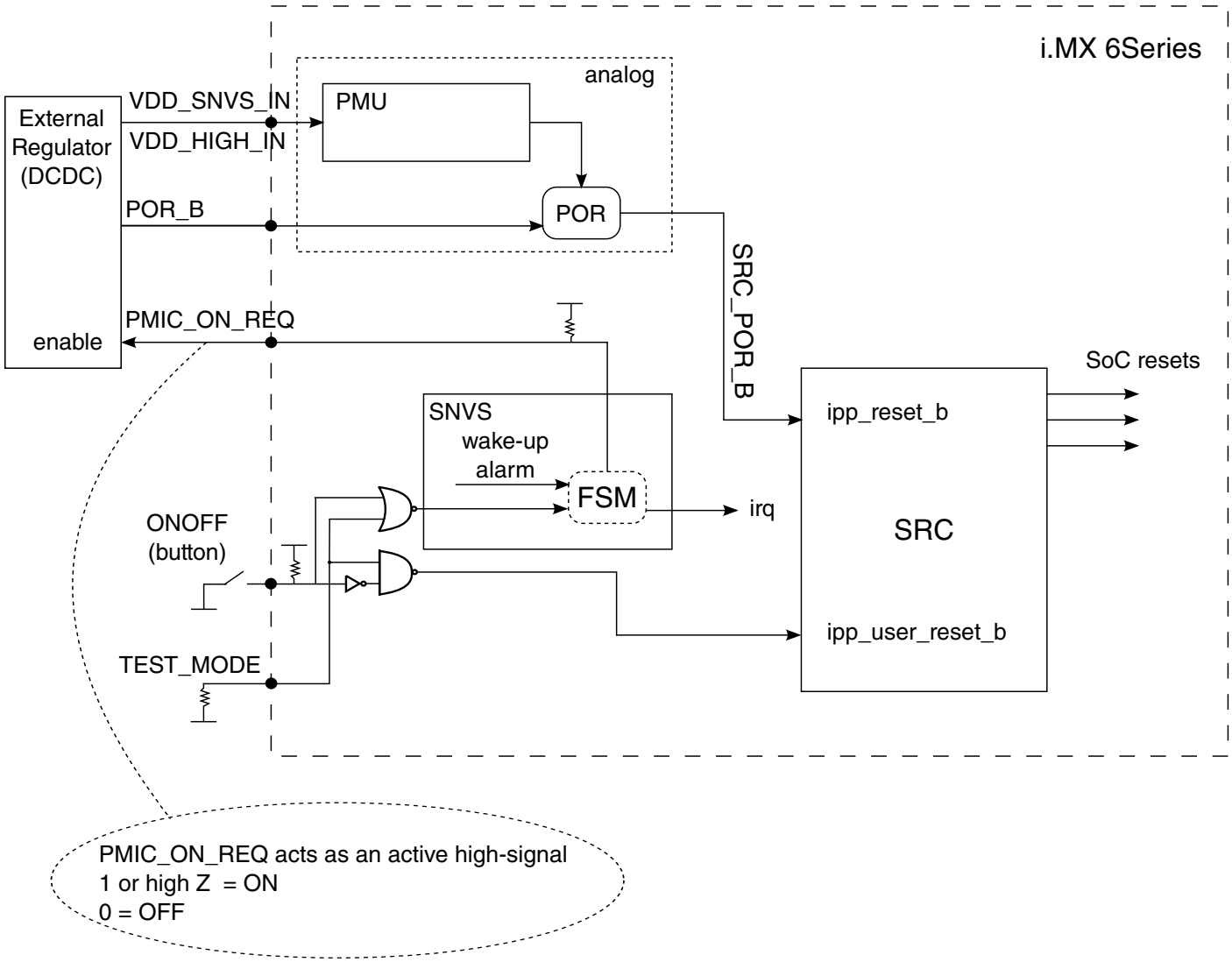


Figure 49-1. Chip reset scheme under PMU control

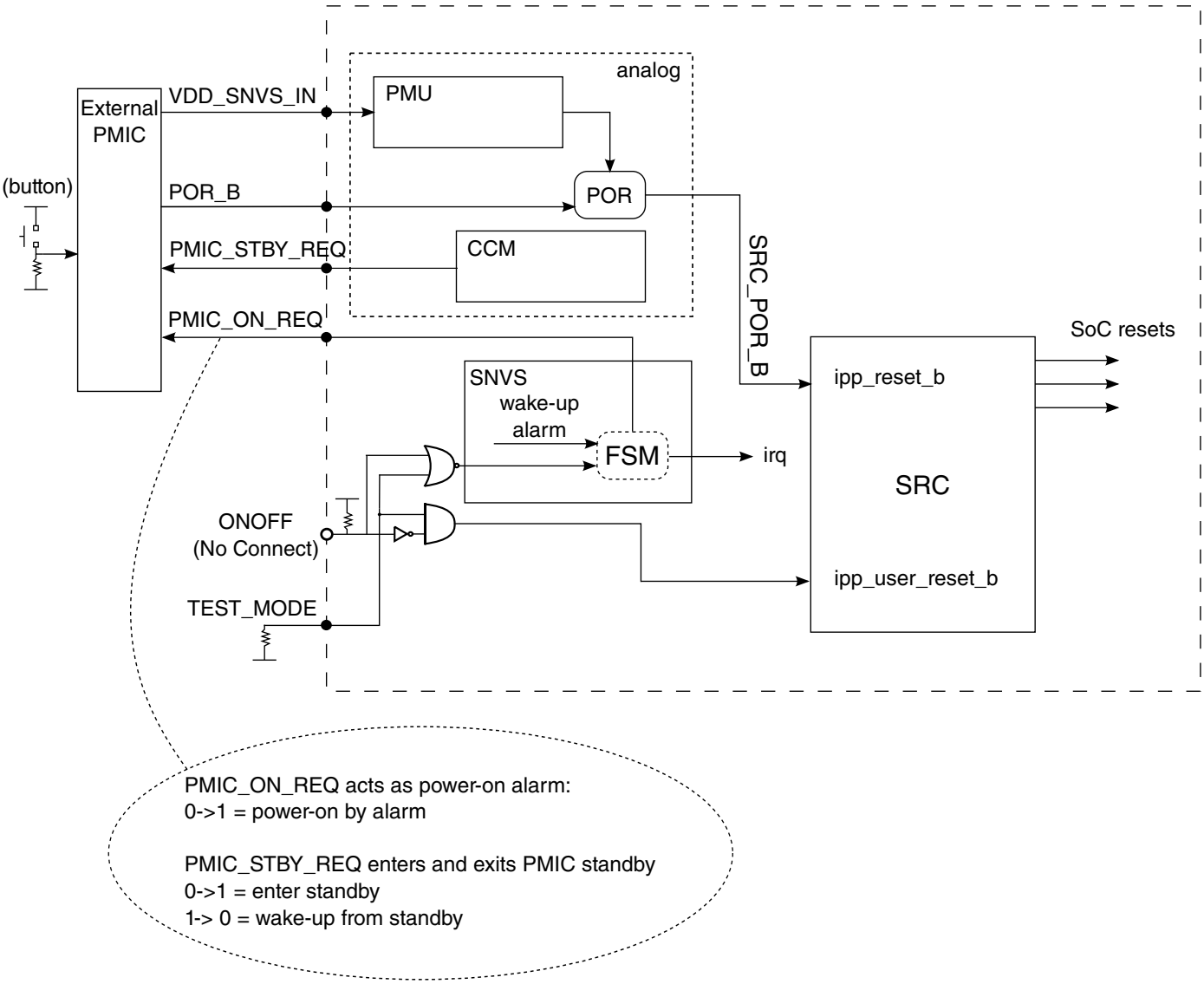


Figure 49-2. Chip reset scheme under external PMIC control

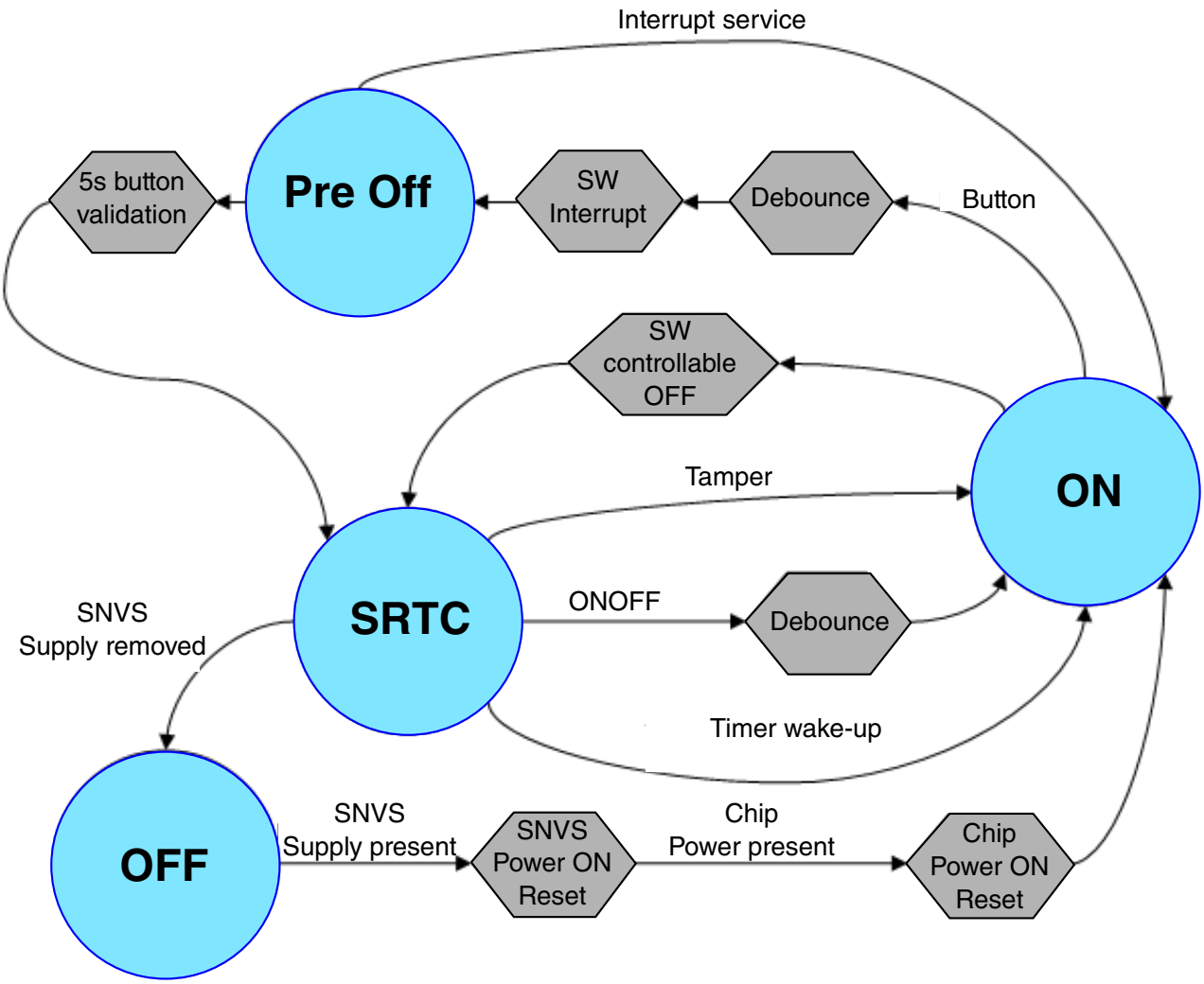


Figure 49-3. Chip on/off state flow diagram

49.4.2 Finite-State Machine (FSM)

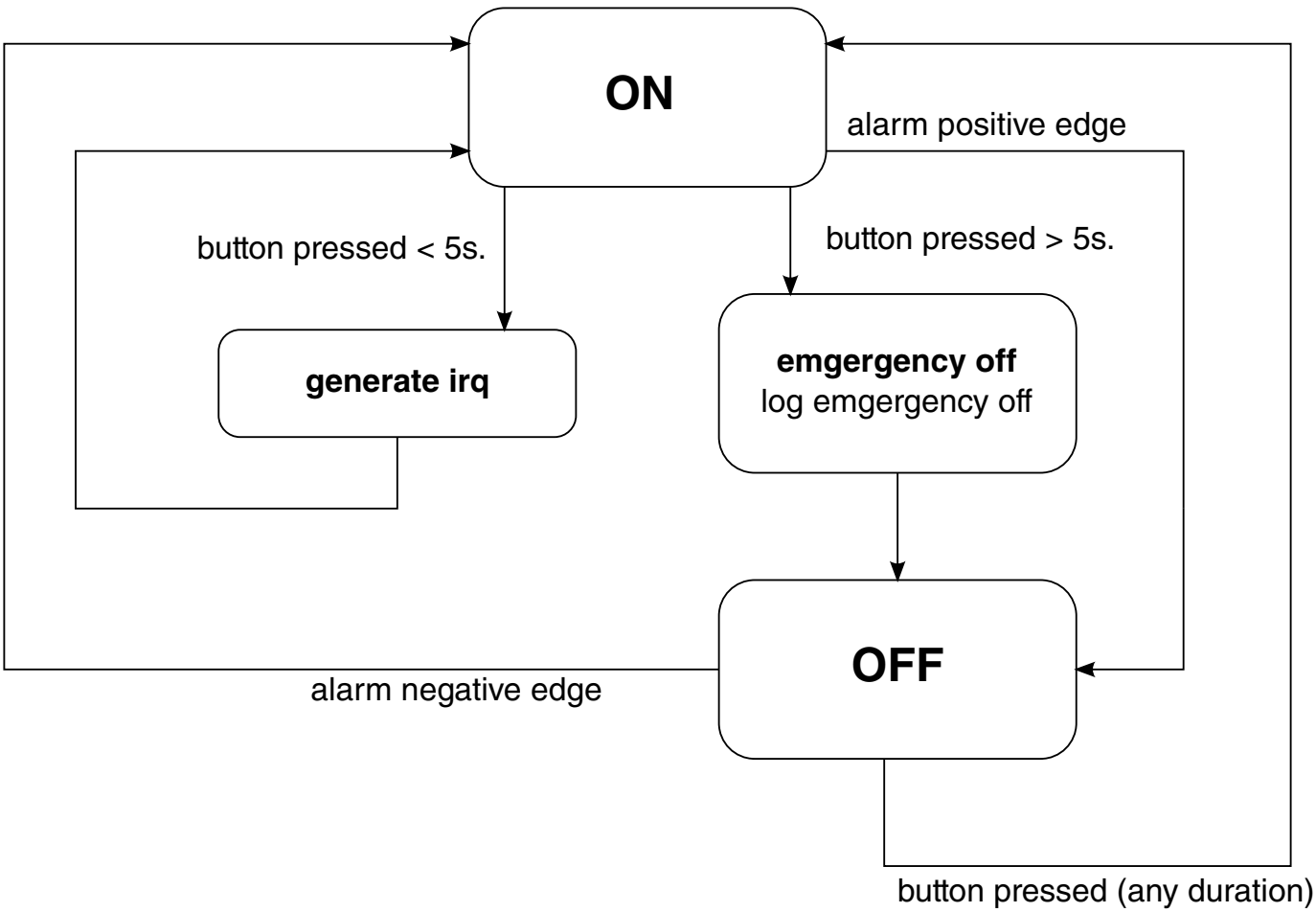


Figure 49-4. FSM

49.4.3 Power mode transitions

Table 49-3. Power mode transitions

Power mode	Configuration with external PMIC	Configuration with internal PMIC
ON, first time	1. Either coin cell or SoC power supply is connected to SNVS. 2. When button is pressed, PMIC powers on.	1. Either coin cell or SoC power supply is connected to SNVS. 2. When button is pressed, 'state' goes ON, PMIC_ON_REQ goes '1'. 3. External regulator is enabled.
Normal ON to OFF, by button	1. Button is pressed for a short duration on the external PMIC. 2. Interrupt request (irq) is sent to SoC from external PMIC.	1. SoC button is pressed for a short duration. 2. Interrupt request (irq) is sent to SoC from FSM. 3. Alarm timer is set up by software routine and started.

Table continues on the next page...

Table 49-3. Power mode transitions (continued)

Power mode	Configuration with external PMIC	Configuration with internal PMIC
	3. SoC is programming PMIC for power off when standby is asserted. 4. In CCM STOP mode, Standby is asserted, PMIC gates SoC supplies.	4. Upon alarm_in assertion to '1', PMIC_ON_REQ goes '0'. 5. External regulator goes OFF.
Emergency ON to OFF, by button	1. Button is pressed for an extended time on the external PMIC. 2. PMIC is powering off.	1. Button is pressed for longer than 5 seconds on the SoC. 2. FSM validates button pressed for 5 seconds. 3. Emergency power off is logged, PMIC_ON_REQ goes '0', alarm_mask goes '1'. 4. External regulator goes OFF.
OFF to ON, by button	1. Button is pressed on the external PMIC. 2. PMIC powers ON.	1. Button is pressed on the SoC. 2. PMIC_ON_REQ goes '1', alarm_mask goes '0'. 3. External regulator powers ON.
OFF to ON, by timer alarm	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. PMIC receives assertion of PMIC_ON_REQ and wakes up.	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. External regulator is enabled by PMIC_ON_REQ = 1.

49.5 Power-On Reset and power sequencing

This module generates an internal POR_B signal that is logically AND'ed with any externally applied SRC_POR_B signal. The internal POR_B signal will be held low until all of the following conditions are met:

- 4ms after the external power supply VDDHIGH_IN is valid
- 1ms after the VDD_SOC_CAP supply is valid

The 4ms and 1ms delays are derived from counting the 32 kHz RTC clock cycles; the accuracy depends on the accuracy of the RTC. When the RTC crystal is either absent or in the process of powering up, an internal ring oscillator will be the source of RTC, which is not as accurate as the crystal.

49.5.1 External POR using SRC_POR_B

If the external SRC_POR_B signal is used to control the processor POR, SRC_POR_B must remain low (asserted) until the VDD_ARM_CAP and VDD_SOC_CAP supplies are stable.

49.5.2 Internal POR

If the external SRC_POR_B signal is not used (always held high or left unconnected), the processor defaults to the internal POR function (PMU controls generation of the POR based on the power supplies).

If the internal POR function is used, the following power supply requirements must be met:

- VDD_ARM_IN and VDD_SOC_IN may be supplied from the same source, or
- VDD_SOC_IN can be supplied before VDD_ARM_IN with a maximum delay of 1 ms.

49.6 Functional Description

49.6.1 Reset Control

This section details the reset control of this device.

49.6.1.1 Reset inputs and outputs

The reset control logic receives reset requests from all potential reset sources. All the immediate sources of reset are directly passed to the reset stretching block, whereas the resets requiring qualification are passed on to the reset qualification logic before they are sent to the reset stretching block.

All reset inputs and outputs are described in the following figure:

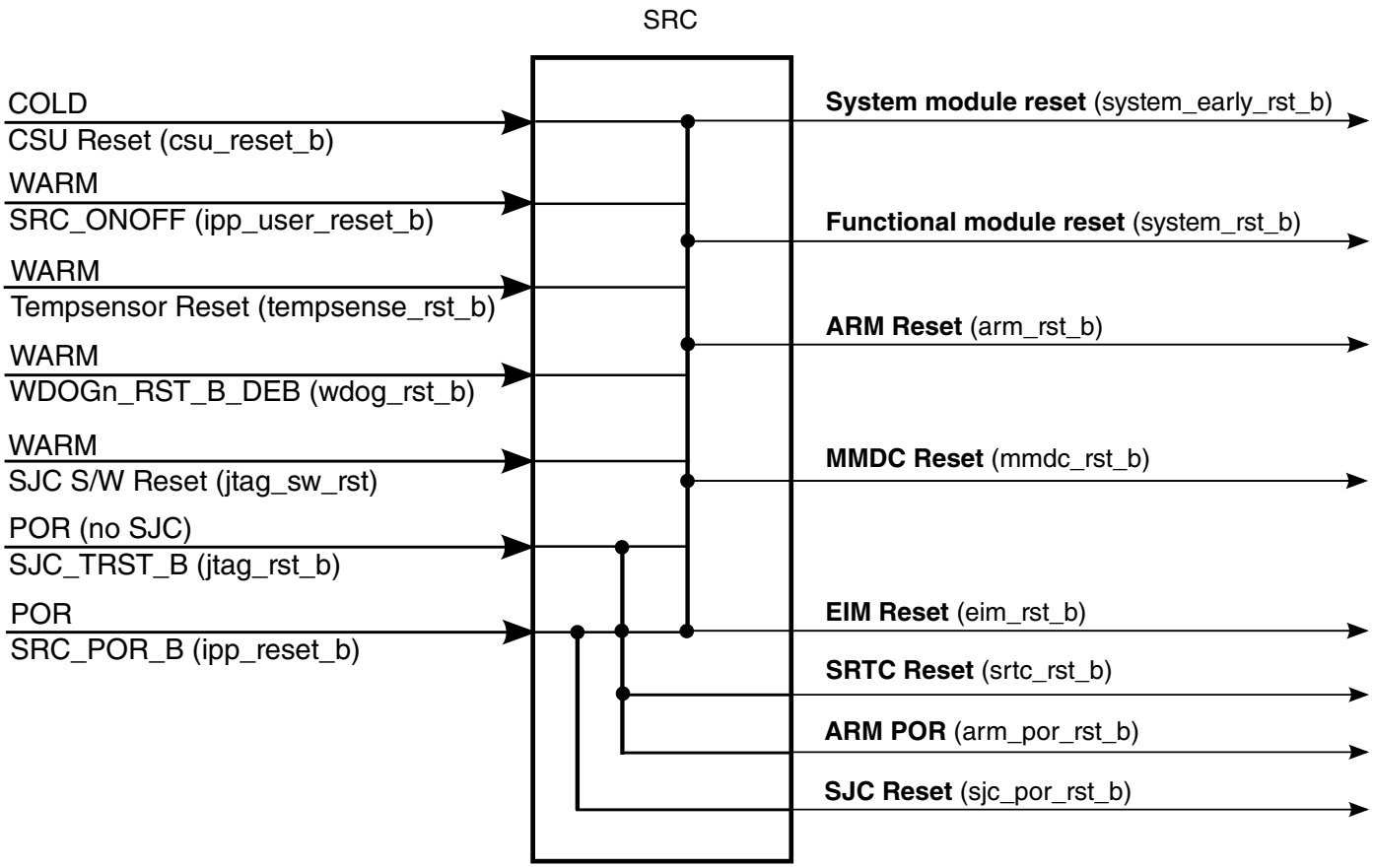


Figure 49-5. SRC inputs and outputs

The reset types and modules they affect are shown in Table 49-4. As there is no chip POR, the POR_B is used to reset the entire chip including test logic and JTAG modules.

NOTE

All resets are expected to be active low except jtag_sw_rst.

Table 49-4. SRC reset functionality

SoC Modules	POR	COLD	WARM
System modules (PLLs, fuses, etc)	yes	yes	yes
Functional modules	yes	yes	yes
ARM	yes	yes	yes
ARM SoC	yes	yes	yes
MMDC	yes	yes	yes
ARM POR	yes	no	no
ARM debug	yes	no	no
SJC	yes	no	no
SRTC	yes	no	no

The reset priorities are POR (strongest), COLD and WARM (weakest). If a stronger reset is asserted during the sequence of a weaker reset, then the weaker sequence will be overridden, and the stronger reset sequence will commence. There is no priority within a reset type (POR, etc). If a reset is asserted during the reset sequence of the same type, the reset sequence will be interrupted and restarted.

The following lists the functionality of each of these reset outputs:

- system_early_rst_b - Resets the system modules that need to start first as CCM, OCOTP_CTRL, FUSEBOX, MMDC, etc.
- system_rst_b - Resets functional modules
- arm_rst_b - Resets ARM module (on regular system reset)
- mmdc_rst_b - Resets MMDC
- arm_por_rst_b - Resets ARM por input
- arm_soc_rst_b - Reset for ARM SOC
- arm_dbg_rst_b - Reset debug logic of ARM
- test_logic_rst_b - Reset test logic (IOMUXC, DAP)
- sjc_por_rst_b - Reset to SJC
- srtc_rst_b - Resets SRTC

NOTE

It is assumed that each reset source will deassert after its assertion, either due to reset generated to the system from SRC, or by negation of the reset source (if it came from an external source to the chip). In the latter case, the reset source is assumed to be held for at least 2 XTALI clocks so it can be sampled by SRC.

49.6.1.2 Reset Handling

49.6.1.2.1 Reset Qualification

The reset qualification logic qualifies the reset source before sending it out to the chip as a valid reset. WARM resets are in this category. All remaining reset sources are immediate resets and are acknowledged by the reset circuitry the moment they are asserted.

WARM resets are not immediate resets. WARM resets do reset the CCM, the source of MMDC clock. So, if a WARM reset were to immediately reset the CCM, then the MMDC clock would be shut off and this may cause the MMDC data to be lost. During normal mode of operation of the chip, the protocol that is followed before shutting off the

MMDC clock is that an mmdc_dvfs_req signal is sent to MMDC and only after the MMDC sends an acknowledge signal, mmdc_dvfs_ack, is the clock to the MMDC gated off.

However, the implication here is that a valid WARM reset source condition will not be able to cause a chip reset until the MMDC sends the acknowledge signal (mmdc_dvfs_ack). For example, a JTAG reset event has occurred but the JTAG reset will not be serviced until the mmdc_dvfs_ack signal is received. So, essentially all WARM reset sources depend on the MMDC providing the mmdc_dvfs_ack acknowledge signal before the reset is performed. When the MMDC is not used, mmdc_dvfs_ack is defaulted high.

The occurrence of WARM reset results in the assertion of warm_reset signal before the system resets are asserted to indicate to the MMDC that the reset occurred is a WARM reset.

A reset source is updated in the Reset Status Register (SRC_SRSR) when it becomes valid, provided it is asserted for the minimum amount of time after asserting. So, all immediate resets are immediately updated in the Status Register (SRC_SRSR). WARM resets would be updated when the mmdc_dvfs_ack signal is received from the MMDC.

Once the reset is qualified, depending on the source of the reset, internal resets are asserted appropriately.

49.6.1.2.2 Reset Sequence and De-Assertion

The SRC_ONOFF will assert immediately after any reset source is recognized (except for the case of WARM reset when MMDC needs to answer mmdc_dvfs_ack first). After all of the reset sources are released, the SRC will start the following set of events depending on the type of reset that occurred.

49.6.1.2.3 POR (SRC_POR_B)

SRC_POR_B is an external reset signal. When the chip is powered up, the reset signal is passed through the POR_B pin indicating power-up sequence. The SRC resets the entire chip including the JTAG (SJC) module. All SRC registers will be reset during the POR sequence.

As soon as SRC_POR_B occurs, all resets are asserted and the entire chip is reset by SRC. The SRC_POR_B is stretched for 2 XTALI cycles and the stretching sequence takes place after 2 XTALI clocks of POR_B pin deassertion.

The sjc_por_rst_b and srtc_rst_b signals are deasserted together with SRC_POR_B signal. Those outputs are also deasserted after the stretching of SRC_POR_B has deasserted.

Once the above resets deassert, system_early_rst_b reset is deasserted after 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock ouputs and the system root clocks.

When the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation.

SRC then enables OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL.

- SRC will prepare the boot information and then de assert mmdc_rst_b.
- SRC will wait 8 ipg clock cycles, and then SRC will enable MMDC clocks.
- SRC will wait 8 XTALI clocks to allow MMDC to generate fixed external clock to external memory SDRAM.
- SRC will enable GPU clocks.
- After 8 ipg cycles, SRC will disable GPU clocks.
- After 8 ipg cycles, resets to all modules will be de-asserted (system_rst_b, gpu_rst_b).
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

49.6.1.2.4 COLD RESET

The sequence is similar to SRC_POR_B except the memory repair operation is not performed.

Once the reset source deasserts, system_early_rst_b reset is deasserted after at least 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock outputs and the system root clocks.

Once the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation. See CCM for more information.

Once system_clk_ready arrives at the SRC, it will enable OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL. OCOTP_CTRL will notify with iim_ready_flag once the fusebox loading finishes.

- SRC will prepare the boot information and then deassert mmdc_rst_b.
- SRC will wait 8 ipg clock cycles, and then SRC will enable MMDC clocks.

- SRC will wait 8 XTALI clocks to allow MMDC to generate fixed external clock to external memory SDRAM.
- SRC will enable GPU clocks so that reset will penetrate this module.
- After 8 ipg cycles, SRC will disable GPU clocks.
- After 8 ipg cycles resets to all modules will be deasserted (system_rst_b, gpu_rst_b).
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

49.6.1.2.5 WARM RESET

WARM reset will be enabled only if SRC_SCR[warm_reset_enable] bit is programmed. Otherwise, all WARM reset sources will generate a COLD reset. This bit will be reset only by a POR.

A WARM reset is similar to a COLD reset except that before the reset is sent, a signal to MMDC is asserted mmdc_dvfs_req (generates DVFS assertion to MMDC) to request to prepare MMDC to a WARM reset, finishing the transactions placing MMDC in self-refresh. Another signal will be asserted to MMDC (warm_reset) that will wrap the WARM reset sequence and will notify MMDC that a WARM reset is in process.

One of the sources of the WARM reset is SRC_ONOFF. If this is the case, it is qualified for 4 XTALI edges. The WARM reset is initiated immediately after 4 XTALI edges. The system does not come out of reset until the the SRC_ONOFF is released.

In case the handshake mechanism with MMDC is stuck, meaning that no mmdc_dvfs_ack is recieved, COLD reset will be generated after a number of XTALI clocks. The number of XTALI clocks is defined in register the SRC_SCR[warm_rst_bypass_count] bitfield (default of this bitfield is 16 XTALI counts.)

The following is a basic description of the WARM reset sequence:

1. ARM sets SRC_SCR[warm_reset_enable] bit to enable the WARM Reset functionality. If this bit is not set, all WARM reset sources will result in COLD reset.
2. Assertion of one of the WARM reset sources.
3. The reset source is qualified in the SRC.
4. If mmdc_dvfs_ack signal is low, then SRC triggers the MMDC to switch to self-refresh mode using mmdc_dvfs_req signal. This is done through the CCM to combine with the DVFS sent from the CCM in case of frequency change of MMDC.
5. Wait for mmdc_dvfs_ack signal from the MMDC. If no ack is recieved during warm_rst_bypass_count number of XTALI clocks, COLD reset will be generated.
6. Assert warm_reset signal to MMDC.
7. SRC asserts system resets

The deassertion sequence is exactly the same as in the Cold Reset except waiting for 8 XTALIs for MMDC to generate fixed external clock to external memory MMDC. This stage is not needed in WARM reset since MMDC is held in self-refresh in WARM reset and there is no need to reconfigure it when exiting WARM reset.

WARM BOOT

Software can save any needed information in the memory before initiating a WARM reset. In this case, software will set SRC_SRSR[warm_boot] bit before initiating WARM reset. After the system returns to run mode, the warm_boot bit will still be set, indicating the software that data was saved in memory and can be reused.

NOTE

mmdc_dvfs_req and acknowledge during WARM reset can be masked in the CCM by configuration of register CCDR[17:16].

49.6.2 Parallel Reset Requests

SRC will follow the following rules in the case of parallel reset requests:

1. The order of strength of resets is POR - strongest, cold - medium, warm - weakest.
2. If a stronger reset is asserted during weaker reset sequence, then the stronger reset will take over and the stronger reset process will commence. The following cases fall into this category:
   - POR reset request in the middle of cold or warm reset process - the cold or warm reset process will be stopped and the POR sequence will start.
   - COLD reset request in the middle of warm reset process - the warm reset process will be stopped and the cold sequence will start.
3. If a weaker reset is asserted during stronger reset sequence, then the stronger reset sequence will continue without interference. If at the end of the stronger reset process the weaker request is still asserted then the weaker sequence will commence. The following cases fall into this category:
   - COLD or WARM reset requests in the middle of POR reset process - the POR process will continue without interference.
   - WARM reset request in the middle of COLD reset process - the COLD process will continue without interference.
4. If a similar reset request is asserted during the process of reset handling, then the process of reset handling will start over (with the same process). The following cases fall into this category:

- POR reset request in the middle of POR reset process - the POR process will start over.
- COLD reset request in the middle of COLD reset process - the COLD process will start over.

There is one exception to this category: WARM reset request in the middle of WARM reset process. In this case, the new WARM reset process cannot restart because MMDC is reset and there can't be a handshake with MMDC if it is reset. In this case the first WARM reset will continue, and only if the second WARM reset is still asserted after the first one has finished, the WARM sequence will start again.

49.6.3 Boot Mode Control

49.6.3.1 BOOT_MODE Pin Latching

The exact boot sequence is controlled by the values of the BOOT_MODE pins on this device.

The value of the BOOT_MODE pins will be latched after the OCOTP_CTRL asserts the fuse read completion flag. After latching, the values of the BOOT_MODE pins are used to determine the booting options of the core as described in the SRC_SBMRx registers.

The boot mode general purpose bits can be provided to the SRC from either e-fuses or GPIO signals. The gpio_bt_sel e-fuse defines the source to be used to derive the boot information. When gpio_bt_sel is set, e-fuses are used. When cleared, GPIO signals are used.

The boot information is provided in SRC_SBMR1 register. The figure below shows the selection of boot mode information.

NOTE

BOOT_MODE[1:0] inputs of SRC are connected to BOOT_MODE[1:0] pins.

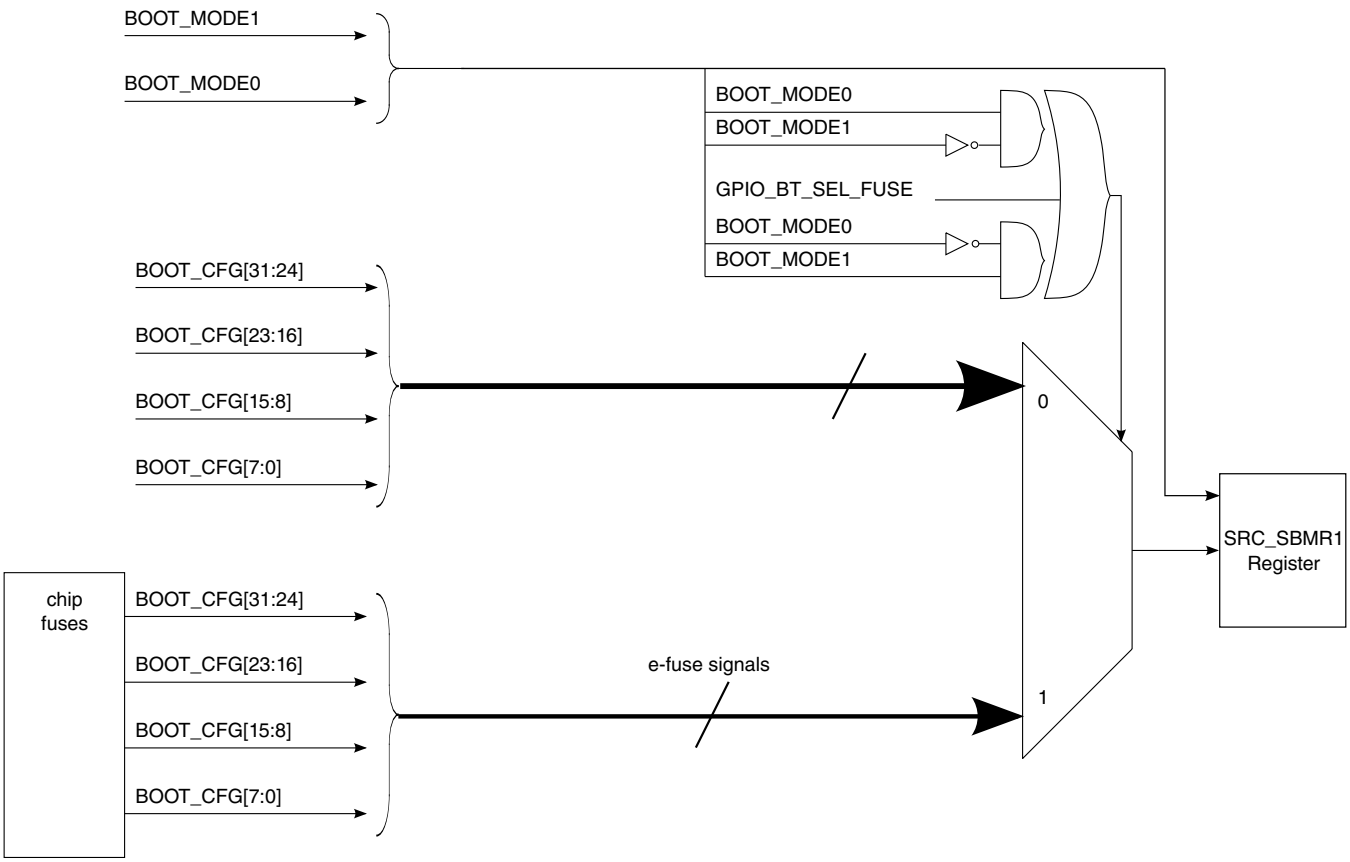


Figure 49-6. Boot mode information

49.7 SRC Memory Map/Register Definition

SRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20D_8000	SRC Control Register (SRC_SCR)	32	R/W	0000_0521h	49.7.1/3072
20D_8004	SRC Boot Mode Register 1 (SRC_SBMR1)	32	R	0000_0000h	49.7.2/3075
20D_8008	SRC Reset Status Register (SRC_SRSR)	32	R/W	0000_0001h	49.7.3/3075
20D_8014	SRC Interrupt Status Register (SRC_SISR)	32	R	0000_0000h	49.7.4/3078
20D_801C	SRC Boot Mode Register 2 (SRC_SBMR2)	32	R	0000_0000h	49.7.5/3080
20D_8020	SRC General Purpose Register 1 (SRC_GPR1)	32	R/W	0000_0000h	49.7.6/3081
20D_8024	SRC General Purpose Register 2 (SRC_GPR2)	32	R/W	0000_0000h	49.7.7/3082
20D_8028	SRC General Purpose Register 3 (SRC_GPR3)	32	R/W	0000_0000h	49.7.8/3082
20D_802C	SRC General Purpose Register 4 (SRC_GPR4)	32	R/W	0000_0000h	49.7.9/3083

Table continues on the next page...

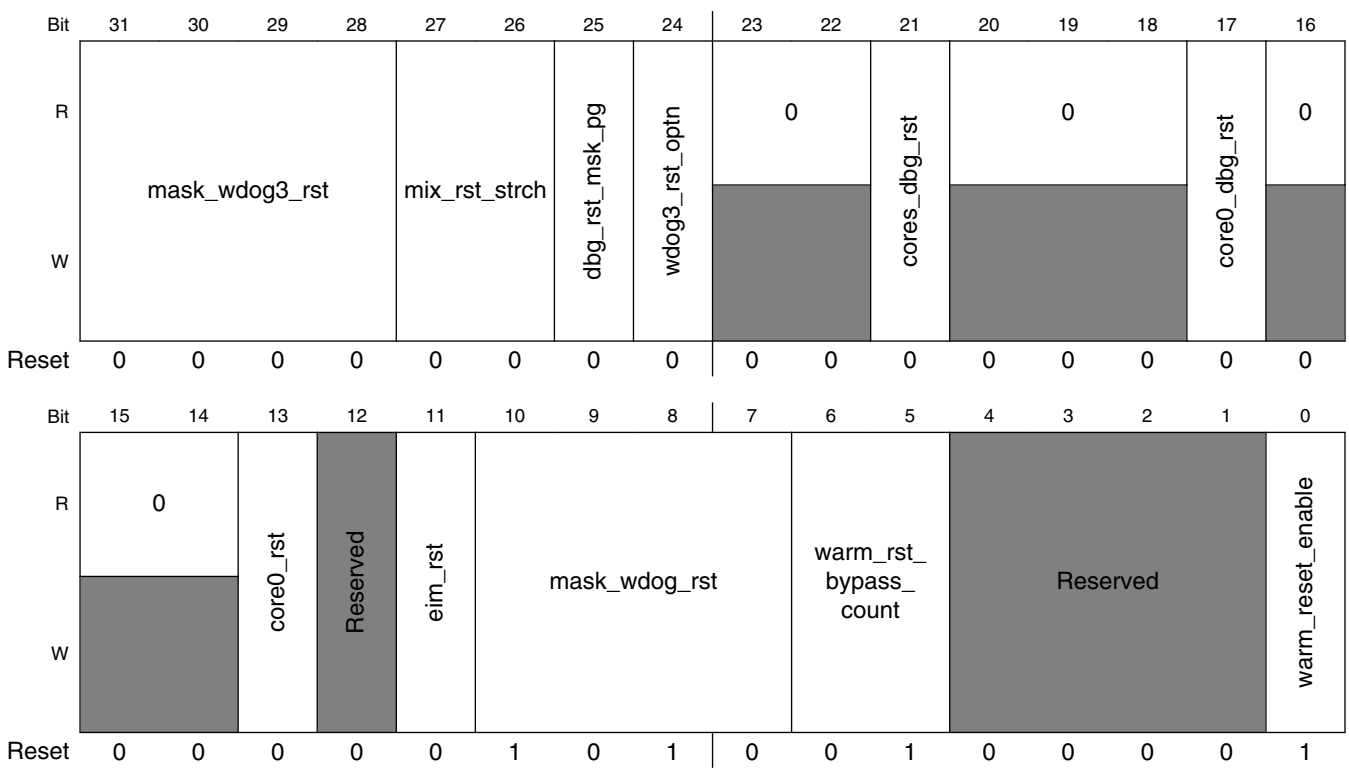
SRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20D_8030	SRC General Purpose Register 5 (SRC_GPR5)	32	R/W	0000_0000h	49.7.10/ 3083
20D_8034	SRC General Purpose Register 6 (SRC_GPR6)	32	R/W	0000_0000h	49.7.11/ 3084
20D_8038	SRC General Purpose Register 7 (SRC_GPR7)	32	R/W	0000_0000h	49.7.12/ 3084
20D_803C	SRC General Purpose Register 8 (SRC_GPR8)	32	R/W	0000_0000h	49.7.13/ 3085
20D_8040	SRC General Purpose Register 9 (SRC_GPR9)	32	R/W	0000_0000h	49.7.14/ 3085
20D_8044	SRC General Purpose Register 10 (SRC_GPR10)	32	R/W	0000_0000h	49.7.15/ 3086

49.7.1 SRC Control Register (SRC_SCR)

The Reset control register (SCR), contains bits that control operation of the reset controller.

Address: 20D_8000h base + 0h offset = 20D_8000h



SRC_SCR field descriptions

Field	Description
31–28 mask_wdog3_rst	Mask wdog3_rst_b source. If these 4 bits are coded from A to 5 then, the wdog3_rst_b input to SRC will be masked and the wdog3_rst_b will not create a reset to the chip. **NOTE:** During the time the WDOG3 event is masked using SRC logic, it is likely that the WDOG3 Reset Status Register (WRSR) bit 1 (which indicates a WDOG3 timeout event) will get asserted. Software / OS developer must prepare for this case. Re-enabling the WDOG3 is possible, by un-masking it in SRC, though it must be preceded by servicing the WDOG3. However, for the case that the event has been asserted, the status bit (WRSR bit-1) will remain asserted, regardless of servicing the WDOG3 module. **NOTE:** Hardware reset is the only way to cause the de-assertion of that bit. Any other code than 0101 will be coded to 1010 i.e. wdog3_rst_b is not masked 0101 wdog3_rst_b is masked 1010 wdog3_rst_b is not masked
27–26 mix_rst_strch	SoC mix (Audio, ENET, uSDHC, EIM, QSPI, OCRAM, MMDC, etc) power up reset stretch mix reset width = (mix_rst_strtch +1)* 88 ipg_clk cycles 00 mix reset width is 88 ipg_cycle cycles 01 mix reset width is 2 * 88 ipg_cycle cycles 10 mix reset width is 3 * 88 ipg_cycle cycles 11 mix reset width is 4 * 88 ipg_cycle cycles
25 dbg_rst_msk_pg	Do not assert debug resets after power gating event of core 0 do not mask core debug resets (debug resets will be asserted after power gating event) 1 mask core debug resets (debug resets won't be asserted after power gating event)
24 wdog3_rst_optn	Wdog3_rst_b option 0 Wdog3_rst_b asserts M4 reset (default) 1 Wdog3_rst_b asserts global reset
23–22 Reserved	This read-only field is reserved and always has the value 0.
21 cores_dbg_rst	Software reset for debug of arm platform only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert arm platform debug reset 1 assert arm platform debug reset
20–18 Reserved	This read-only field is reserved and always has the value 0.
17 core0_dbg_rst	Software reset for core0 debug only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 debug reset 1 assert core0 debug reset
16–14 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRC_SCR field descriptions (continued)

Field	Description
13 core0_rst	Software reset for core0 only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 reset 1 assert core0 reset
12 -	This field is reserved. Reserved
11 eim_rst	EIM reset is needed in order to reconfigure the eim chip select. The software reset bit must de-asserted. The eim chip select configuration should be updated. The software bit must be re-asserted since this is not self-refresh.
10–7 mask_wdog_rst	Mask wdog_rst_b source. If these 4 bits are coded from A to 5 then, the wdog_rst_b input to SRC will be masked and the wdog_rst_b will not create a reset to the chip. **NOTE:** During the time the WDOG event is masked using SRC logic, it is likely that the WDOG Reset Status Register (WRSR) bit 1 (which indicates a WDOG timeout event) will get asserted. software / OS developer must prepare for this case. Re-enabling the WDOG is possible, by un-masking it in SRC, though it must be preceded by servicing the WDOG. However, for the case that the event has been asserted, the status bit (WRSR bit-1) will remain asserted, regardless of servicing the WDOG module. (Hardware reset is the only way to cause the de-assertion of that bit). any other code will be coded to 1010 i.e. wdog_rst_b is not masked 0101 wdog_rst_b is masked 1010 wdog_rst_b is not masked (default)
6–5 warm_rst_ bypass_count	Defines the XTALI cycles to count before bypassing the MMDC acknowledge for WARM reset. If the MMDC acknowledge will not be asserted before this counter has elapsed, then a COLD reset will be initiated. 00 Counter not to be used - system will wait until MMDC acknowledge until it is asserted. 01 Wait 16 XTALI cycles before changing WARM reset to a COLD reset. 10 Wait 32 XTALI cycles before changing WARM reset to a COLD reset. 11 Wait 64 XTALI cycles before changing WARM reset to a COLD reset
4–1 -	This field is reserved. Reserved
0 warm_reset_ enable	WARM reset enable bit. WARM reset will be enabled only if warm_reset_enable bit is set. Otherwise all WARM reset sources will generate COLD reset. 0 WARM reset disabled 1 WARM reset enabled

49.7.2 SRC Boot Mode Register 1 (SRC_SBMR1)

The Boot Mode register (SBMR) contains bits that reflect the status of Boot Mode Pins of the chip. The reset value is configuration dependent (depending on boot/fuses/IO pads).

If SRC_GPR10[28] bit is set, this bit instructs the ROM code to use the SRC_GPR9 registe as if it is SBMR1. This allows software to override the fuse bits and boot from an alternate boot source.

Address: 20D_8000h base + 4h offset = 20D_8004h

Bit	31–24	23–16	15–8	7–0
R	BOOT_CFG4[7:0]	BOOT_CFG3[7:0]	BOOT_CFG2[7:0]	BOOT_CFG1[7:0]
W				
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

SRC_SBMR1 field descriptions

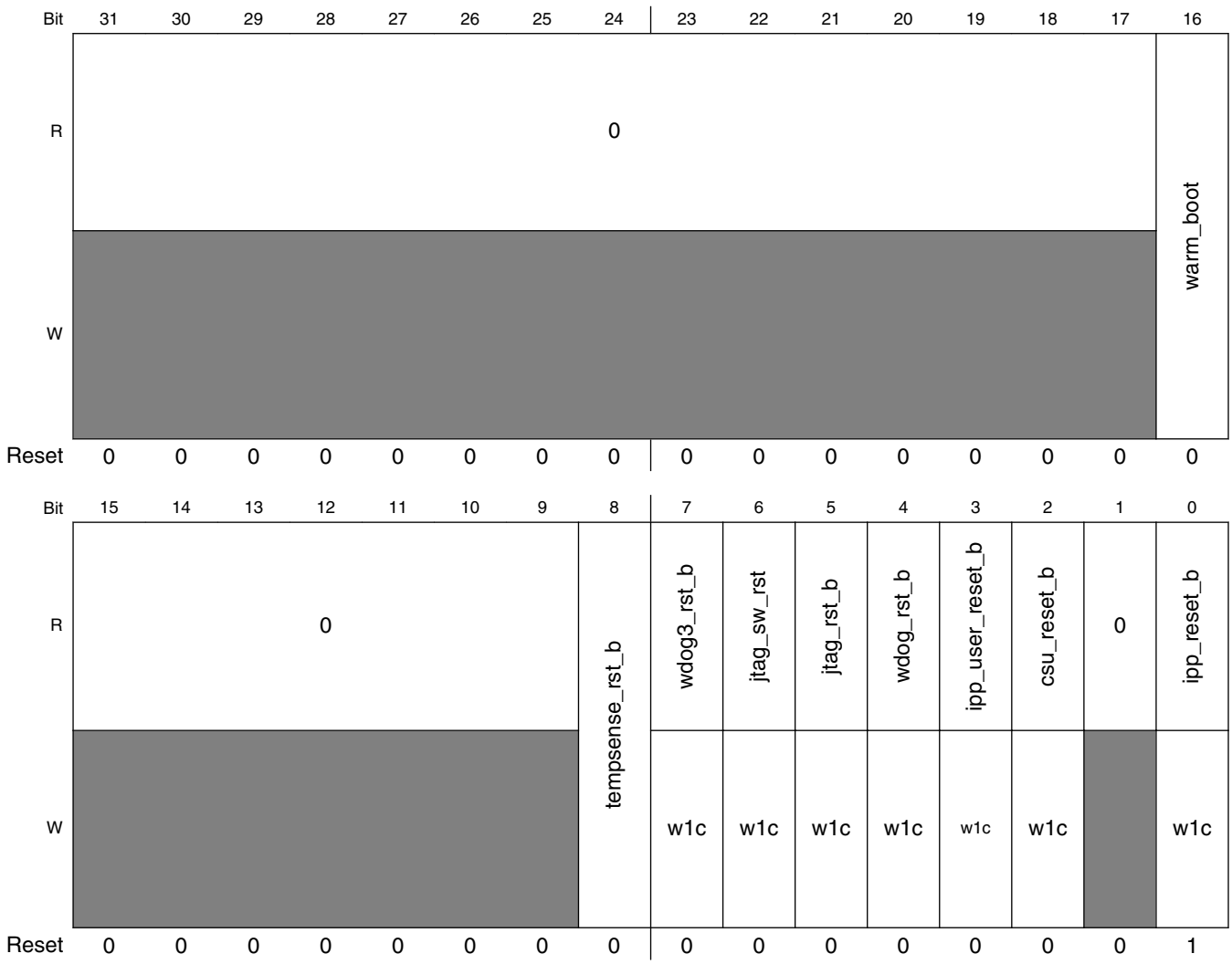
Field	Description
31–24 BOOT_ CFG4[7:0]	Refer to fusemap.
23–16 BOOT_ CFG3[7:0]	Refer to fusemap.
15–8 BOOT_ CFG2[7:0]	Refer to fusemap.
BOOT_ CFG1[7:0]	Refer to fusemap.

49.7.3 SRC Reset Status Register (SRC_SRSR)

The SRSR is a write to one clear register which records the source of the reset events for the chip. The SRC reset status register will capture all the reset sources that have occurred. This register is reset on ipp_reset_b. This is a read-write register.

For bit[6-0] - writing zero does not have any effect. Writing one will clear the corresponding bit. The individual bits can be cleared by writing one to that bit. When the system comes out of reset, this register will have bits set corresponding to all the reset sources that occurred during system reset. Software has to take care to clear this register by writing one after every reset that occurs so that the register will contain the information of recently occurred reset.

Address: 20D_8000h base + 8h offset = 20D_8008h



SRC_SRSR field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 warm_boot	WARM boot indication shows that WARM boot was initiated by software. This indicates to the software that it saved the needed information in the memory before initiating the WARM reset. In this case, software will set this bit to '1', before initiating the WARM reset. The warm_boot bit should be used as indication only after a warm_reset sequence. Software should clear this bit after warm_reset to indicate that the next warm_reset is not performed with warm_boot. Please refer to Reset Sequence and De-Assertion for details on warm_reset. 0 WARM boot process not initiated by software. 1 WARM boot initiated by software.
15–9 Reserved	This read-only field is reserved and always has the value 0.
8 tempsense_rst_b	Temper Sensor software reset. Indicates whether the reset was the result of software reset from on-chip Temperature Sensor.

Table continues on the next page...

SRC_SRSR field descriptions (continued)

Field	Description
	0 Reset is not a result of software reset from Temperature Sensor. 1 Reset is a result of software reset from Temperature Sensor.
7 wdog3_rst_b	IC Watchdog3 Time-out reset. Indicates whether the reset was the result of the watchdog3 time-out event. 0 Reset is not a result of the watchdog3 time-out event. 1 Reset is a result of the watchdog3 time-out event.
6 jtag_sw_rst	JTAG software reset. Indicates whether the reset was the result of software reset from JTAG. 0 Reset is not a result of software reset from JTAG. 1 Reset is a result of software reset from JTAG.
5 jtag_rst_b	HIGH - Z JTAG reset. Indicates whether the reset was the result of HIGH-Z reset from JTAG. 0 Reset is not a result of HIGH-Z reset from JTAG. 1 Reset is a result of HIGH-Z reset from JTAG.
4 wdog_rst_b	IC Watchdog Time-out reset. Indicates whether the reset was the result of the watchdog time-out event. 0 Reset is not a result of the watchdog time-out event. 1 Reset is a result of the watchdog time-out event.
3 ipp_user_reset_b	Indicates whether the reset was the result of the ipp_user_reset_b qualified reset. 0 Reset is not a result of the ipp_user_reset_b qualified as COLD reset event. 1 Reset is a result of the ipp_user_reset_b qualified as COLD reset event.
2 csu_reset_b	Indicates whether the reset was the result of the csu_reset_b input. **NOTE:** If case the csu_reset_b occurred during a WARM reset process, during the phase that ipg_clk is not available yet, then the occurrence of CSU reset will not be reflected in this bit. 0 Reset is not a result of the csu_reset_b event. 1 Reset is a result of the csu_reset_b event.
1 Reserved	This read-only field is reserved and always has the value 0.
0 ipp_reset_b	Indicates whether reset was the result of ipp_reset_b pin (Power-up sequence) 0 Reset is not a result of ipp_reset_b pin. 1 Reset is a result of ipp_reset_b pin.

49.7.4 SRC Interrupt Status Register (SRC_SISR)

Address: 20D_8000h base + 14h offset = 20D_8014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0		core0_wdog_rst_req	Reserved				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_SISR field descriptions

Field	Description
31–9 Reserved	This read-only field is reserved and always has the value 0.
8–6 Reserved	This read-only field is reserved and always has the value 0.
5 core0_wdog_rst_req	WDOG reset request from core0. Read-only status bit.
-	This field is reserved. Reserved

49.7.5 SRC Boot Mode Register 2 (SRC_SBMR2)

The Boot Mode register (SBMR), contains bits that reflect the status of Boot Mode Pins of the chip. The default values for those bits depends on the values of pins/fuses during reset sequence, hence the question mark on their default value.

Address: 20D_8000h base + 1Ch offset = 20D_801Ch

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

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

SRC_SBMR2 field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 BMOD[1:0]	BMOD[1:0] shows the latched state of the BOOT_MODE1 and BOOT_MODE0 signals on the rising edge of POR_B. See the Boot mode pin settings section of System Boot.
23–5 Reserved	This read-only field is reserved and always has the value 0.
4 BT_FUSE_SEL	BT_FUSE_SEL (connected to gpio bt_fuse_sel) shows the state of the BT_FUSE_SEL fuse. See Fusemap for additional information on this fuse.
3 DIR_BT_DIS	DIR_BT_DIS shows the state of the DIR_BT_DIS fuse. See Chapter 5, Fusemap for additional information on this fuse.
2 Reserved	This read-only field is reserved and always has the value 0.
SEC_ CONFIG[1:0]	SECONFIG[1] shows the state of the SECONFIG[1] fuse. See Fusemap for additional information on this fuse. SECONFIG[0] shows the state of the SECONFIG[0] fuse. This fuse is shown as reserved in Fusemap (address 0x440[1]) because it does not have a user-relevant function.

49.7.6 SRC General Purpose Register 1 (SRC_GPR1)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 20h offset = 20D_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR1 field descriptions

Field	Description
PERSISTENT_ ENTRY0	Holds entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

49.7.7 SRC General Purpose Register 2 (SRC_GPR2)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 24h offset = 20D_8024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR2 field descriptions

Field	Description
PERSISTENT_ARG0	Holds argument of entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

49.7.8 SRC General Purpose Register 3 (SRC_GPR3)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 28h offset = 20D_8028h

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

SRC_GPR3 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

49.7.9 SRC General Purpose Register 4 (SRC_GPR4)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 2Ch offset = 20D_802Ch

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

SRC_GPR4 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

49.7.10 SRC General Purpose Register 5 (SRC_GPR5)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 30h offset = 20D_8030h

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

SRC_GPR5 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

49.7.11 SRC General Purpose Register 6 (SRC_GPR6)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 34h offset = 20D_8034h

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

SRC_GPR6 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

49.7.12 SRC General Purpose Register 7 (SRC_GPR7)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 38h offset = 20D_8038h

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

SRC_GPR7 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

49.7.13 SRC General Purpose Register 8 (SRC_GPR8)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 3Ch offset = 20D_803Ch

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

SRC_GPR8 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

49.7.14 SRC General Purpose Register 9 (SRC_GPR9)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 40h offset = 20D_8040h

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

SRC_GPR9 field descriptions

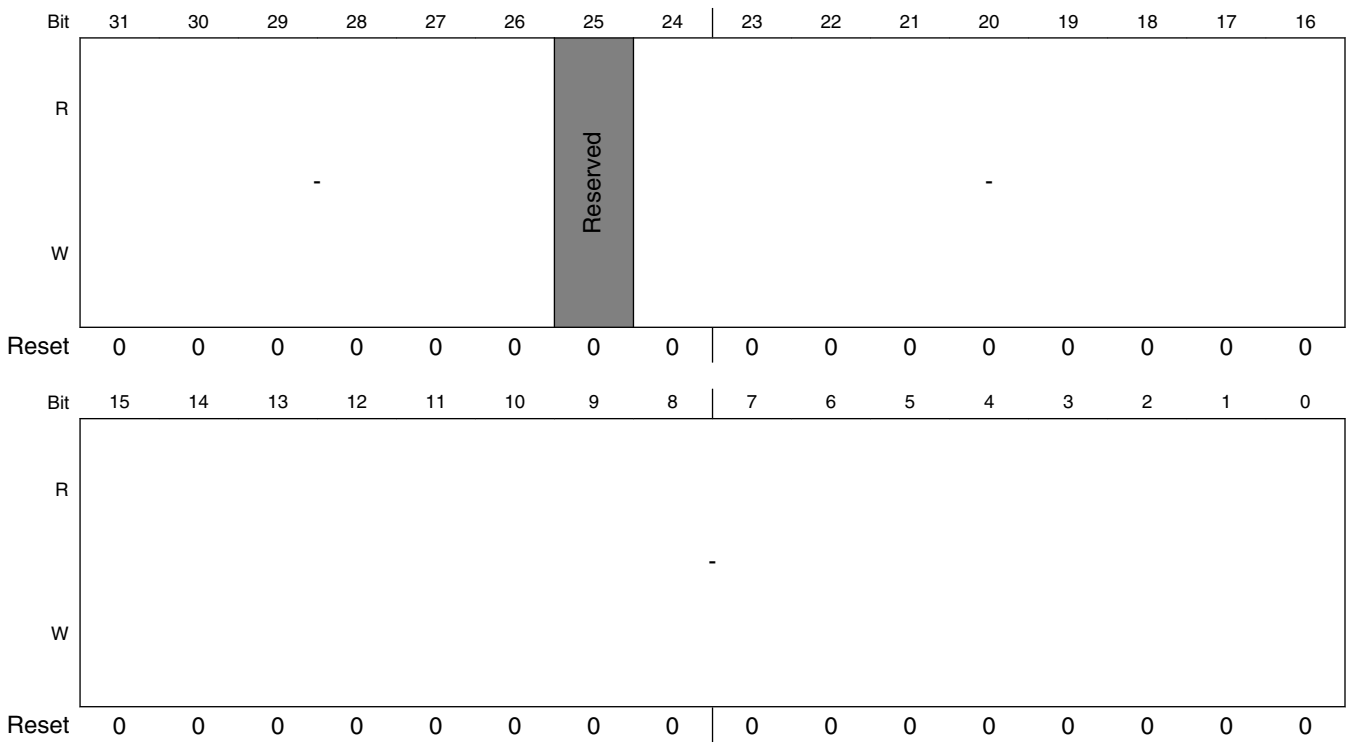
Field	Description
-	This field is reserved. Reserved.

49.7.15 SRC General Purpose Register 10 (SRC_GPR10)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 20D_8000h base + 44h offset = 20D_8044h



SRC_GPR10 field descriptions

Field	Description
31–26 -	Read/write bits, for general purpose **NOTE:** Reset only by POR
25 -	This field is reserved. Reserved.
-	Read/write bits, for general purpose **NOTE:** Reset only by POR

Chapter 50
Temperature Monitor (TEMPMON)

50.1 Overview

The temperature sensor module implements a temperature sensor/conversion function based on a temperature-dependent voltage to time conversion.

The module features alarm functions that can raise independent interrupt signals if the temperature is above two high-temperature thresholds and below a low temperature threshold. These temperature thresholds are programmable and designated as low, high and panic temperature. The panic threshold is a special programmable threshold in that if the temperature increases above this value and the temperature-panic-reset interrupt is enabled in the System Reset Controller, the hardware will assume that software no longer has control over the thermal situation and will initiate a reset of the chip.

In order to avoid false panic temperature initiated resets, the panic alarm will not fire until the temperature panic condition has been met for four consecutive conversion cycles. A self-repeating mode can also be programmed which executes a temperature sensing operation based on a programmed delay.

Since the high and low temperature thresholds are programmable, they form a sliding temperature bracket that can be tailored to the application's needs. For example, at start-up software can set the low temperature threshold to the minimum temperature code and the high temperature threshold to a maximum operating temperature for the system.

The system can then use this module to monitor the on-die temperature and take appropriate actions such as throttling back the core frequency when a the high temperature interrupt is set. Once the high temperature interrupt is set then the system can program the low temperature threshold to a desired cool down temperature. The system would then switch to monitoring the low temperature alarm and wait for its interrupt to be set. With this scheme, once the low temperature interrupt is set then software could be assured that the temperature has cooled down to a safe level and the process could be repeated.

The high-level implementation of the temperature sensor is shown in the figure below.

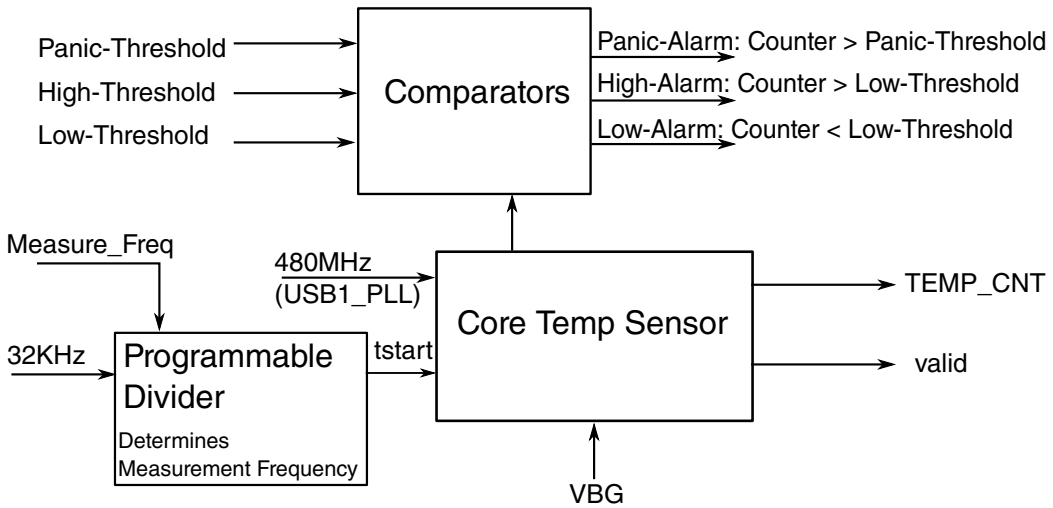


Figure 50-1. High Level Temp Sensor System Diagram

As shown in the figure above, the temperature sensor uses and assumes that the bandgap reference, 480MHz PLL and 32KHz RTC modules are properly programmed and fully settled for correct operation.

50.2 Software Usage Guidelines

During normal system operation software can use the temperature sensor counter output (TEMP_CNT) in conjunction with the fused temperature calibration data to determine the on-die operational temperature or to set an over-temperature interrupt alarm to within a couple of °C.

Based on calibration, two sets of temperature and counter values will be available via fuses on the device. These data points will correspond to the points (N_1, T_1) and (N_2, T_2) in the curve below.

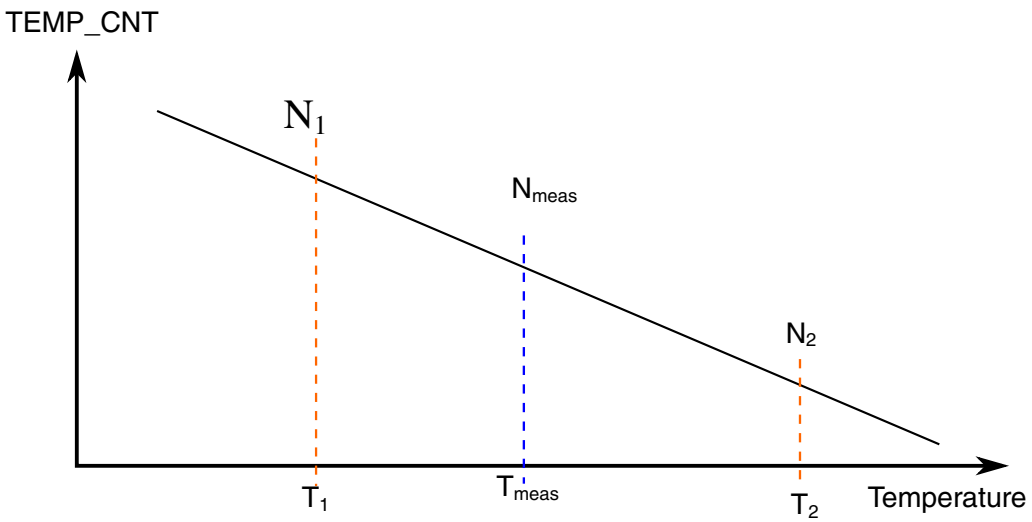


Figure 50-2. Temperature Measurement Cycle

After a temperature measurement cycle, software should use the calibration points in conjunction with the temperature code value in the TEMPMON_TEMPSENSE0[TEMP_CNT] bitfield to calculate the temperature for the device using the following equation:

$T_{meas} = T_2 - (N_{meas} - N_2) * ((T_2 – T_1) / (N_1 – N_2))$

Likewise, to determine the alarm counter value to be written in the TEMPMON_TEMPSENSE0 register for a temperature based interrupt, the above equation can be solved for the N_{meas} value that should be used based on the desired temperature trigger.

The temperature calibration point fuse values are available in the OCOTP_ANA1 register. The temperature calibration values are fused individually for each part in the product testing process. The fields of this register are described in the following table.

Table 50-1. OCOTP_ANA1 Temperature Sensor Calibration Data

Bit Range	Bit Mask	Name	Description
[31:20]	FFF0_0000h	ROOM_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at room temperature (25.0 °C).
[19:8]	000F_FF00h	HOT_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at the hot temperature, i.e. HOT_TEMP.
[7:0]	0000_00FFh	HOT_TEMP	The hot temperature test point. Each LSB equals 1 °C.

The points on the calibration curve are as follows.

- (N_1, T_1) = (ROOM_COUNT, 25.0)

- $(N_2, T_2) = (HOT_COUNT, HOT_TEMP)$
- $(N_{meas}, T_{meas}) = (TEMP_CNT, T_{meas})$

Substituting the fields from OCOTP_ANA1 into the earlier equation results in the following:

$T_{meas} = HOT_TEMP - (N_{meas} - HOT_COUNT) * ((HOT_TEMP - 25.0) / (ROOM_COUNT – HOT_COUNT))$

50.3 TEMPMON Memory Map/Register Definition

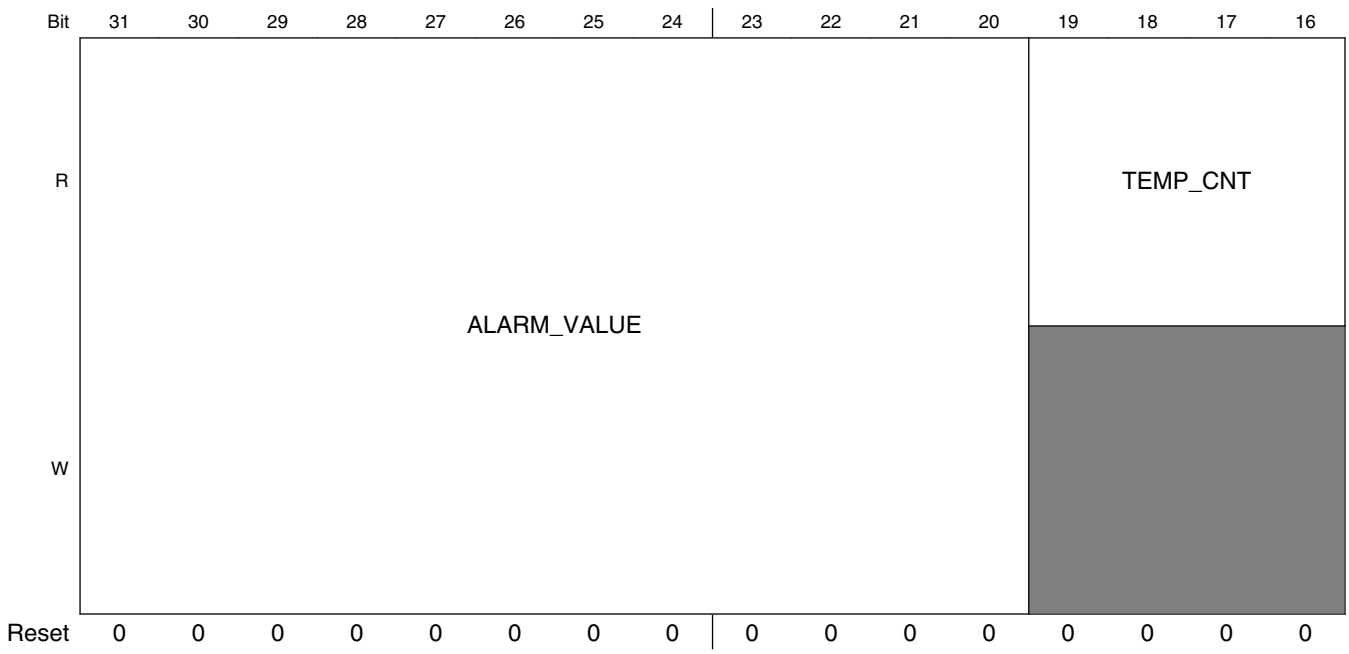
TEMPMON memory map

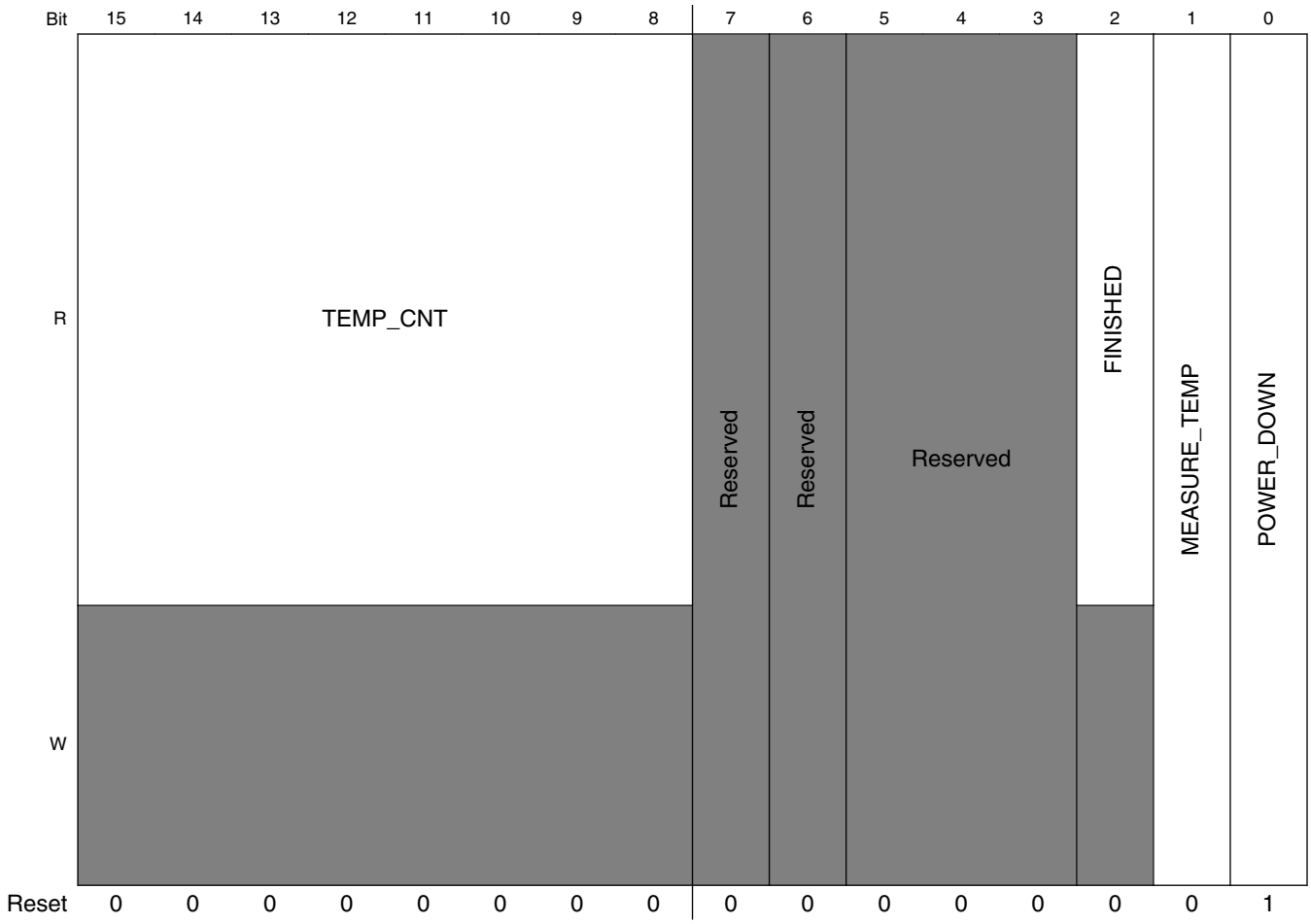
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8180	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0)	32	R/W	0000_0001h	50.3.1/3091
20C_8184	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_SET)	32	R/W	0000_0001h	50.3.1/3091
20C_8188	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_CLR)	32	R/W	0000_0001h	50.3.1/3091
20C_818C	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_TOG)	32	R/W	0000_0001h	50.3.1/3091
20C_8190	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1)	32	R/W	0000_0001h	50.3.2/3093
20C_8194	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_SET)	32	R/W	0000_0001h	50.3.2/3093
20C_8198	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_CLR)	32	R/W	0000_0001h	50.3.2/3093
20C_819C	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_TOG)	32	R/W	0000_0001h	50.3.2/3093
20C_8290	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2)	32	R/W	0000_0000h	50.3.3/3094
20C_8294	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_SET)	32	R/W	0000_0000h	50.3.3/3094
20C_8298	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_CLR)	32	R/W	0000_0000h	50.3.3/3094
20C_829C	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_TOG)	32	R/W	0000_0000h	50.3.3/3094

50.3.1 Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0*n*)

This register defines the basic controls for the temperature sensor minus the frequency of automatic sampling which is defined in the tempsensor.

Address: 20C_8000h base + 180h offset + (4d × i), where i=0d to 3d





TEMPMON_TEMPSENSE0*n* field descriptions

Field	Description
31–20 ALARM_VALUE	This bit field contains the temperature count (raw sensor output) that will generate an alarm interrupt.
19–8 TEMP_CNT	This bit field contains the last measured temperature count.
7 -	This field is reserved. Reserved.
6 -	This field is reserved. Reserved.
5–3 -	This field is reserved. Reserved
2 FINISHED	Indicates that the latest temp is valid. This bit should be cleared by the sensor after the start of each measurement. 0 **INVALID** — Last measurement is not ready yet. 1 **VALID** — Last measurement is valid.
1 MEASURE_ TEMP	Starts the measurement process. If the measurement frequency is zero in the TEMPSENSE1 register, this results in a single conversion.

Table continues on the next page...

TEMPMON_TEMPSENSE0*n* field descriptions (continued)

Field	Description
	0 **STOP** — Do not start the measurement process. 1 **START** — Start the measurement process.
0 POWER_DOWN	This bit powers down the temperature sensor. 0 **POWER_UP** — Enable power to the temperature sensor. 1 **POWER_DOWN** — Power down the temperature sensor.

50.3.2 Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1*n*)

This register defines the automatic repeat time of the temperature sensor.

Address: 20C_8000h base + 190h offset + (4d × i), where i=0d to 3d

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

TEMPMON_TEMPSENSE1*n* field descriptions

Field	Description
31–16 -	This field is reserved. Reserved.
MEASURE_ FREQ	This bits determines how many RTC clocks to wait before automatically repeating a temperature measurement. The pause time before remeasuring is the field value multiplied by the RTC period. 0x0000 Defines a single measurement with no repeat. 0x0001 Updates the temperature value at a RTC clock rate. 0x0002 Updates the temperature value at a RTC/2 clock rate. . . . — 0xFFFF Determines a two second sample period with a 32.768KHz RTC clock. Exact timings depend on the accuracy of the RTC clock.

50.3.3 Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2*n*)

This register defines the automatic repeat time of the temperature sensor.

Address: 20C_8000h base + 290h offset + (4d × i), where i=0d to 3d

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

TEMPMON_TEMPSENSE2*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–16 PANIC_ALARM_VALUE	This bit field contains the temperature that will generate a panic interrupt when exceeded by the temperature measurement.
15–12 -	This field is reserved. Reserved.
LOW_ALARM_VALUE	This bit field contains the temperature that will generate a low alarm interrupt when the field is greater than the temperature measurement.

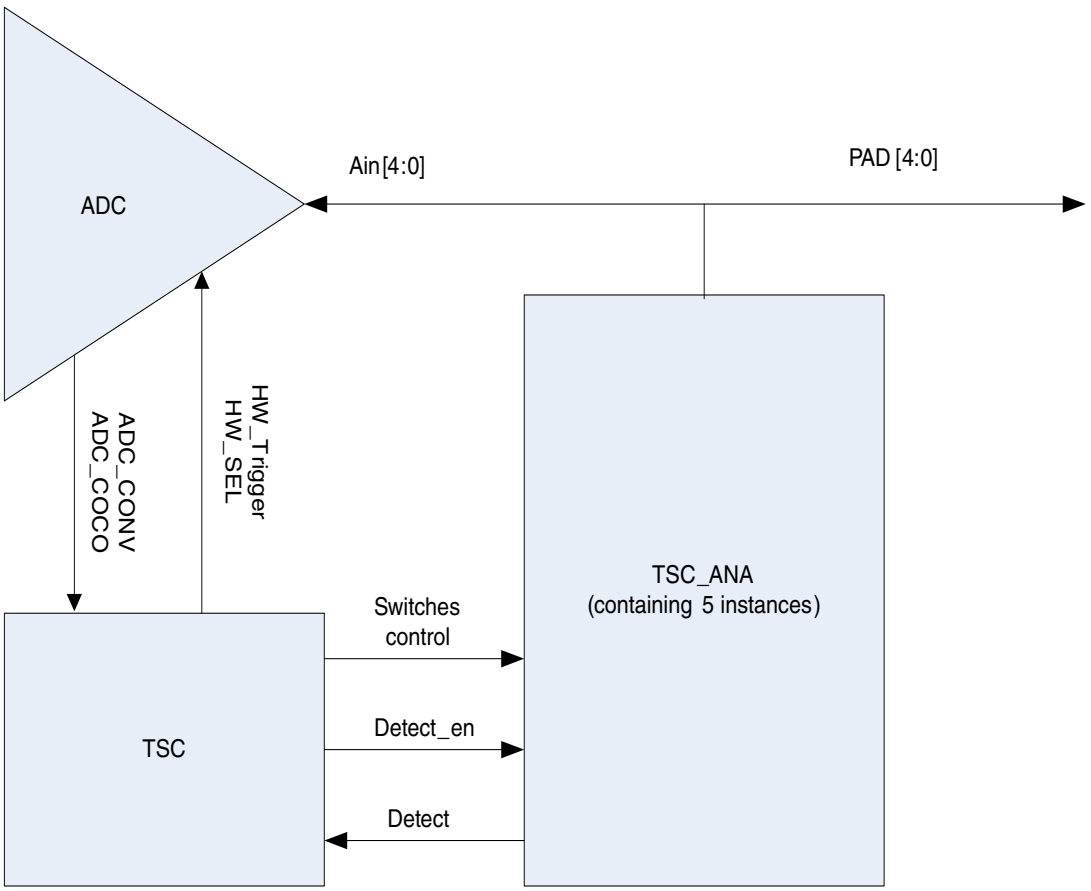
Chapter 51
Touch Screen Controller (TSC)

51.1 Overview

This block describes the Touch Screen Controller (TSC), which is used for ADC and touch screen analogue block.

TSC is responsible for providing control of ADC and touch screen analogue block to form a touch screen system, which achieves function of touch detection and touch location detection. The controller utilizes ADC hardware trigger function and control switches in touch screen analogue block. The controller only supports 4-wire of 5-wire screen touch modes.

Figure 51-1. TSC block diagram



51.1.1 Features

The features of TSC controller are following.

- Configure registers: 32-bit, fully support sky-blue bus interface
- 4-wire or 5-wire mode of touch screen
- Low power wake up functions
- ADC average function and custom 8-bit, 10-bit, and 12-bit conversion result
- Custom pre-charge and de-glitch threshold time setting
- Total control five analogue groups of switches
- Fully asynchronous interface to ADC and analogue switches
- Easy software operation
- Software takes control of operation flow
- Strong debug functions—enable software recognize the IP as a trnasparent box and operation ouput directly
- Software reset function

51.2 Functional Description

This block works with ADC and TSC analogue to form a touch screen system. The system is responsible to detect touch on screen and measure the coordinate of the touch point. The touch screen controller, which is at the heart of the system, given control to TSC analogue and ADC. TSC analogue provides positive or negative voltage to the screen if the touch screen controller provides relevant signals. The ADC convert coordinates value if requested by touch screen controller.

51.2.1 Operating modes

The TSC can be worked in the following modes.

51.2.1.1 Idle

The TSC always stays at idle status if finish a coordinate measurement. The TSC only starts to detect a touch or starts to measure coordinate value if set the start_sen bit. The TSC returns to idle status after finish current task if disable bit is set by software.

51.2.1.2 Pre-charge

Pre-charge is a preparation for detection stage. Before detection stage, the upper layer of screen is required to charge to positive high. The time required for the pre-charge can be set in the pre-charge_timer.

51.2.1.3 Detection

Detection is a stage to sense touch detect on the screen. This stage is initialed a delay (set by pre-charge_timer) in order to wait enough time for the lower screen layer to achieve even-potential status. After the TSC send out detect enable signal to detect signal from analogue, once receiving the signal, TSC starts to measure coordinates or waiting for the instruction from software according to the value of auto_measure. If auto_measure is set then the TSC automatically measures coordinates after detect a touch. Otherwise TSC

waits for software order after detects a touch (and generates an interrupt). The software can choose to drop the measure (and return back to idle) by setting drop_measure, or start the measure by setting start_measure.

51.2.1.4 Measurement

During the measure stage, the TSC hardware gives control to TSC analogue and ADC. No software operation is needed. If requesting average function, do average configuration for ADC.

51.2.1.5 Data valid check

After measure the coordinate value, TSC do a touch detects again. If no touch has been detected, then previous measured coordinates' value is invalid. Otherwise, the measured coordinates' value is valid.

51.2.1.6 Interrupt

Each interrupt provides three software interface bits: interrupt enable, interrupt signal enable, and interrupt signal.

51.2.1.7 Reset

The TSC has two resets: ipg_reset_b and sw_rst. The ipg_reset_b reset is a hardware reset, which resets all registers in TSC block. While the sw_rst is a software reset, which resets every register except ips directly access ones.

51.2.1.8 Debug mode

The TSC provides fully software control signals. Once debug_en has been set, then all TSC outputs will be controlled by software. Software can also observe all TSC inputs through debug interface. Furthermore, the debug registers also provides current state machine states. Software can always check the current hardware state.

51.2.2 Configuration

51.2.2.1 TSC configurations

The following tables provide 4-wire and 5-wire screen touch modes.

Table 51-1. 4-wire screen touch mode

	wiper	ynlr	ypll	xnur	xpul
X measurement	OFF	OFF	OFF	LOW	HIGH
Y measurement	OFF	LOW	HIGH	OFF	OFF
Pre-charge	OFF	OFF	OFF	HIGH (200K, PULL UP)	HIGH (200K, PULL UP)
Intermediate	OFF	OFF	OFF	OFF	HIGH (200K)
Touch screen dection	OFF	LOW	LOW	OFF	HIGH (200K)

Table 51-2. 5-wire screen touch mode

	wiper	ynlr	ypll	xnur	xpul
X measurement	OFF	LOW	HIGH	LOW	HIGH
Y measurement	OFF	LOW	LOW	HIGH	HIGH
Pre-charge	HIGH (200K, PULL UP)	OFF	OFF	OFF	OFF
Intermediate[1]	HIGH (200K)	OFF	OFF	OFF	OFF
Touch screen dection	HIGH (200K)	LOW	LOW	OFF	OFF

1. All other intermediate states switche to OFF.

51.2.2.2 TSC-ADC-TSC analogue configuration

The touch screen controller needs to co-work with ADC and TSC analogue.

The ADC is responsible for analogue value conversion. Following tables show the channels that the ADC used.

Table 51-3. 4-wire screen touch mode

X measurement	Channel 1	Connects to TSC ana YNLR

Table continues on the next page...

Table 51-3. 4-wire screen touch mode (continued)

Y measurement	Channel 3	Connects to TSC ana YNUR

Table 51-4. 5-wire screen touch mode

X measurement	Channel 0	Connects to TSC ana WIPER
Y measurement	Channel 0	Connects to TSC ana WIPER

51.2.2.3 TSC, TSC analogue and ADC connection

The following table describes the connections among the TSC, TSC analogue, and ADC.

Table 51-5. TSC, TSC analogue and ADC connection

TSC controller ports	TSC analogue IP ports	TSC analogue system integration instance Name	ADC Ain ports
wiper_pull_up	pu_en	1	Ain [0]
wiper-200k_pull_up	pull_up-200k_en		
wiper_pull_down	pd_en		
ynlr_pull_up	pu_en	2	Ain [1]
ynlr_200k_pull_up	pull_up_200k_en		
ynlr_pull_down	pd_en		
ypll_pull_up	pu_en	3	Ain [2]
ypll_200k_pull_up	pull_up_200k_en		
ypll_pull_down	pd_en		
xnur_pull_up	pu_en	4	Ain [3]
xnur_200k_pull_up	pull_up_200k_en		
xnur_pull_down	pd_en		
xpul_pull_up	pu_en	5	Ain [4]
xpul_200k_pull_up	pull_up_200k_en		
xpul_pull_down	pd_en		

51.2.2.4 TSC and GPIO

From TSC point of view, it does not care what kinds of PAD used. It can use digital pad or analogue pad.

Table 51-6. The ports used in TSC and GPIO

TSC function ports	TSC analogue instance	ADC Ain pin	GPIO ports
wiper	1	Ain [0]	GPIO1_IO00
ynlr	2	Ain [1]	GPIO1_IO01
ypll	3	Ain [2]	GPIO1_IO02
xnur	4	Ain [3]	GPIO1_IO03
xpul	5	Ain [4]	GPIO1_IO04

NOTE

For the PAD used for TSC, make sure the PAD behavior , such as a wire only connects to the Pin and TSC analogue switches.

For the PAD used for GPIO, for example, has a keeper to prevent the GPIO pad working from a wire only connects the Pin and TSC. In such condition, disable keeper when TSC working.

51.2.2.5 ADC-TSC co-working

The TSC is designed to work with the ADC. Two IP must work together to achieve the correct functions. The coordinate work includes two steps:

1. TSC starts ADC

   TSC sends two signals: trigger and hardware trigger select signal: HWTS[4:0].

   ADC regards triggers as clock to capture the select signal: HWTS. The details of information about trigger and HWTS is show in the ADC hardware trigger chapter. The TSC hardware makes sure that the HWTS signals ready one clock cycles before sending trigger signals. Make sure that the trigger delay is less than HWTS delay + one clock cycle.

   The TSC send HWTS as in the following table:

	TSC 4-wire mode	TSC 5-wire mode
x-coordinate measure	HWTS = 5'b01000	HWTS = 5'b10000
y-coordinate measure	HWTS = 5'b00010	HWTS = 5'b10000

TSC HWTS[4:0] connects to the ADC HWTS[4:0]. The integration change will lead change in driver implementation. On ADC side, HWTS = 1 << x indicates the x logic channel is selected to start hardware ADC conversion. Configure ADC_HCx to configure x logic channel and conversion will store in ADC_Rx.

Each ADC hardware trigger logic channel can be configured to one of many physical channels. The details of the ADC configuration, refer to the ADC hardware trigger chapter. Software driver configures the physical ADC. A sample configuration is shown in the following table.

	TSC 4-wire mode	TSC 5-wire mode
x-coordinate measure	ADC measures channel 1 or 2, so configure ADC_HC3[ADCH] to 1 or 2.	ADC measure channel 0, so configure ADC_HC[4] to 0.
y-coordinate measure	ADC measures channel3 or 4, so configure ADC_HC1[ADCH] to 3 or 4.	ADC measures channel 0, so configure ADC_HC[4] to 0.

Before TSC starts work, software driver configure ADC_HCx. TSC hardware automatically sends trigger and HWTS at the correct time.

2. ADC sends back converted value and complete signals to TSC.
When ADC finishes conversion, the ADC would send a complete signal to TSC. A set of handshake signals between ADC and TSC makes sure that the TSC could receive complete signals, store correct ADC converted value, and clear ADC coco signals at correct order. All parts of this step are automatically done by hardware.

NOTE

For different ADC conversion modes, such 8-bit, 10-bit, 12-bit mode, average mode, please configure in ADC registers.

51.3 TSC Memory Map/Register Definition

The TSC Memory Map and Register Definition is described in the following section.

TSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
204_0000	PS Input Buffer Address (TSC_BASIC_SETTING)	32	R/W	0000_0000h	51.3.1/3103
204_0010	PS Input Buffer Address (TSC_PS_INPUT_BUFFER_ADDR)	32	R/W	0000_0000h	51.3.2/3104

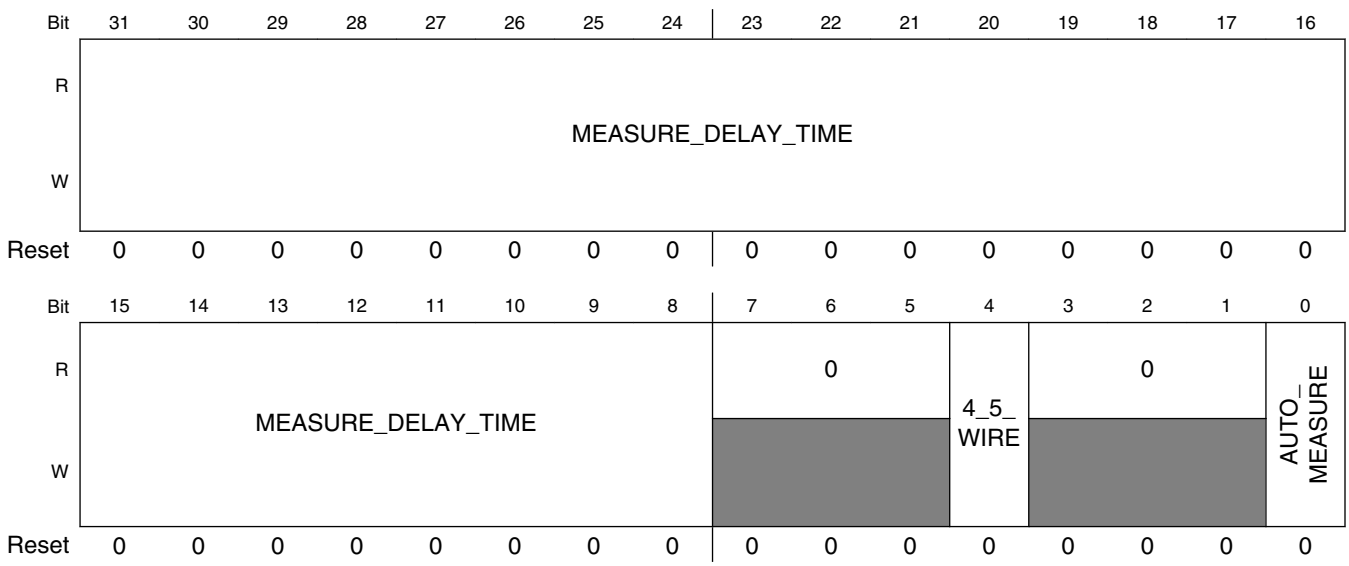
Table continues on the next page...

TSC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
204_0020	Flow Control (TSC_FLOW_CONTROL)	32	R/W	0000_0000h	51.3.3/3105
204_0030	Measure Value (TSC_MEASEURE_VALUE)	32	R	0000_0000h	51.3.4/3106
204_0040	Interrupt Enable (TSC_INT_EN)	32	R/W	0000_0000h	51.3.5/3107
204_0050	Interrupt Signal Enable (TSC_INT_SIG_EN)	32	R/W	0000_0000h	51.3.6/3108
204_0060	Intterrupt Status (TSC_INT_STATUS)	32	R/W	0000_0000h	51.3.7/3109
204_0070	TSC_DEBUG_MODE	32	R/W	0000_0000h	51.3.8/3111
204_0080	TSC_DEBUG_MODE2	32	R/W	0000_0000h	51.3.9/3113

51.3.1 PS Input Buffer Address (TSC_BASIC_SETTING)

Address: 204_0000h base + 0h offset = 204_0000h



TSC_BASIC_SETTING field descriptions

Field	Description
31–8 MEASURE_ DELAY_TIME	Measure Delay Time Before X-axis or Y-axis measurement, the screen need some time before even potential distribution ready.
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 4_5_WIRE	4/5 Wire detection 4-Wire Detection Mode or 5-Wire Detection Mode 0 4-Wire Detection Mode 1 5-Wire Detection Mode

Table continues on the next page...

TSC_BASIC_SETTING field descriptions (continued)

Field	Description
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 AUTO_ MEASURE	Auto Measure This field indicates after detect touch, whether automatic start measurement. SW must make sure that the detect interrupt should be disable. This bit is not HW self-clear. 0 Disable Auto Measure 1 Auto Measure

51.3.2 PS Input Buffer Address (TSC_PS_INPUT_BUFFER_ADDR)

Address: 204_0000h base + 10h offset = 204_0010h

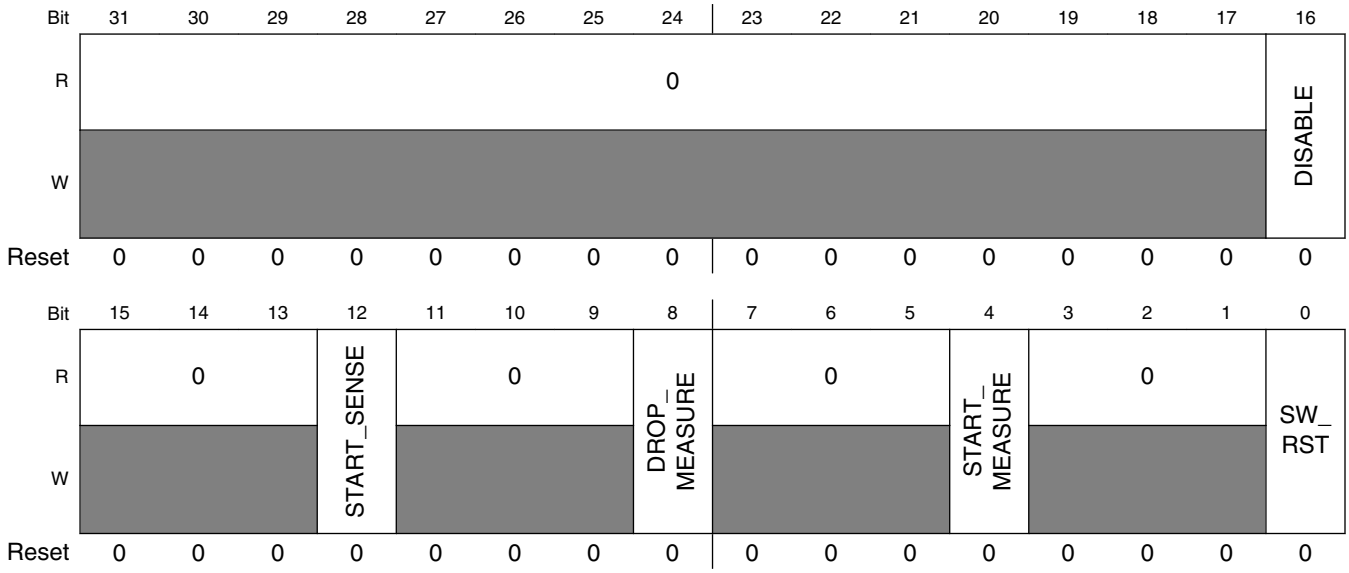
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRE_CHARGE_TIME																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TSC_PS_INPUT_BUFFER_ADDR field descriptions

Field	Description
PRE_CHARGE_ TIME	Auto Measure This field indicates after detect touch, whether automatic start measurement. SW must make sure that the detect interrupt should be disable. This bit is not HW self-clear. 0 Disable Auto Measure 1 Auto Measure

51.3.3 Flow Control (TSC_FLOW_CONTROL)

Address: 204_0000h base + 20h offset = 204_0020h



TSC_FLOW_CONTROL field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 DISABLE	This bit is for SW disable registers. After set this bit, the hardware returns to idle status after finish the current state operation. SW must check the SW_IDLE bit to confirm that the controller has return to idle status. It's a HW self-clean bit. 0 Leave HW state machine control 1 SW set to idle status
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 START_SENSE	Start Sense It's a HW self-clean bit. 0 Stay at idle status 1 Start sense detection and (if auto_measure set to 1) measure after detect a touch
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 DROP_ MEASURE	Drop Measure This field indicates whether start measure X/Y coordinate value after detect a touch It's a HW self-clean bit. 0 Do not drop measure for now 1 Drop the measure and controller return to idle status

Table continues on the next page...

TSC_FLOW_CONTROL field descriptions (continued)

Field	Description
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 START_ MEASURE	Start Measure This field indicates whether start measure X/Y coordinate value after detect a touch It's a self-clean bit. 0 Do not start measure for now 1 Start measure the X/Y coordinate value
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 SW_RST	Soft Reset This is a synchronization reset, which reset all HW registers. **NOTE:** All SW accessible registers would keep as it original setting Software reset would be self-clear.

51.3.4 Measure Value (TSC_MEASEURE_VALUE)

Address: 204_0000h base + 30h offset = 204_0030h

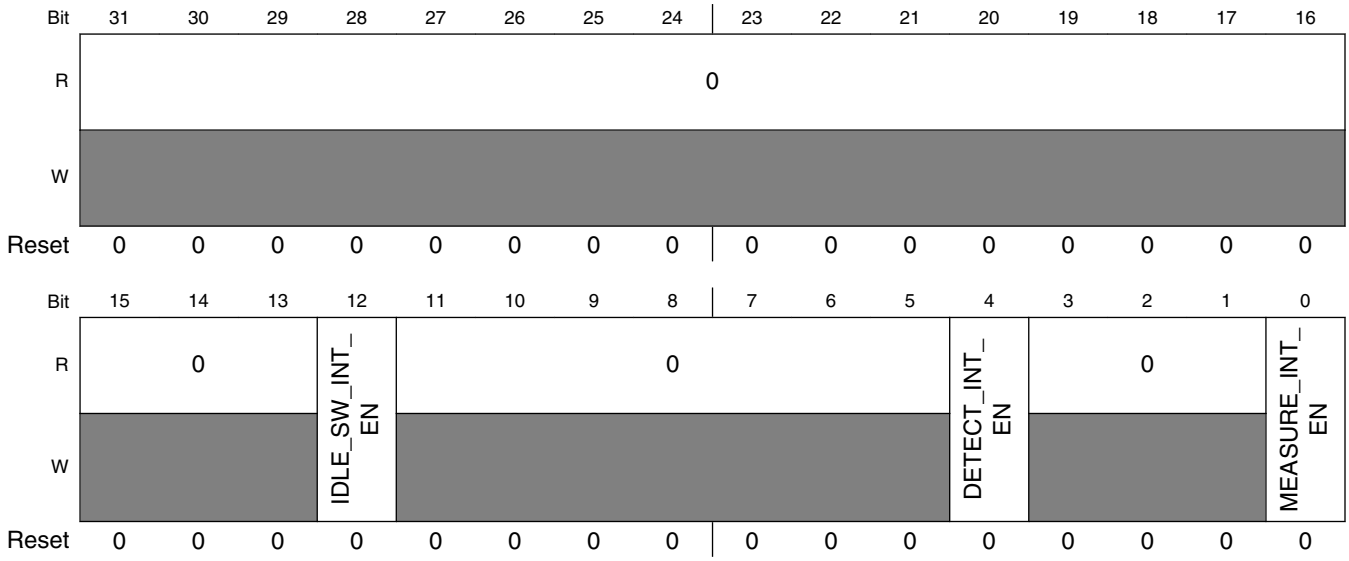
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				X_VALUE												0				Y_VALUE											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TSC_MEASEURE_VALUE field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27–16 X_VALUE	X Value Coordinate value **NOTE:** It is an ADC conversion value, SW need to convert it to fit screen size.
15–12 Reserved	This read-only field is reserved and always has the value 0.
Y_VALUE	Y Value Y coordinate value, note, it is an ADC conversion value, SW need to convert it to fit screen size

51.3.5 Interrupt Enable (TSC_INT_EN)

Address: 204_0000h base + 40h offset = 204_0040h

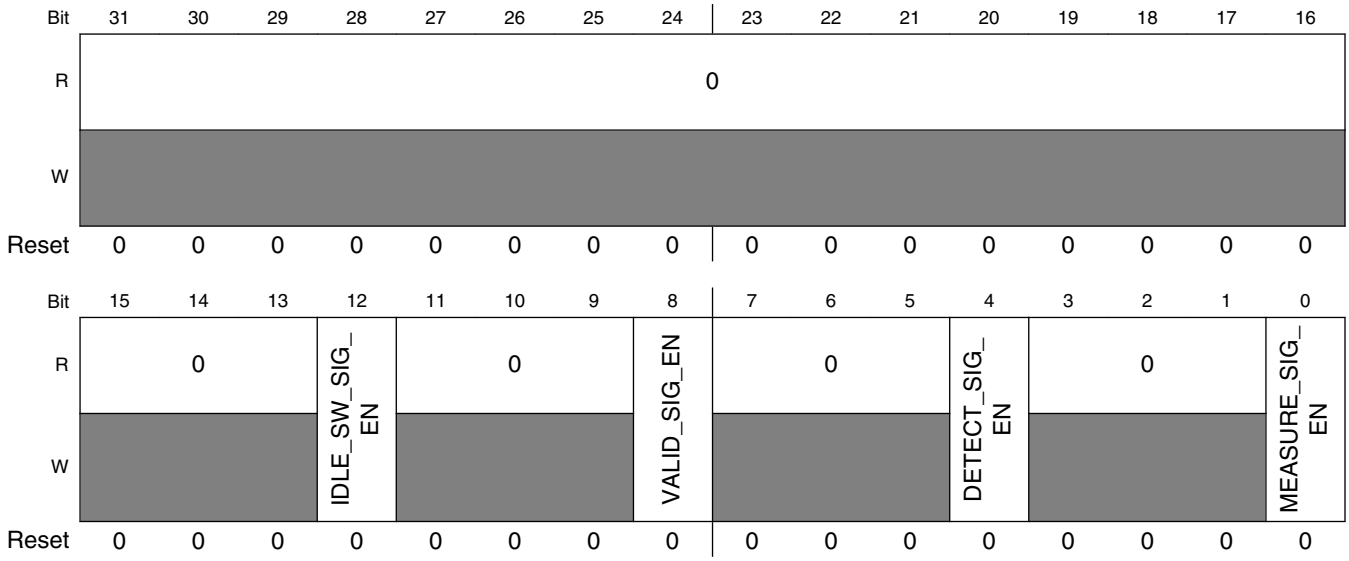


TSC_INT_EN field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 IDLE_SW_INT_EN	Idle Software Interrupt Enable This field indicates whether enable the software return to idle status interrupt. This interrupt is generated if the controller has return to idle status. This bit is only valid if disable bit in flow control register set to 1'b1. 0 Disable idle software interrupt 1 Enable idle software interrupt
11–5 Reserved	Reserved This read-only field is reserved and always has the value 0.
4 DETECT_INT_EN	Detect Interrupt Enable This field indicates whether enable the detect interrupt. This interrupt is generated if there is a touch detect after pre-charge ready. 0 Disable detect interrupt 1 Enable detect interrupt
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 MEASURE_INT_EN	Measure Interrupt Enable This field indicates whether enable the measure interrupt. This interrupt is generated after the touch detection which follows measurement. 0 Disable measure

51.3.6 Interrupt Signal Enable (TSC_INT_SIG_EN)

Address: 204_0000h base + 50h offset = 204_0050h



TSC_INT_SIG_EN field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 IDLE_SW_SIG_EN	Idle Software Signal Enable This field indicates whether enable the software return to idle status. This signal is generated if the controller has return to idle status. This bit is only valid if disable bit in flow control register set to 1'b1. 0 Disable idle software signal 1 Enable idle software signal
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 VALID_SIG_EN	Valid Signal Enable This field indicates whether enable the valid signal. This field generated at the same time with MEASURE_SIG_EN. 0 Disable valid signal 1 Enable valid signal
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 DETECT_SIG_EN	Detect Signal Enable This field indicates whether enable the detect signal. 0 Disable detect signal 1 Enable detect signal

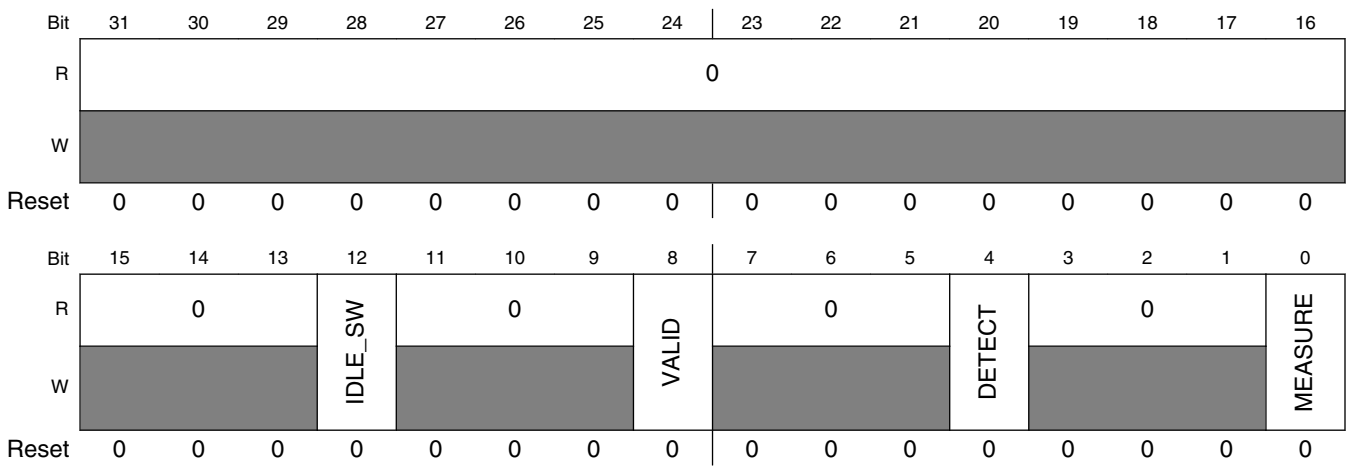
Table continues on the next page...

TSC_INT_SIG_EN field descriptions (continued)

Field	Description
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 MEASURE_SIG_EN	Measure Signal Enable This field indicates whether enable the measure signal.

51.3.7 Intterrupt Status (TSC_INT_STATUS)

Address: 204_0000h base + 60h offset = 204_0060h



TSC_INT_STATUS field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 IDLE_SW	Idle Software This field indicates whether the state machine return to idle status. This signal is generated if the controller has return to idle status. This bit is only valid if disable bit in flow control register set to 1'b1. 0 Haven't return to idle status 1 Already return to idle status
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 VALID	Valid Signal This field indicats whether the measure value is valid. This field generated at the same time with MEASURE_SIG. 0 There is no touch detected after measurement, indicates that the measured value is not valid 1 There is touch detection after measurement, indicates that the measure is valid
7–5 Reserved	This read-only field is reserved and always has the value 0.

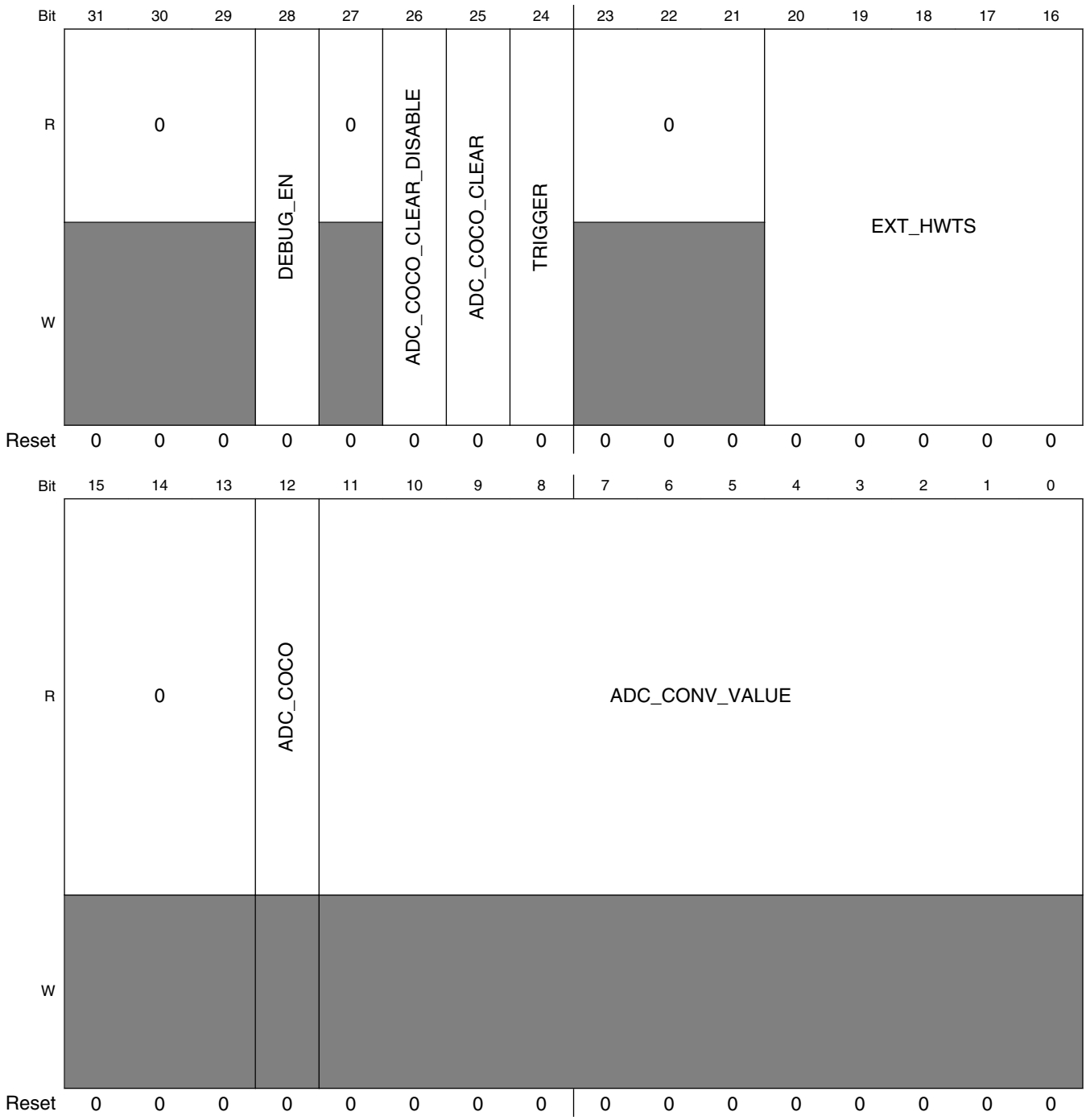
Table continues on the next page...

TSC_INT_STATUS field descriptions (continued)

Field	Description
4 DETECT	Detect Signal This field indicates whether there is a detect signal. 0 Does not exist a detect signal 1 Exist detect signal
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 MEASURE	Measure Signal This field indicates whether there is a measure signal. 0 Does not exist a measure signal 1 Exist a measure signal

51.3.8 TSC_DEBUG_MODE

Address: 204_0000h base + 70h offset = 204_0070h

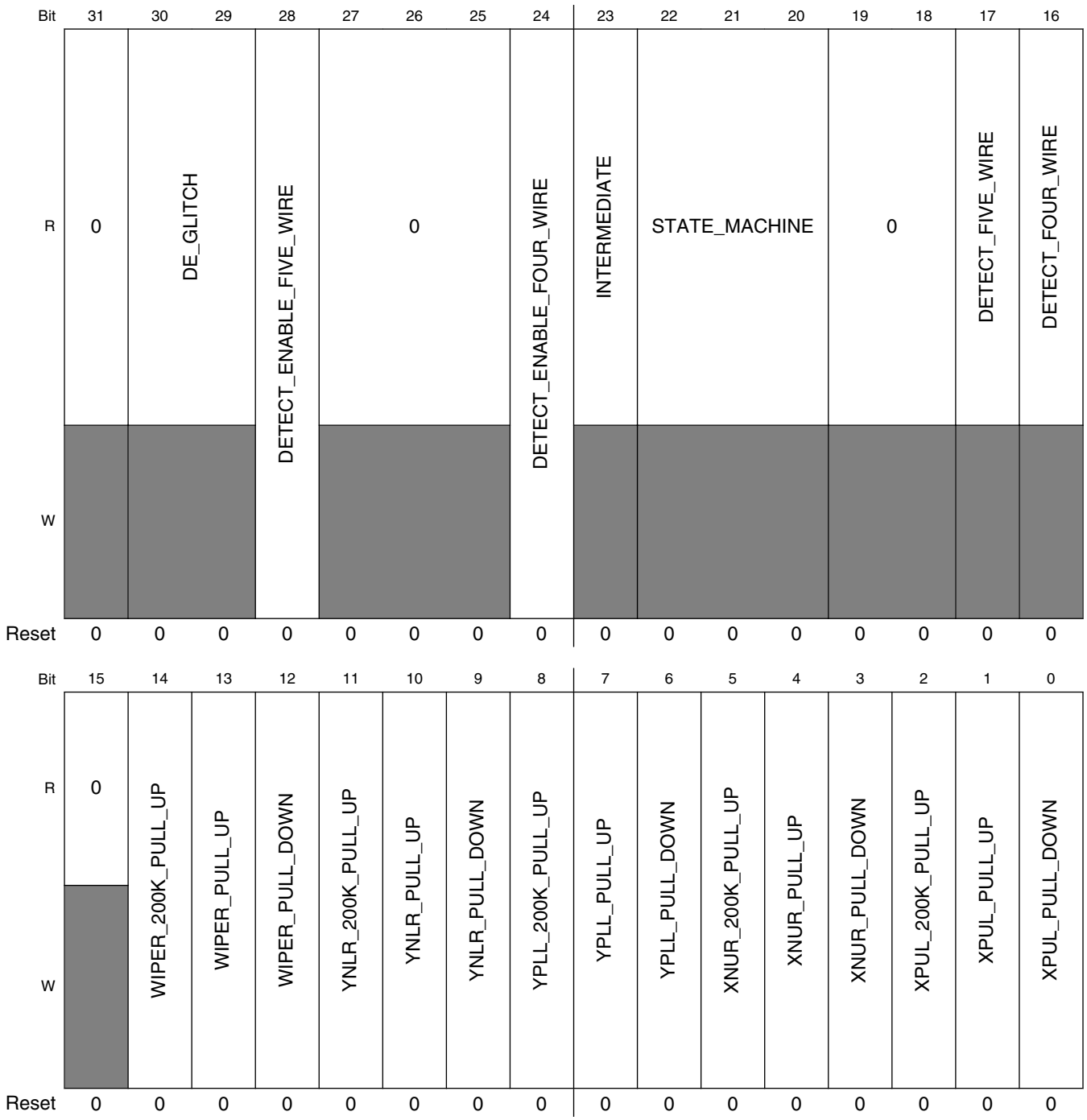


TSC_DEBUG_MODE field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DEBUG_EN	Debug Enable The RO registers in debug_mode and debug_mode2 always reflect the latest value and is not controlled by this bit. All RW SW bits are controlled by this bit. 0 Enable debug mode 1 Disable debug mode
27 Reserved	This read-only field is reserved and always has the value 0.
26 ADC_COCO_ CLEAR_ DISABLE	ADC COCO Clear Disable This bit could prevent TSC HW generates an ADC COCO clear signal. **NOTE:** This bit does not effect ADC_COCO_CLEAR bit. 0 Allow TSC hardware generates ADC COCO clear 1 Prevent TSC from generate ADC COCO clear signal — —
25 ADC_COCO_ CLEAR	ADC Coco Clear Original ADC coco only clear is system read conv result from IPS bus. However, TSC read conv result directly from ADC to lower SW load, so that ADC requires TSC generates a clear signal. 0 No ADC COCO clear 1 Set ADC COCO clear
24 TRIGGER	Trigger Hardware trigger signal to ADC 0 No hardware trigger signal 1 Hardware trigger signal, the signal must last at least 1 ips clock period
23–21 Reserved	This read-only field is reserved and always has the value 0.
20–16 EXT_HWTS	Hardware Trigger Select Signal Hardware trigger select signal, select which channel to start conversion.
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 ADC_COCO	ADC COCO Signal This signal is generated by ADC.
ADC_CONV_ VALUE	ADC Conversion Value This signal is generated by ADC.

51.3.9 TSC_DEBUG_MODE2

Address: 204_0000h base + 80h offset = 204_0080h



TSC_DEBUG_MODE2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–29 DE_GLITCH	This field indicates glitch threshold. The threshold is defined by number of clock cycles. A detect signal is only valid if it is exist longer than threshold. Any signal transfer through detect and length is smaller than threshold is regards as a glitch. 00 • Normal function: 0x1fff ipg clock cycles; • Low power mode: 0x9 low power clock cycles 01 • Normal function: 0xfff ipg clock cycles; • Low power mode: :0x7 low power clock cycles 10 • Normal function: 0x7ff ipg clock cycles; • Low power mode:0x5 low power clock cycles 11 • Normal function: 0x3 ipg clock cycles; • Low power mode:0x3 low power clock cycles
28 DETECT_ ENABLE_FIVE_ WIRE	Detect Enable Five Wire 0 Do not read five wire detect value, read default value from analogue 1 Read five wire detect status from analogue
27–25 Reserved	This read-only field is reserved and always has the value 0.
24 DETECT_ ENABLE_FOUR_ WIRE	Detect Enable Four Wire 0 Do not read four wire detect value, read default value from analogue 1 Read four wire detect status from analogue
23 INTERMEDIATE	Intermediate State It's an intermediate state, between two state machine states, in order to provide better timing for analogue. It lasts for only once cycle. Intermediate state exists after pre-charge, detect, x- measure, y-measure, and 2nd-pre-charge, 2nd-detect. Note, the intermediate after pre-charge and 2nd-pre-charge is different. 0 Not in intermedia 1 Intermedia
22–20 STATE_ MACHINE	State Machine 000 Idle 001 Pre-charge 010 Detect 011 X-measure 100 Y-measure 101 Pre-charge 110 Detect
19–18 Reserved	This read-only field is reserved and always has the value 0.
17 DETECT_FIVE_ WIRE	Detect Five Wire This field in only valid when the touch controller is under detect mode. That is not in idle. **NOTE:** It is in measure mode, not per-charge mode. This is an asynchronous signal.

Table continues on the next page...

TSC_DEBUG_MODE2 field descriptions (continued)

Field	Description
	0 No detect signal 1 Yes, there is a detect on the touch screen.
16 DETECT_FOUR_WIRE	Detect Four Wire This field in only valid when the touch controller is under detect mode. That is not in idle. **NOTE:** It is in measure mode, not per-charge mode. This is an asynchronous signal. 0 No detect signal 1 Yes, there is a detect on the touch screen.
15 Reserved	This read-only field is reserved and always has the value 0.
14 WIPER_200K_PULL_UP	Wiper Wire 200K Pull Up Switch This is a control signal of this wiper wire 200k pull up switch. 0 Close the switch 1 Open up the switch
13 WIPER_PULL_UP	Wiper Wire Pull Up Switch This is a control signal of this switch. 0 Close the switch 1 Open up the switch
12 WIPER_PULL_DOWN	Wiper Wire Pull Down Switch This is a control signal of this wiper wire pull down switch. 0 Close the switch 1 Open up the switch
11 YNLR_200K_PULL_UP	YNLR Wire 200K Pull Up Switch This is a control signal of this YNLR wire 200K pull up switch. 0 Close the switch 1 Open up the switch
10 YNLR_PULL_UP	YNLR Wire Pull Up Switch This is a control signal of this YNLR wire pull up switch. 0 Close the switch 1 Open up the switch
9 YNLR_PULL_DOWN	YNLR Wire Pull Down Switch This is a control signal of this YNLR wire pull down switch. 0 Close the switch 1 Open up the switch
8 YPLL_200K_PULL_UP	YPLL Wire 200K Pull Up Switch This is a control signal of this YPLL wire 200k pull up switch.

Table continues on the next page...

TSC_DEBUG_MODE2 field descriptions (continued)

Field	Description
	0 Close the switch 1 Open up the switch
7 YPLL_PULL_UP	YPLL Wire Pull Up Switch This is a control signal of this YPLL wire pull up switch. 0 Close the switch 1 Open the switch
6 YPLL_PULL_DOWN	YPLL Wire Pull Down Switch This is a control signal of this YPLL wire pull down switch. 0 Close the switch 1 Open up the switch
5 XNUR_200K_PULL_UP	XNUR Wire 200K Pull Up Switch This is a control signal of this XNUR wire 200K pull up switch. 0 Close the switch 1 Open up the switch
4 XNUR_PULL_UP	XNUR Wire Pull Up Switch This is a control signal of this XNUR Wire Pull Up Switch. 0 Close the switch 1 Open up the switch
3 XNUR_PULL_DOWN	XNUR Wire Pull Down Switch This is a control signal of this XNUR Wire Pull Down Switch. 0 Close the switch 1 Open up the switch
2 XPUL_200K_PULL_UP	XPUL Wire 200K Pull Up Switch This is a control signal of this XPUL Wire 200K Pull Up Switch. 0 Close the switch 1 Open up the switch
1 XPUL_PULL_UP	XPUL Wire Pull Up Switch This is a control signal of this XPUL Wire Pull Up Switch. 0 Close the switch 1 Open up the switch
0 XPUL_PULL_DOWN	XPUL Wire Pull Down Switch This is a control signal of this XPUL wire pull down switch. 0 Close the switch 1 Open up the switch

Chapter 52
TrustZone Address Space Controller (TZASC)

52.1 Overview

The TrustZone Address Space Controller (TZASC) protects security-sensitive SW and data in a trusted execution environment against potentially compromised SW running on the platform.

The TZASC block diagram is shown in figure below.

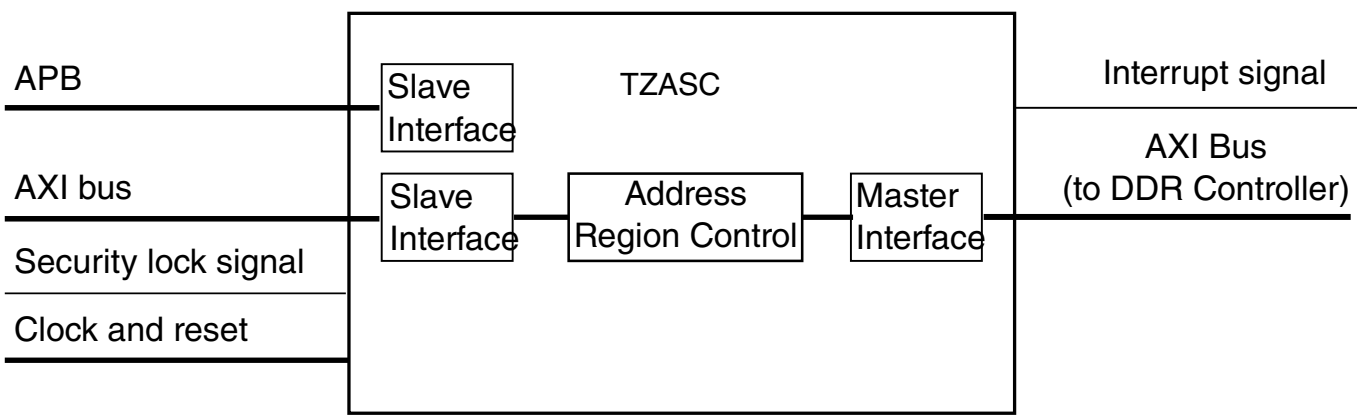


Figure 52-1. TZASC Block Diagram

The TZASC is an IP by ARM ("CoreLink™ TrustZone Address Space Controller TZC-380"), designed to provide configurable protection over program (SW) memory space.

The main features of TZASC are:

- Supports 16 independent address regions
- Access controls are independently programmable for each address region
- Sensitive registers may be locked
- Host interrupt may be programmed to signal attempted access control violations

- AXI master/slave interfaces for transactions
- APB slave interface for configuration and status reporting

52.2 Clocks

The table found here describes the clock sources for TZASC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 52-1. TZASC Clocks

Clock name	Clock Root	Description
aclk	mmdc_axi_clk_root	Module clock

52.3 Address Mapping in various memory mapping modes

The address configured to the TZASC controller(s) must match the "local addresses" as being passed on to the DDR controller(s).

Memory "aliasing" implications on TZASC settings - in systems which does not utilize the maximal supported DDR space the controller is designed for, the whole DDR memory map becomes "aliased" (replicated) by the size of the physical memory used. In such cases, the TZASC must be configured to protect all aliased regions as well (i.e. effectively reducing the number of available TZASC regions, since all aliased regions must be handled, for each "real" space needing protection).

For complete details on TZASC functionality and the programming model, see the ARM document, “CoreLink™ TrustZone Address Space Controller TZC-380 Technical Reference Manual, (Rev r0p1 or newer)”, available at **http://infocenter.arm.com**.

Chapter 53
Universal Asynchronous Receiver/Transmitter (UART)

53.1 Overview

Universal Asynchronous Receiver/Transmitter (UART) provides serial communication capability with external devices through a level converter and an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility.

UART supports NRZ encoding format , RS485 compatible 9 bit data format and IrDA-compatible infrared slow data rate (SIR) format.

Figure 53-1 is the UART block diagram.

The "Module Clock" is the UART_CLK which comes from CCM. The "Peripheral Clock" is the IPG_CLK which comes from CCM.

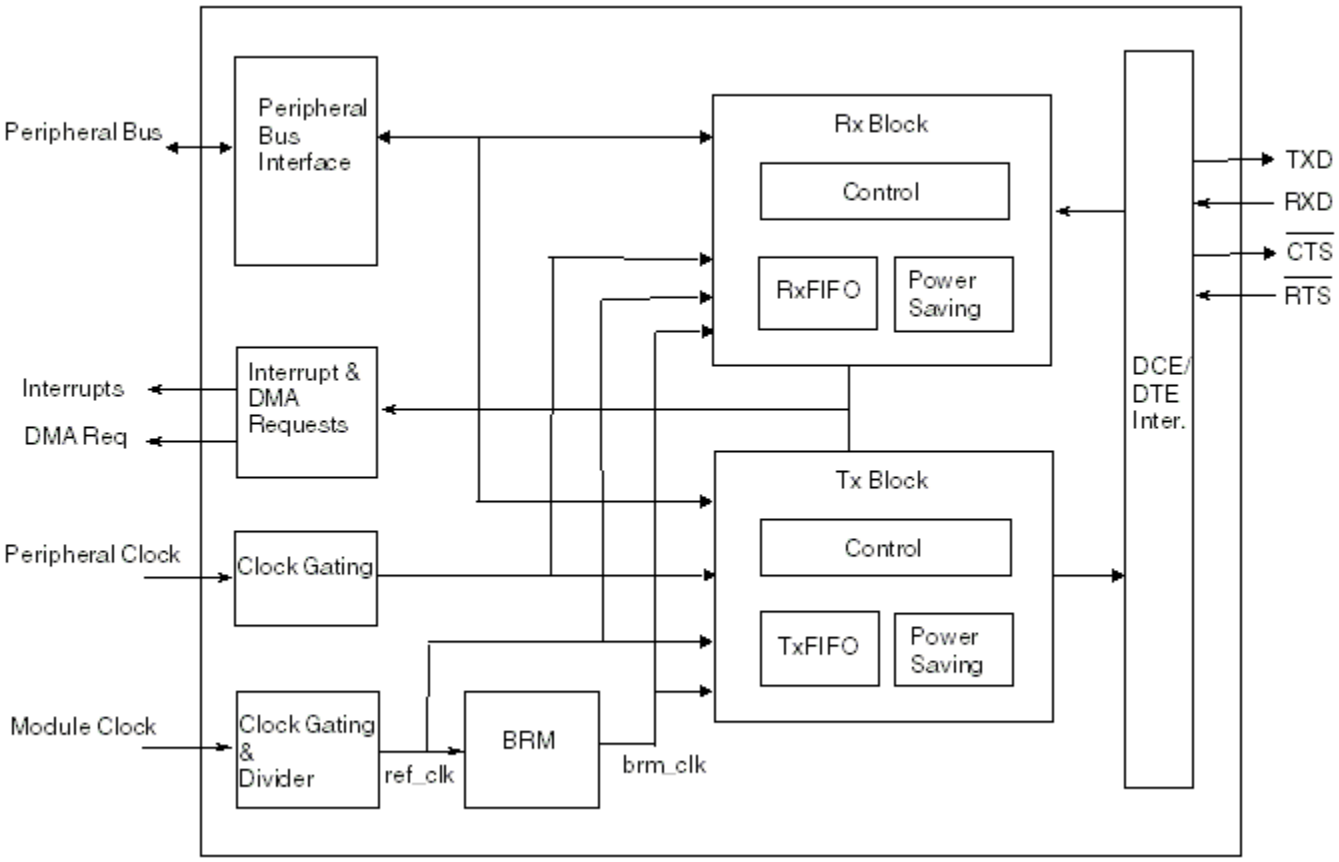


Figure 53-1. UART Block Diagram

53.1.1 Features

The UART includes the following features:

- High-speed TIA/EIA-232-F compatible, up to 5.0 Mbit/s
- Serial IR interface low-speed, IrDA-compatible (up to 115.2 Kbit/s)
- 9-bit or Multidrop mode (RS-485) support (automatic slave address detection)
- 7 or 8 data bits for RS-232 characters, or 9 bit RS-485 format
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Hardware flow control support for request to send (RTS_B) and clear to send (CTS_B) signals
- RS-485 driver direction control via CTS_B signal
- Edge-selectable RTS_B and edge-detect interrupts
- Status flags for various flow control and FIFO states
- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression

- UART internal clocks enable/disable
- Auto baud rate detection (up to 115.2 Kbit/s)
- Receiver and transmitter enable/disable for power saving
- RX_DATA input and TX_DATA output can be inverted respectively in RS-232/ RS-485 mode
- DCE/DTE capability
- RTS_B, IrDA asynchronous wake (AIRINT), receive asynchronous wake (AWAKE) interrupts wake the processor from STOP mode
- Maskable interrupts
- Two DMA Requests (TxFIFO DMA Request and RxFIFO DMA Request)
- Escape character sequence detection
- Software reset (SRST_B)
- Two independent, 32-entry FIFOs for transmit and receive
- The peripheral clock can be totally asynchronous with the module clock. The module clock determines baud rate. This allows frequency scaling on peripheral clock (such as during DVFS mode) while remaining the module clock frequency and baud rate.

53.1.2 Modes of operation

- Serial RS-232NRZ mode
- 9-bit RS-485 mode
- IrDA mode

To set UART in different modes, see the table below.

Table 53-1. UART mode definition

MDEN (UMCR[0])	IREN (UCR1[7])	UART Mode	Descriptioin
0	0	RS-232	RXD/TXD data is serial RS-232 NRZ format
0	1	IrDA (Interface)	RXD/TXD data is IrDA-compatible infrared slow data rate (SIR) format
1	0	RS-485	RXD/TXD data is RS485 compatible 9 bit data format
1	1	Undefined	Undefined

53.2 External Signals

The chip-level IOMUX modifies the direction and routing of the UART signals based on whether the UART is operating in DCE mode (UARTn_UFCR[DCEDTE]=0) or DTE mode (UARTn_UFCR[DCEDTE]=1). The routing of the external signals to the UART module is shown in the figure below.

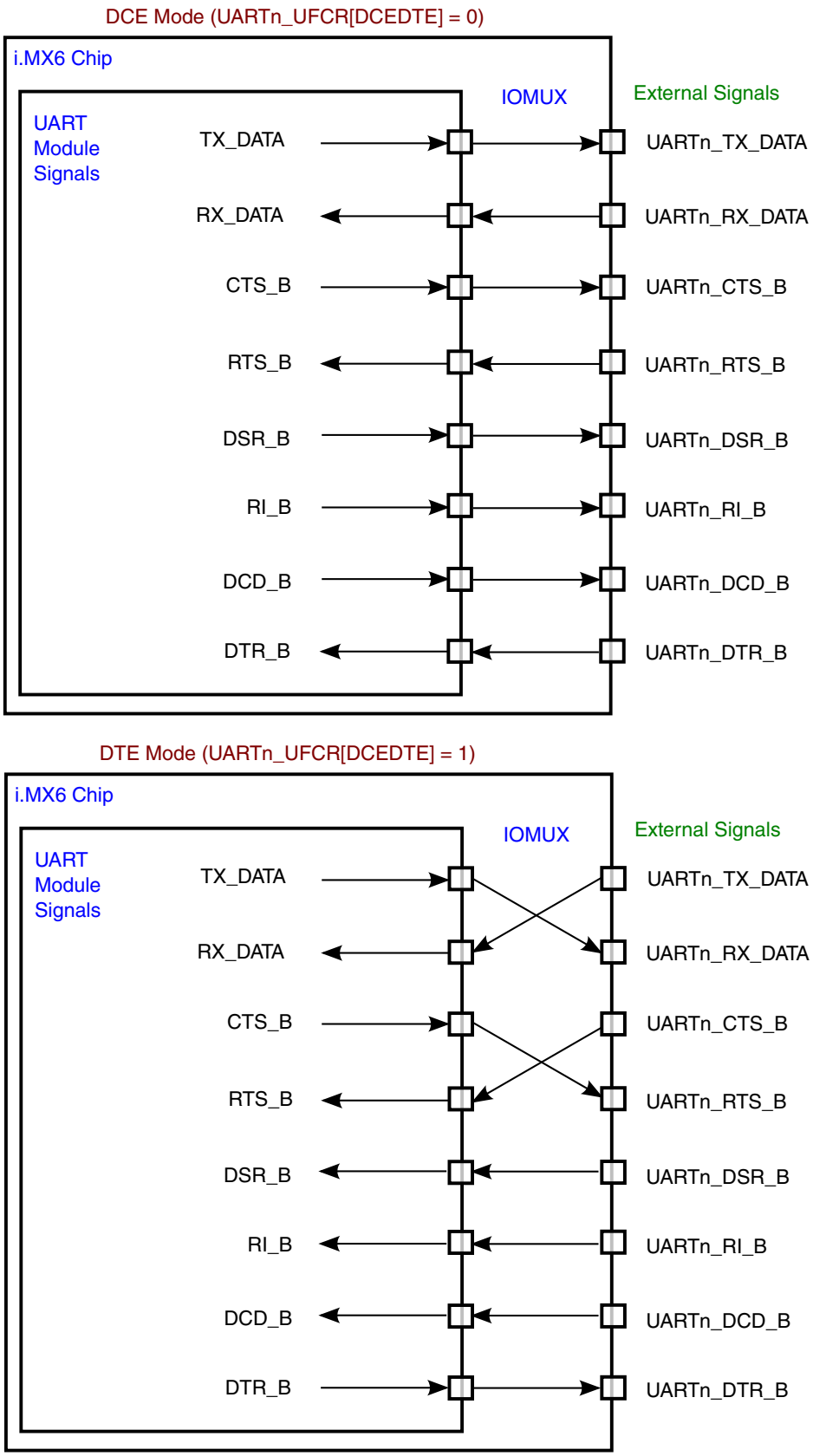


Figure 53-2. UART external signals to module signals routing with respect to DCE/DTE mode

The following table describes the external signals of UART:

Table 53-2. UART External Signals

Signal	Description	Pad	Mode	Direction
UART1_CTS_B	Clear to send	GPIO1_IO06	ALT8	I
		UART1_CTS_B	ALT0	
UART1_RTS_B	Request to send	GPIO1_IO07	ALT8	O
		UART1_RTS_B	ALT0	
UART1_RX_DATA	Serial / infrared data receive	GPIO1_IO03	ALT8	O
		UART1_RX_DATA	ALT0	
UART1_TX_DATA	Serial / infrared data transmit	GPIO1_IO02	ALT8	I
		UART1_TX_DATA	ALT0	
UART2_CTS_B	Clear to send	NAND_DATA06	ALT8	I
		UART2_CTS_B	ALT0	
		UART3_TX_DATA	ALT4	
UART2_RTS_B	Request to send	NAND_DATA07	ALT8	O
		UART2_RTS_B	ALT0	
		UART3_RX_DATA	ALT4	
UART2_RX_DATA	Serial / infrared data receive	NAND_DATA05	ALT8	O
		UART2_RX_DATA	ALT0	
UART2_TX_DATA	Serial / infrared data transmit	NAND_DATA04	ALT8	I
		UART2_TX_DATA	ALT0	
UART3_CTS_B	Clear to send	NAND_CE1_B	ALT8	I
		UART3_CTS_B	ALT0	
UART3_RTS_B	Request to send	NAND_CLE	ALT8	O
		UART3_RTS_B	ALT0	
UART3_RX_DATA	Serial / infrared data receive	NAND_CE0_B	ALT8	O
		UART3_RX_DATA	ALT0	
UART3_TX_DATA	Serial / infrared data transmit	NAND_READY_B	ALT8	I
		UART3_TX_DATA	ALT0	
UART4_CTS_B	Clear to send	ENET1_RX_DATA1	ALT1	I
		LCD_HSYNC	ALT2	
UART4_RTS_B	Request to send	ENET1_RX_DATA0	ALT1	O
		LCD_VSYNC	ALT2	
UART4_RX_DATA	Serial / infrared data receive	LCD_ENABLE	ALT2	O
		UART4_RX_DATA	ALT0	
UART4_TX_DATA	Serial / infrared data transmit	LCD_CLK	ALT2	I
		UART4_TX_DATA	ALT0	
UART5_CTS_B	Clear to send	CSI_DATA03	ALT8	I
		ENET1_TX_DATA0	ALT1	
		GPIO1_IO09	ALT8	
UART5_RTS_B	Request to send	CSI_DATA02	ALT8	O

Table continues on the next page...

Table 53-2. UART External Signals (continued)

Signal	Description	Pad	Mode	Direction
		ENET1_RX_EN	ALT1	
		GPIO1_IO08	ALT8	
UART5_RX_DATA	Serial / infrared data receive	CSI_DATA01	ALT8	O
		GPIO1_IO05	ALT8	
		UART5_RX_DATA	ALT0	
UART5_TX_DATA	Serial / infrared data transmit	CSI_DATA00	ALT8	I
		GPIO1_IO04	ALT8	
		UART5_TX_DATA	ALT0	
UART6_CTS_B	Clear to send	CSI_HSYNC	ALT8	I
		ENET1_TX_DATA1	ALT1	
UART6_RTS_B	Request to send	CSI_VSYNC	ALT8	O
		ENET1_TX_EN	ALT1	
UART6_RX_DATA	Serial / infrared data receive	CSI_PIXCLK	ALT8	O
		ENET2_RX_DATA1	ALT1	
UART6_TX_DATA	Serial / infrared data transmit	CSI_MCLK	ALT8	I
		ENET2_RX_DATA0	ALT1	
UART7_CTS_B	Clear to send	ENET1_TX_CLK	ALT1	I
		LCD_DATA06	ALT1	
UART7_RTS_B	Request to send	ENET1_RX_ER	ALT1	O
		LCD_DATA07	ALT1	
UART7_RX_DATA	Serial / infrared data receive	ENET2_TX_DATA0	ALT1	O
		LCD_DATA17	ALT1	
UART7_TX_DATA	Serial / infrared data transmit	ENET2_RX_EN	ALT1	I
		LCD_DATA16	ALT1	
UART8_CTS_B	Clear to send	ENET2_TX_CLK	ALT1	I
		LCD_DATA04	ALT1	
UART8_RTS_B	Request to send	ENET2_RX_ER	ALT1	O
		LCD_DATA05	ALT1	
UART8_RX_DATA	Serial / infrared data receive	ENET2_TX_EN	ALT1	O
		LCD_DATA21	ALT1	
UART8_TX_DATA	Serial / infrared data transmit	ENET2_TX_DATA1	ALT1	I
		LCD_DATA20	ALT1	

The user must configure the input path to the UART by properly configuring the DAISY bits in the IOMUXC_UARTn_RX_DATA_INPUT and the IOMUXC_UARTn_UART_RTS_B_SELECT_INPUT registers.

For IOMUXC_UARTn_UART_RTS_B_SELECT_INPUT[DAISY]:

- Configurations that select UARTn_RTS_B for the pad are only valid when UARTn_UFCR[DCEDTE]=0 (DCE mode)
- Configurations that select UARTn_CTS_B for the pad are only valid when UARTn_UFCR[DCEDTE]=1 (DTE mode)

For IOMUXC_UARTn_UART_RX_DATA_B_SELECT_INPUT[DAISY]:

- Configurations that select UARTn_RX_DATA for the pad are only valid when UARTn_UFCR[DCEDTE]=0 (DCE mode)
- Configurations that select UARTn_TX_DATA for the pad are only valid when UARTn_UFCR[DCEDTE]=1 (DTE mode)

53.2.1 Detailed Signal Descriptions

53.2.1.1 Interrupt Signals

53.2.1.1.1 *interrupt_uart* - UART Interrupt

Output interrupt request.

53.2.1.2 DMA Request Signals

53.2.1.2.1 *dma_req_rx* - Receiver DMA Request

Output DMA Request signal for receiver interface.

53.2.1.2.2 *dma_req_tx* - Transmitter DMA Request

Output DMA Request signal for transmitter interface. Set at 0 when TXDMAEN (UCR1[3]) is at 1 and TRDY (USR1[13]) is also at 1.

53.2.1.3 Special Signals

53.2.1.3.1 *stop_req* - Stop Mode

Input stop mode. Indicates to UART that ARM platform is going to enter in Stop Mode and clocks are going to stop running.

See Low Power Modes for more information about Stop Mode.

53.2.1.3.2 *doze_req* - Doze Mode

Input doze mode. ARM platform requests UART to switch in doze mode (power saving mode).

See Low Power Modes for more information about Doze Mode.

53.2.1.3.3 *debug_req* - Debug Mode

Input debug mode. Indicates UART it has to enter in debug mode.

See UART Operation in System Debug State, for more information about Debug Mode.

53.3 Clocks

The table found here describes the clock sources for UART.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 53-3. UART Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
ipg_perclk	uart_clk_root	Module clock

53.4 Functional Description

This section provides a complete functional description of the block.

53.4.1 Interrupts and DMA Requests

See the following table for the lists of all interrupt and DMA signals and associated interrupt and DMA sources of the UART. See register description section for explanation of interrupt/DMA enable and status.

Table 53-4. Interrupts and DMA

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
interrupt_uart	RRDYEN	UCR1 (bit 9)	RRDY	USR1 (bit 9)
	IDEN	UCR1 (bit 12)	IDLE	USR2 (bit 12)
	DREN	UCR4 (bit 0)	RDR	USR2 (bit 0)
	RXDSEN	UCR3 (bit 6)	RXDS	USR1 (bit 6)
	ATEN	UCR2 (bit 3)	AGTIM	USR1 (bit 8)
interrupt_uart	TXMPTYEN	UCR1 (bit 6)	TXFE	USR2 (bit 14)
	TRDYEN	UCR1 (bit 13)	TRDY	USR1 (bit 13)
	TCEN	UCR4 (bit 3)	TXDC	USR2 (bit 3)
interrupt_uart	OREN	UCR4 (bit 1)	ORE	USR2 (bit 1)
	BKEN	UCR4 (bit 2)	BRCD	USR2 (bit 2)
	WKEN	UCR4 (bit 7)	WAKE	USR2 (bit 7)
	ADEN	UCR1 (bit 15)	ADET	USR2 (bit 15)
	ACIEN	UCR3 (bit 0)	ACST	USR2 (bit 11)
	ESCI	UCR2 (bit 15)	ESCF	USR1 (bit 11)
	ENIRI	UCR4 (bit 8)	IRINT	USR2 (bit 8)
	AIRINTEN	UCR3 (bit 5)	AIRINT	USR1 (bit 5)
	AWAKEN	UCR3 (bit 4)	AWAKE	USR1 (bit 4)
	FRAERREN	UCR3 (bit 11)	FRAERR	USR1 (bit 10)
	PARERREN	UCR3 (bit 12)	PARITYERR	USR1 (bit 15)
	RTSDEN	UCR1 (bit 5)	RTSD	USR1 (bit 12)
	RTSEN	UCR2 (bit 4)	RTSF	USR2 (bit 4)
	DTREN (DCE)	UCR3 (bit 13)	DTRF	USR2 (bit 13)
	RI (DTE)	UCR3 (bit 8)	RIDELT	USR2 (bit 10)
	DCD (DTE)	UCR3 (bit 9)	DCDDELT	USR2 (bit 6)
	DTRDEN	UCR3 (bit 3)	DTRD	USR1 (bit 7)
	SADEN	UMCR (bit 3)	SAD	USR1 (bit 3)
dma_req_rx	RXDMAEN	UCR1 (bit 8)	RRDY	USR1 (bit 9)
	ATDMAEN	UCR1 (bit 2)	AGTIM	USR1 (bit 8)
	IDDMAEN	UCR4 (bit 6)	IDLE	USR2 (bit 12)
dma_req_tx	TXDMAEN	UCR1 (bit 3)	TRDY	USR1 (bit 13)

53.4.2 Clocks

This section describes clocks and special clocking requirements of the UART.

53.4.2.1 Clock requirements

UART module receives 2 clocks, *peripheral_clock* and *module_clock*. The *peripheral_clock* is used as write clock of the TxFIFO, read clock of the RxFIFO and synchronization of the modem control input pins. It must always be running when UART is enabled. There is an exception in stop mode (see Clocking in Low-Power Modes).

The *module_clock* is for all the state machines, writing RxFIFO, reading TxFIFO, etc. It must always be running when UART is sending or receiving characters.This clock is used in order to allow frequency scaling on *peripheral_clock* without changing configuration of baud rate (*module_clock* staying at a fixed frequency).

The constraints on *peripheral_clock* and *module_clock* are as follows:

- *peripheral_clock* and *module_clock* can totally be asynchronous. They can also be synchronous.
- Due to the 16x oversampling of the incoming characters, *module_clock* frequency must always be greater or equal to 16x the maximum baud rate. For example, if max baud rate is 4 Mbit/s, *module_clock* must be greater or equal to 4 M x 16 = 64 MHz.

NOTE

The restriction that *peripheral_clock* frequency must be higher or equal to 16x baud rate has been removed. There is no limitation on *peripheral_clock* frequency to baud rate.

53.4.2.2 Maximum Baud Rate

The max baud rate the UART can support is determined by the max frequency of the module_clock.

For example, if the SoC can provide the fastest module_clock 66.5 MHz, the UART can transmit and receive serial data with the maximum baud rate 66.5M/16 = 4.15 Mbit/s.

The UART supports serial IR interface low speed. In the low speed IrDA mode, the max baud rate is 115.2 Kbit/s. To support the 115.2 Kbit/s, module_clock frequency must be higher or equal to 1.8432 MHz.

53.4.2.3 Clocking in Low-Power Modes

The UART supports 2 low-power modes: DOZE and STOP.

In STOP mode (input pin *stop_req* is at '1'), the UART doesn't need any clock. In this mode the UART can wake-up the ARM platform with the asynchronous interrupts (see Low Power Modes).

- If before entering in STOP mode the software has enabled RTSDEN interrupt, when RTS will change state (put at '0' by external device started to send), the asynchronous interrupt will wake-up the system, *peripheral_clock* and *module_clock* will be provided to the UART before first start bit, so that no data will be lost.
- If RTS doesn't change state (already at '0' before entering in STOP mode), then wake-up interrupt (AWAKE) will be sent at the arrival of first Start bit (on falling edge). In this case, the UART must receive the *peripheral_clock* and *module_clock* during the first half of start bit to correctly receive this character (for example, at 115.2 Kbit/s, UART must receive *peripheral_clock* and *module_clock* at maximum 4.3 microseconds after falling edge of Start bit). If the UART receives *peripheral_clock* and *module_clock* too late, first character will be lost, and so should be dropped. Also, if autobaud detection is enabled, the first character won't be correctly received and another autobaud detection will need to be initiated.

In Doze mode, UART behavior is programmable through DOZE bit (UCR1[1]). If DOZE bit is set to '1', then UART is disabled in Doze mode, and in consequence, UART clocks can be switched-off (after being sure UART is not transmitting nor receiving). On the contrary, if DOZE bit is set to '0', UART is enabled and it must receive *peripheral_clock* and *module_clock* .

53.4.3 General UART Definitions

Definitions of terms that occurs the following discussions are given in this section.

- Bit Time-The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).
- Start bit-The bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- Stop bit-1 bit time of logic 1 that indicates the end of a data frame.
- BREAK-A frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- Mark - When no data is being sent, the serial port's transmit pin's voltage is 1 and is said to be in a MARK state.
- Space - The serial port can also be forced to keep the transmit pin at a 0 and is said to be the SPACE or BREAK state.
- Frame-A start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the

format specified and must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.

- Framing Error-An error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending BREAKs. However, when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.
- Parity Error-An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RX_DATA input. Parity error is calculated only after an entire frame is received.
- Idle-One in NRZ encoding format and selectable polarity in IrDA mode.
- Overrun Error-An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RX_DATA input.

53.4.3.1 RTS_B - UART Request To Send

The UART Request To Send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by setting '0' on the RTS_B pin.

Normally, the transmitter waits until this signal is active (low) before transmitting a character, however when the Ignore RTS (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. An interrupt (RTSD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion. When RTS_B is set to '1' during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode.

53.4.3.2 RTS Edge Triggered Interrupt

The input to the RTS_B pin can be programmed to generate an interrupt on a selectable edge.

See the table below for summary of the operation of the RTS edge triggered interrupt (RTSF).

To enable the RTS_B pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit (UCR2[4]) to 1. Writing 1 to the RTS_B edge triggered interrupt flag (RTSF) bit (USR2[4]) clears the interrupt flag. The interrupt can occur on the rising edge, falling edge, or either edge of the RTS_B input. The request to send edge control (RTEC) field (UCR2[10:9]) programs the edge that generates the interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge. This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 53-5. RTS_B Edge Triggered Interrupt Truth Table

RTS_B	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On…	interrupt_uart
X	0	X	X	0	Interrupt disabled	1
1->0	1	0	0	0	Rising edge	1
0->1	1	0	0	1	Rising edge	0
1->0	1	0	1	1	Falling edge	0
0->1	1	0	1	0	Falling edge	1
1->0	1	1	X	1	Either edge	0
0->1	1	1	X	1	Either edge	0

There is another RTS_B interrupt that is not programmable. The status bit RTSD asserts the *interrupt_uart* interrupt when the RTS_B delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

53.4.3.3 CTS_B - Clear To Send

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the CTS_B trigger level is programmed to trigger at 32 characters received and the receiver detects the valid start bit of the 33 character, it de-asserts this pin. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode.

53.4.3.4 Programmable CTS_B Deassertion

The CTS_B output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level (UCR4[15:10]) at any value less than 32 deasserts the CTS_B pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). However, the receiver continues to receive characters until the RxFIFO is full.

53.4.3.5 TX_DATA - UART Transmit

This is the transmitter serial output. When operating in RS-232/RS-485 mode, NRZ encoded data is transmitted, and the data can be inverted (controlled by INVT (UCR3[1])) before transmitted. When operating in infrared mode, a 3/16 bit-period pulse is output for each 0 bit transmitted, and no pulse is output for each 1 bit transmitted.

For RS-232/RS-485 applications, this pin must be connected to an RS-232/RS-485 transmitter. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode. See Figure 53-3.

53.4.3.6 RX_DATA - UART Receive

This is the receiver serial input. When operating in RS-232/RS-485 mode, NRZ encoded data is expected, and the data can be inverted (controlled by INVR (UCR4[9])) before sampled. When operating in infrared mode, a narrow pulse is expected for each 0 bit received and no pulse is expected for each 1 bit received.

External circuitry must convert the IR signal to an electrical signal. RS-232/RS-485 applications require an external RS-232/RS-485 receiver to convert voltage levels. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode. See the figure below.

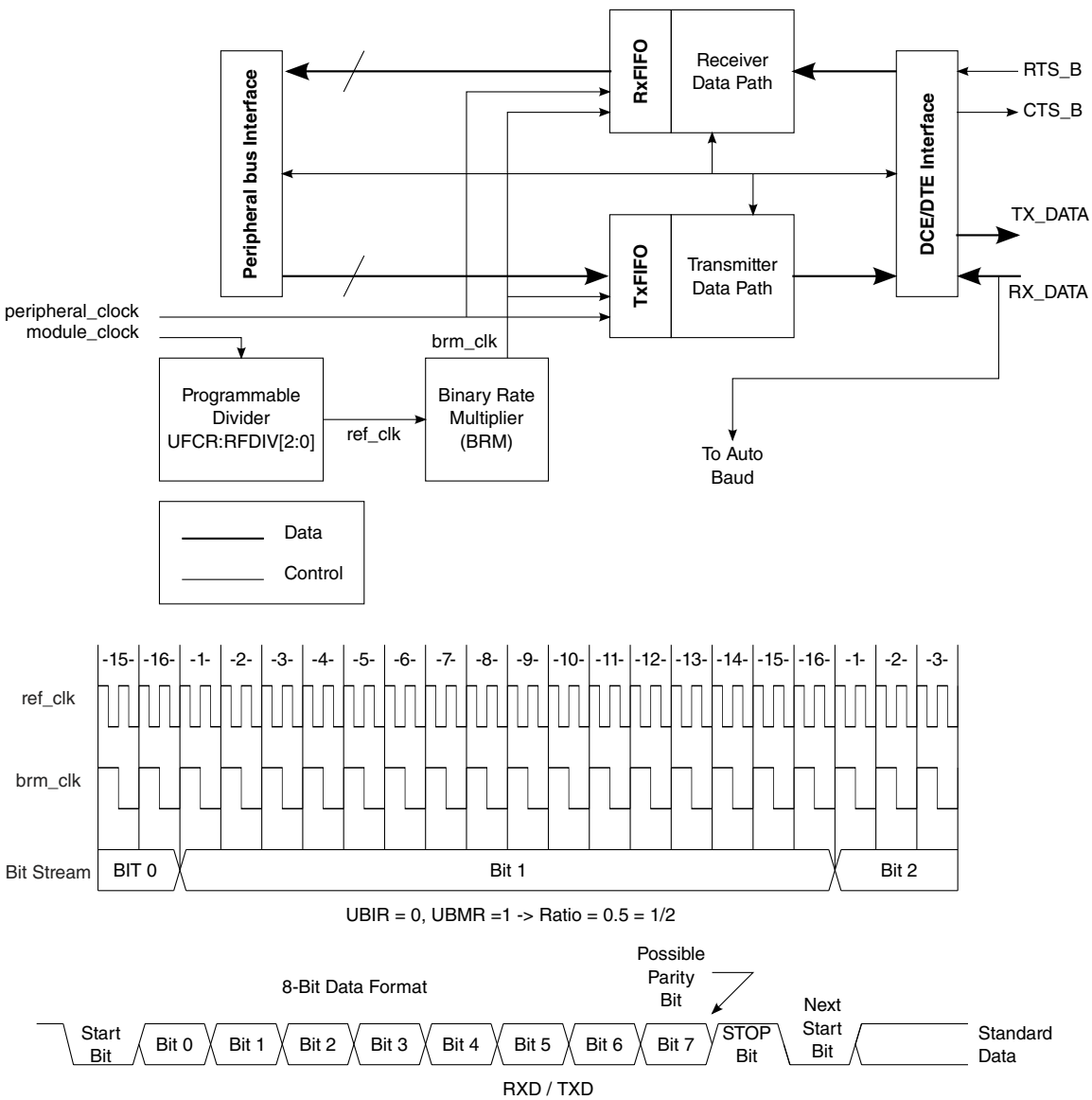


Figure 53-3. UART Simplified Block and Clock Generation Diagrams

53.4.4 Transmitter

The transmitter accepts a parallel character from the ARM platform and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character.

When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. RTS_B can be used to provide flow-control of the serial data. When RTS_B is set to '1', the transmitter finishes sending the character in progress (if any), stops, and waits for RTS_B to be set to '0' again. Generation of BREAK characters and parity errors (for debugging purposes) is supported. The transmitter operates from the clock provided by the Binary Rate Multiplier(BRM). Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the UTXD register with the byte data to the [7:0] bits. The data is written consecutively if the TxFIFO is not full. It is read (internally) consecutively if the TxFIFO is not empty. TXFULL bit (UTS[4]) can be used to control whether TXFIFO is full or not. The TxFIFO can be written regardless of the transmitter is disabled or enabled. If the UART is disabled, user can still write data into the TxFIFO correctly. But in this case the write access will yield to a transfer error.

53.4.4.1 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the TXFE interrupt between writes to the TxFIFO.

When TxFIFO is empty, the software can either send one or several characters. If the software sends one character, it would write the character into the UTXD register, then that character is immediately transferred to the transmitter shift register, assuming the transmitter is already enabled. Without interrupt suppression logic, the TXFE interrupt flag would be set immediately. But, with this logic, the interrupt flag is set when the last bit of the character has been transmitted, for example, before the transmission of the parity bit (if exists) and the stop bit(s).

So, the suppression logic doesn't immediately send the TXFE interrupt flag. It allows the software to write another character to the TxFIFO before the interrupt flag is asserted.

When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt flag is asserted. Writing data to the TxFIFO would release the interrupt flag. The interrupt flag is asserted on the following conditions:

- System Reset

- UART software reset
- When a single character has been written to Transmitter FIFO and then the Transmitter FIFO and the Transmitter Shift Register become empty until another character is written to the Transmitter FIFO
- The last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See the figure below.

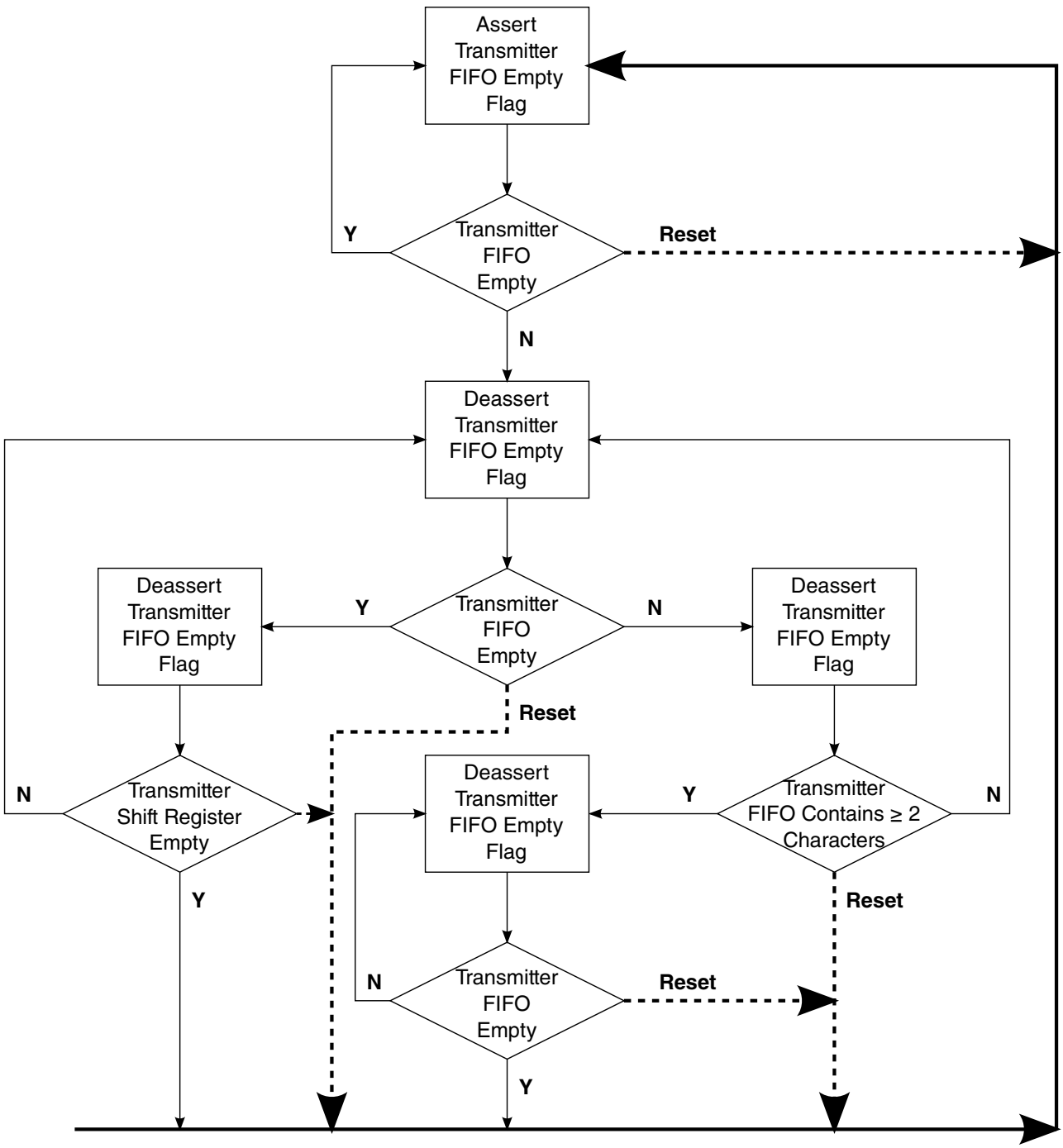


Figure 53-4. Transmitter FIFO Empty Interrupt Suppression Flow Chart

53.4.4.2 Transmitting a Break Condition

Asserting SNDBRK bit of the UCR1 Register forces the transmitter to send a break character (continuous zeros). The transmitter will finish sending the character in progress (if any) before sending break until this bit is reset.

The user is responsible to ensure that this bit is high for long enough to generate a valid BREAK. The transmitter samples SNDBRK after every bit is transmitted. Following completion of the BREAK transmission, the UART will transmit two mark bits. The user can continue to fill the FIFO and any character remaining will be transmitted when the break is terminated.

53.4.5 Receiver

See the figure below for the receiver flow chart.

The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center.

Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the URXD register when parity is enabled. Frame errors and BREAKs are also checked and reported. When a new character is ready to be read by the ARM platform from the RxFIFO, the receive data ready (RDR = USR2[0]) bit is asserted and an interrupt is posted (if DREN = UCR4[0] = 1). If the receiver trigger level is set to 2 (RXTL[5:0] = UFCR[5:0] = 2), and 2 chars have been received into RxFIFO, the receiver ready interrupt flag (RRDY = USR1[9]) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = UCR1[9] = 1). If the UART Receiver Register (URXD) is read once, and in consequence there is only 1 character in the RxFIFO, the interrupt generated by the RDR bit is automatically cleared. The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ encoded data is expected when the IR interface is disabled. The RxFIFO contains 32 half-word entries. Characters received are written consecutively into this FIFO. If the FIFO is full and a 33rd characters is received, this character will be ignored and the USR2[ORE] bit will be set.

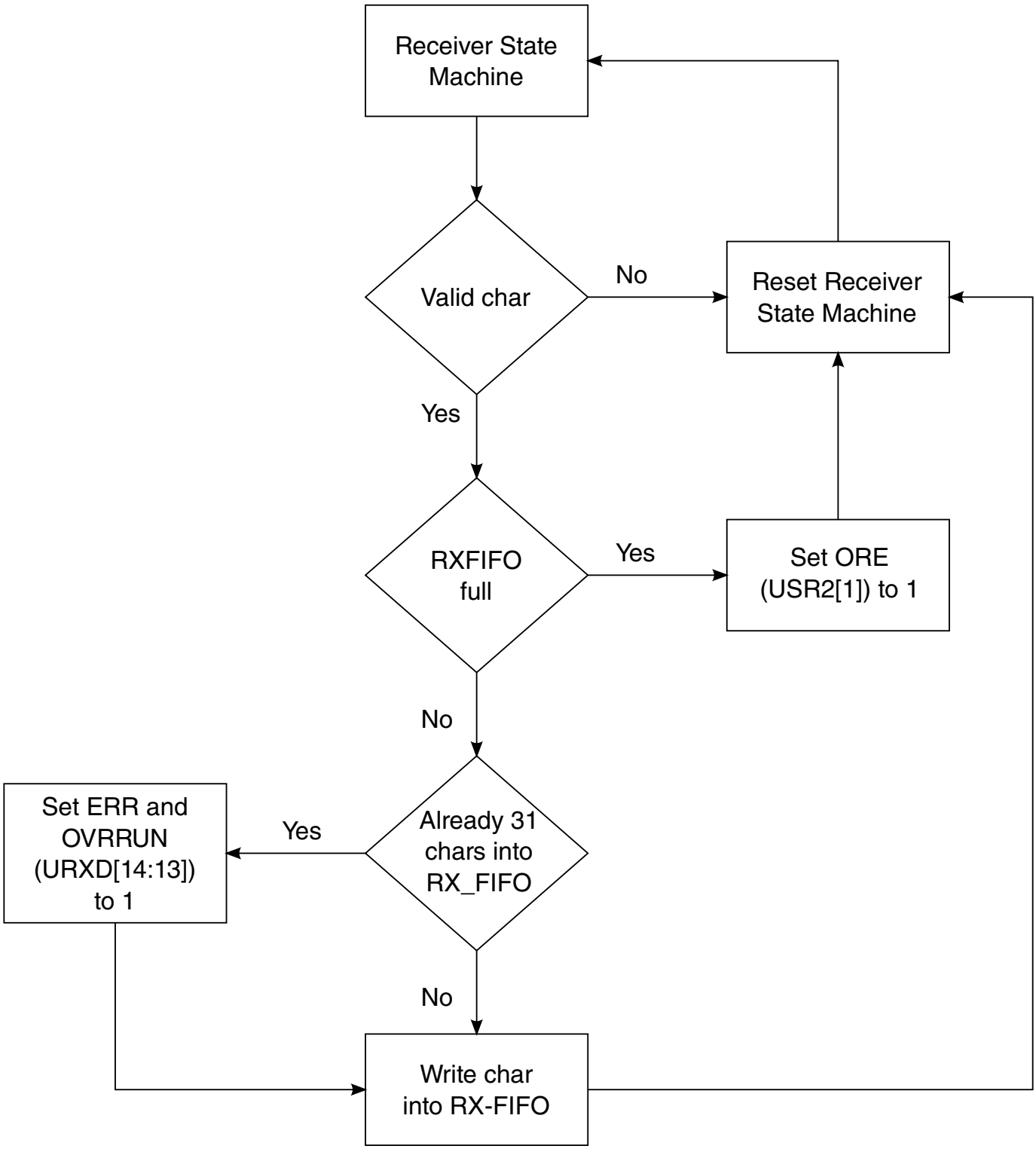


Figure 53-5. Receiver Flow Chart

53.4.5.1 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message.

For an idle condition to occur:

- RxFIFO must be empty and
- RX_DATA pin must be idle for more than a configured number of frames (ICD[1:0] = UCR1[11:10]).

When the idle condition detected interrupt enable (IDEN = UCR1[12]) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt (see the table below). When an idle condition is detected, the IDLE (USR2[12]) bit is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

Table 53-6. Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	*interrupt_uart*
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames
NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal. This table shows how this interrupt affects the *interrupt_uart* signal.				

During a normal message there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

53.4.5.2 Aging Character Detect

The receiver block also includes the possibility to detect when at least one character has been sitting into the RxFIFO for a time corresponding to 8 characters. This aging character capability allows the UART to inform the ARM platform that there is less character into the RxFIFO than the Rx trigger and, no new character has been detected on the RXD line.

The aging capability is a timer which starts to count as soon as the RxFIFO is not empty and its trigger level is not reached (RRDY=0). This counter is reset when either a RxFIFO read is performed or another character starts to present on the RXD line. If none of those two events occurs, the bit AGTIM (USR1[8]) is set when the counter has

measured a time corresponding to 8 characters. AGTIM is cleared by writing a 1 to it. AGTIM can flag an interrupt to ARM platform on *interrupt_uart* if ATEN (UCR2[3]) has been set.

To summarize, AGTIM is set when:

- There is at least one character into RxFIFO.
- No read has occurred on RxFIFO and RXD line has stayed high, for a time corresponding to 8 characters.
- The RxFIFO trigger is not reached (RRDY=0)

53.4.5.3 Receiver Wake

The WAKE bit (USR2[7]) is set when the receiver detects a qualified Start bit. For this, two conditions must be fulfilled, firstly a falling edge on RX_DATA line must be detected and secondly the RX_DATA line must stays at low level for more than a half-bit duration.

When the wake interrupt enable WKEN (UCR4[7]) bit is enabled, the receiver flags an interrupt (*interrupt_uart*) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect. The WAKE status bit can be asserted in either serial RS-232 mode or IR mode. The generation of the WAKE interrupt needs the clock *module_clock* .

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = UCR3[4] = 1), and the ARM platform is in STOP mode, and UART clocks have been shut-off, then a falling edge detected on the receive pin (RX_DATA) asserts the AWAKE bit (USR1[4]) and the *interrupt_uart* interrupt to wake the ARM platform from STOP mode. Re-enable UART clocks and clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect. When IR interface is enabled (UCR1[7]=1), the AWAKE bit is always not asserted. The generation of the asynchronous AWAKE interrupt does not need any clocks.

In IR mode, if the asynchronous IR WAKE interrupt is enabled (AIRINTEN = UCR3[5] = 1), and if the ARM platform is in STOP mode (UART clocks are off when ARM platform in STOP mode), then the detection of a falling edge on the receive pin (RXD_IR), asserts the AIRINT bit (USR1[5]), and the *interrupt_uart* interrupt. This interrupt wakes the ARM platform from STOP mode. Software re-enables UART clocks and clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect. When IR interface is disabled (UCR1[7]=0), the AIRINT bit is always not asserted. The generation of the asynchronous AIRINT interrupt does not need any clocks.

Recommended procedure for programming the asynchronous interrupts is to first clear them by writing 1 to the appropriate bit in the UART Status Register 1 (USR1). Poll or enable the interrupt for the Receiver IDLE Interrupt Flag (RXDS) in the USR1. When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RX_DATA pin is idle (high). After following this procedure, enable the asynchronous interrupt and enter STOP mode.

53.4.5.4 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit (USR2[2]) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect.

Asserting BRCD would generate an interrupt on *interrupt_uart*. The interrupt generation can be masked using the control bit BKEN (UCR4[2]). Receiving a break condition will also effect the following bits in the receiver register URXD:

URXD(11) = BRK. While high this bit indicates that the current char was detected as a break.

URXD(12) = FRMERR. The frame error bit will always be set when BRK is set.

URXD(10) = PRERR. If odd parity was selected the parity error bit will also be set when BRK is set.

URXD(14) = ERR. The error detect bit indicates that the character present in the rx data field has an error status. This can be asserted by a break.

53.4.5.5 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to a 16x clock (*brm_clk*) and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of the *brm_clk*.

See Figure 53-6. The receiver is provided with the majority vote value, which is 2 out of the 3 samples. For examples of the majority vote results of the vote logic, see the following table.

Table 53-7. Majority Vote Results

Samples	Vote
000	0
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of *brm_clk*, however the receiver uses 16x oversampling to take its value in the middle of the sample character.

The receiver starts to count when the Start bit is set however it does not capture the contents of the RxFIFO at the time the Start bit is set. The start bit is validated when 0s are received for 7 consecutive 1/16 of bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame (see Table 53-7). All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

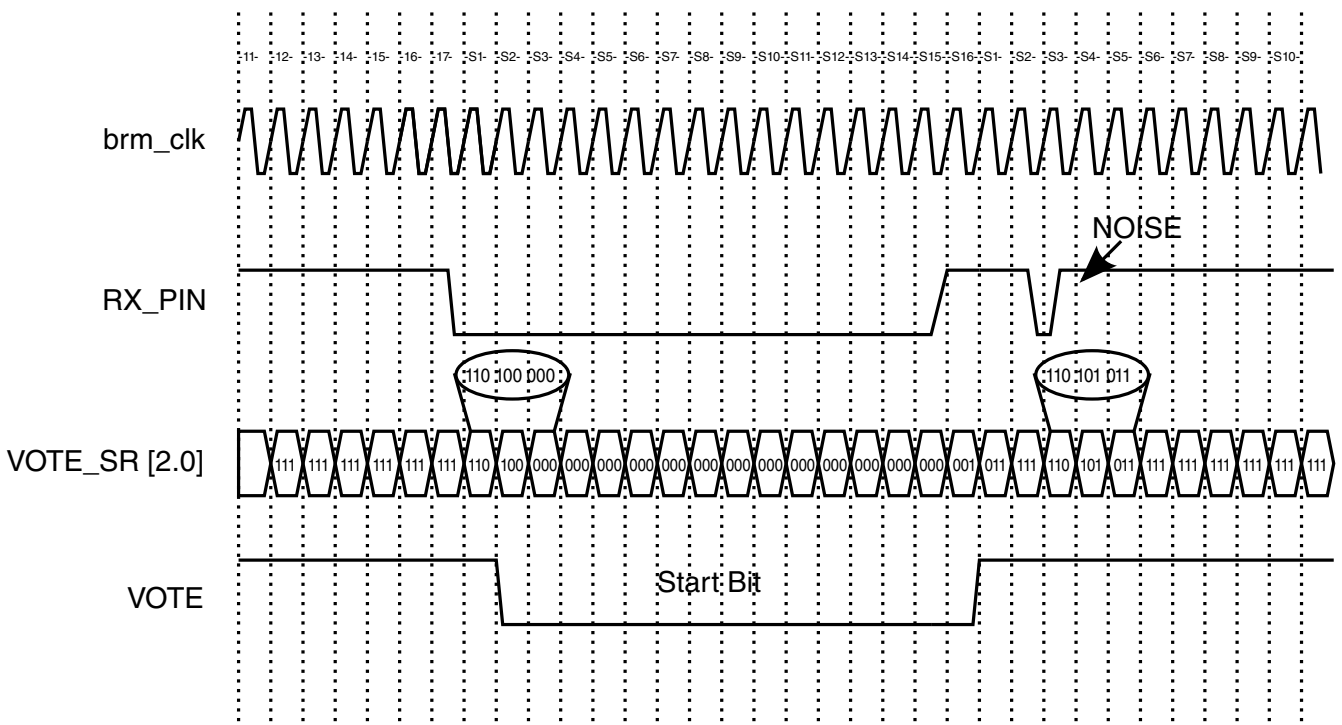


Figure 53-6. Majority Vote Results

A new feature has been recently implemented, it allows to re-synchronize the counter on each edge of RX_DATA line. This is automatic and allows to improve the immunity of UART against signal distortion.

There is a special case when the *brm_clk* frequency is too low and is unable to capture a 0 pulse in IrDA. In this case, the software must set the IRSC (UCR4[5]) bit so that the reference clock (after internal divider) is used for the voting logic. The pulse is validated by counting the length of the pulse.

Refer to Infrared Interface for more details.

53.4.5.6 Baud Rate Automatic Detection Logic

When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = UCR1[14] = 1) and write 1 to the ADET bit (USR2[15]) to clear it.

When ADET=0 and ADBR =1, the detection starts. Then, once the beginning of start bit (transition from 1-to-0 of RX_DATA) has been detected, UART starts a counter (UBRC) working at reference frequency. Once the end of start bit is detected (transition from 0-to-1 of RX_DATA), the value of UBRC - 1 is directly copied into UBMR register. UBIR register is filled with 0x000F.

So, at the end of start bit, registers gets following values:

```
UBRC = number of reference clock periods (after divider) during Start bit.
UBIR = 0x000F
UBMR = UBRC - 1
```

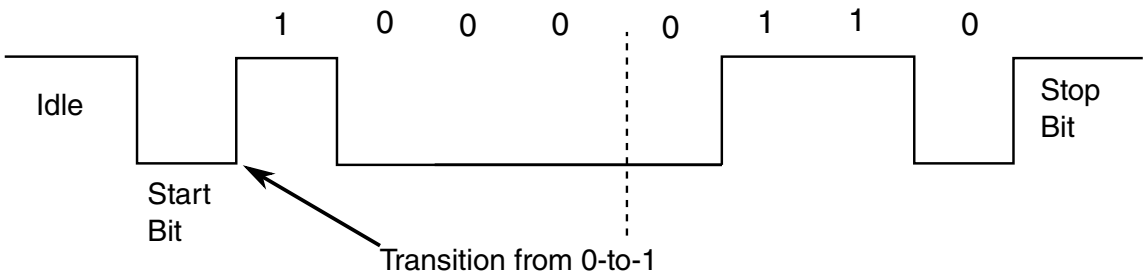
The updated values of the 3 registers can be read.

See Table 53-8 for list of parameters for baud rate detection and Figure 53-7 for baud rate detection protocol diagram.

If any of the UART BRM registers are simultaneously written by the baud rate automatic detection logic and by the peripheral data bus, the peripheral data bus would have lower priority.

Table 53-8. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	*interrupt_uart*
0	X	Manual Configuration	1
1	0	Auto Detection Started	1
1	1	Auto Detection Complete	0
NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal.			



Note: LSB Transmitted first.

Figure 53-7. Baud Rate Detection Protocol Diagram

53.4.5.6.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character "A" or "a" to verify proper detection of the incoming baud rate. When an ASCII character "A" (0x41) or "a" (0x61) is received and no error occurs, the Automatic Detect baud rate bit is set (ADET=1) and if the interrupt is enabled (ADEN=UCR1[15]=1), an interrupt *interrupt_uart* is generated.

When an ASCII character "A" or "a" is not received (because of a bit error or the reception of another character), the auto detection sequence restarts and waits for another 1-to-0 transition.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character "A" or "a" is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baud rate.

The UART interrupt is active (*interrupt_uart* = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character "A" or "a" following the automatic baud rate detection interrupt.

The 16-bit UART Baud Rate Count Register (UBRC) is reset to 4 and stays at 0xFFFF when an overflow occurs. The UBRC register counts (measures) the duration of start bit. When the start bit is detected and counted, the UART Baud Rate Count Register retains its value until the next automatic baud rate detection sequence is initiated.

The Baud Rate Count Register counts only when auto detection is enabled.

53.4.5.6.2 New Baud Rate Determination

In order to fight against the problems caused by the distortion and the noise on the RX_DATA line, the duration of the baud rate measurement has been extended.

Previously, as described above, this determination was based on the measurement of the START bit duration. Now, this measurement is based on the duration of START bit + bit0. Bit0 is the first bit following the START bit. In fact, the counter which is started at the falling edge of START bit is no longer stopped at next rising edge (end of START bit), but it is stopped at the next falling edge (end of bit0). As the character sent is always a "A" (41h) or a "a" (61h), this second falling edge will always be present and it will indicate the end of bit0. Once this counter is stopped, the result is divided by 2 and used by the BRM to determine the incoming baud rate.

NOTE

UBRC register contains the result of this division by two, in consequence it reflects the measurement of the duration of one bit.

53.4.5.6.2.1 New Autobaud Counter Stopped bit and Interrupt

A new bit has been added in USR2 register: ACST (USR2[11]). This bit is set immediately after the determination of the baud rate.

So,

- if ADNIMP is not set (default), ACST is set to 1 after the end of bit0,
- If ADNIMP is set to 1, ACST is set to 1 at the end of START bit.

If ACIEN (UCR3[0]) is set to 1, ACST will flag an interrupt on *interrupt_uart* signal. This interrupt informs the ARM platform that the BRM has just been set with the result of the bit length measurement. If needed, the ARM platform can perform a read of UBMR (or UBRC) register and determine by itself the baud rate measured. Then the ARM platform has the possibility to correct the BRM registers with the nearest standardized baud rate.

NOTE

ACST is set only if ADBR is set to 1, for example, the UART is autobauding.

Clear the ACST bit by writing 1 to it. Writing 0 to the ACST bit has no effect.

53.4.6 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape sequence.

Too much time between two of the "+" characters is interpreted as two "+" characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the UART Escape Character Register (UESC). The software must also enable escape detection feature by setting ESCEN (UCR2[11]) to 1. The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum allowable time between 2 successive escape characters (see the table below). The escape timer is programmable in intervals of 2 ms to a maximum interval of 8.192 seconds.

Table 53-9. Escape Timer Scaling

UTIM Register	Maximum Time Between Specified Escape Characters
0x000	2 ms
0x001	4 ms
0x002	6 ms
0x003	8 ms
0x004	10 ms
...	...
0F8	498 ms
0F9	500 ms
...	...
9C3	5 s
...	...
FFD	8.188 s
FFE	8.190 s
FFF	8.192 s
NOTE: To calculate the time interval: (UTIM_Value + 1) x 0.002 = Time_Interval Example: (09C3 + 1) x 0.002 = 5 s.	

The escape sequence detection feature is available for all the reference frequencies. Before using Escape Sequence Detection, the user must fill the ONEMS register. This 24-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider which is applied on *module_clock* clock.

Example I:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 2 with the internal divider: UFCR[9:7] = 3'b100

$$ONEMS = \frac{66.5 \times 10^6}{2 \times 1000} = 33250 = 81E2h$$

Figure 53-8. Calculation of Frequency for ONEMS Register

Example II:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 1with the internal divider: UFCR[9:7] = 3'b101

$$ONEMS = \frac{66.5 \times 10^6}{1000} = 66500 = 103C4h$$

Figure 53-9. Calculation of Frequency for ONEMS Register

The escape sequence detection interrupt is asserted when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected (ESCF set). Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

53.5 Binary Rate Multiplier (BRM)

The BRM sub-block receives *ref_clk* (*module_clock* clock after divider). From this clock, and with integer and non-integer division, BRM generates a 16x baud rate clock .

The UART transmitte will shift data out based on this 16x baud rate clock. The UART receiver will sample the serial data line based on this 16x baud rate clock. The input and output frequency ratio is programmed in the UART BRM Incremental Register (UBIR) and UART BRM MOD Register (UBMR). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the UBIR = 0x000F and write the divisor to the UBMR register. All values written to these registers must be one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR register must be written before writing to the UBMR register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine what values are to be programmed into UBIR and UBMR for a given reference frequency and desired baud rate. The following equation can be used to help determine these values:

$$\text{BaudRate} = \frac{\text{Ref Freq}}{\left(16 \times \frac{\text{UBMR} + 1}{\text{UBIR} + 1} \right)}$$

Figure 53-10. Frequency and Baud Rate for UBIR and UBMR

With:

Reference Frequency (Hz): UART Reference Frequency (*module_clock* after RFDIV divider)

Baud Rate (bit/s): Desired baud rate.

Integer Division ÷ 21

Reference Frequency = 19.44 MHz

UBIR = 0x000F

UBMR = 0x0014

Baud Rate = 925.7 kbit/s

NOTE

Observe that each value written to the registers is one less than the actual value.

Non-Integer Division

```
Reference Frequency = 16 MHz
Desired Baud Rate = 920 Kbits/s
```

$$\frac{UBMR + 1}{UBIR + 1} = \frac{RefFreq}{16 \times BaudRate} = \frac{16 \times 10^6}{16 \times 920 \times 10^3} = 1.087$$

```
Ratio = 1.087 = 1087 / 1000
UBIR = 999 (decimal)= 0x3E7
UBMR = 1086 (decimal)= 0x43E
Non-Integer Division
Reference Frequency = 25 MHz
Desired Baud Rate = 920 kbit/s
Ratio = 1.69837 = 625 / 368
UBIR = 367 (decimal)= 0x16F
UBMR = 624 (decimal)= 0x270
```

Non-Integer Division

```
Reference Frequency: 30 MHz
Desired Baud Rate = 115.2 kbit/s
Ratio = 16.276043 = 65153 / 4003
UBIR = 4002 (decimal) = 0x0FA2
UBMR = 65152 (decimal) = 0xFE80
```

53.6 Infrared Interface

53.6.1 Generalities-Infrared

The Infrared interface is selected when IREN (UCR1[7]) is set to 1.

The Infrared Interface is compatible with IrDA Serial Infrared Physical Layer Specification. In this specification, a "zero" is represented by a positive pulse, and a "one" is represented by no pulse (line remains low).

In the UART:

In TX: For each "zero" to be transmitted, a narrow positive pulse which is 3/16 of a bit time is generated. For each "one" to be transmitted no pulse is generated (output is low). External circuitry has to be provided to drive an Infrared LED.

In RX: When receiving, a narrow negative pulse is expected for each "zero" transmitted while no pulse is expected for each "one" transmitted (input is high).

NOTE

Rx part of IR block expects to receive an inverted signal compared to IrDA specification. Circuitry external to the IC transforms the Infrared signal to an electrical signal.

The IR interface has an edge triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a "one" to ENIRI bit.

The behavior of Infrared Interface is determined by 3 bits INVT (UCR3[1]), INVR (UCR4[9]) and IRSC (UCR4[5]).

53.6.2 Inverted Transmission and Reception bits (INVT & INVR)

The values of INVT and INVR depend of the IrDA transceiver connected on the TXD_IR and RXD_IR pins of the UART. If this transceiver is not inverting on both paths Tx and Rx, a Zero is represented by a positive pulse and a One is represented by no pulse (line remains low). In this case, the bit INVT must be set to 0 and the bit INVR must be set to 1 (because Rx IR block expects an inverted signal).

On the contrary user must set INVT=1 and INVR=0 if both paths of the transceiver are inverting, that is, a Zero is represented as a negative pulse and a One is represented by no pulse (line remains high). The transceiver can also be inverting on only one path (Tx or Rx), in this case INVT and INVR must be together equal to 1 or to 0, depending on which path is inverted.

53.6.3 InfraRed Special Case (IRSC) Bit

The value to apply to IRSC bit is based on 2 parameters: the baud rate and the Minimum Pulse Duration (MPD) of the transceiver.

According to IrDA Standard Specification, for SIR (Serial IR) baud rates from 2.4 Kbit/s to 115.2 Kbit/s this nominal pulse duration is equal to 3/16 of a bit duration (at the selected baud rate). But, for all the baud rates a Minimum Pulse Duration is also specified. According to IrDA Standard, a Zero is represented by a light pulse, so the IrDA transceiver can't emit a light pulse shorter than the MPD. For SIR, the MPD is constant and equal to 1.41 µs.

But user must take into account the electrical MPD associated with the transceiver on the receiver path. Typically this value is 2.0 µs, but for some manufacturers MPD can go down to 1.0 µs.

In order to understand the meaning of IRSC bit, one must understand how the RX path works in IrDA mode.

When the UART is in IrDA mode, a Zero is not only detected by the state of the RXD_IR line, but also with the duration of the pulse. This pulse duration can be measured with 2 different clocks. In this case, clock is selected with the IRSC bit.

- If IRSC = 0, the clock used is the BRM clock.
- If IRSC = 1, the clock used is the UART internal clock (UART clock after the divider (RFDIV)).

In normal operation, IRSC=0. This means that at any time, the user must ensure that the frequency of BRM_clock is high enough to measure the pulse. The pulse must last at least 2 BRM clock cycles. If this condition is not fulfilled, IRSC must be set to 1.

Let's examine two examples, for a Minimum Pulse Duration equal to the MPD from the IrDA SIR specification (i.e., 1.41 µs).

1: Calculation of BRM Clock Period (Clock Period < 1.41 µs)

The user wants to receive IrDA data at 115.2 Kbit/s. The UBIR and UBMR registers are set in order to create the BRM_clock with a frequency of 16*baud rate = 16 * 115.2K = 1.843 MHz. But at the same time, in order to correctly detect the pulse, the user must be sure that 2* BRM_clock period is lower than 1.41 µs. Lets check:

BRM_clock period = 1/1843000 = 542 ns

So 2*BRM_clock period = 1.09 µs < 1.41 µs. It is fine.

2: Calculation of BRM Clock Period (Clock Period > 1.41 µs)

This time the user wants to receive at 19.2 Kbit/s. So, the BRM_clock is set to 16*19200 = 307.2 kHz. Let's check if 2* BRM_clock period < 1.41 µs:

1. BRM_clock period =1/307200 = 3.25 µs

   So 2*BRM_clock period = 6.50 µs >> 1.41 µs. It doesn't work.

   So, in this case, the BRM clock can't be used to measure the pulse duration and the user must select the UART internal clock by setting IRSC =1.

NOTE

Like for Escape character detection, when IR Special Case is enabled (IRSC=1), the UART must measure a duration. In order to do that, the user must fill the ONEMS register. Refer to Escape Sequence Detection.

53.6.4 IrDA interrupt

Serial infrared mode (SIR) uses an edge triggered interrupt flag IRINT (USR2[8]). When INVR =0, detection of a falling edge on the RXD pin asserts the IRINT bit. When INVR=1, detection of a rising edge on the RXD pin asserts the IRINT bit. When IRINT and ENIRI bits are both asserted, the *interrupt_uart* interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

53.6.5 Conclusion about IrDA

Before using the UART in IrDA, the baud rate limit must be calculated. This baud rate limit will inform the user if IRSC bit has to be set or not.

Let's determine this limit:

As already described, if IRSC = 0, the following condition must always be fulfilled

$$2 \times BRMClockPeriod < MinPulseDuration$$

Figure 53-11. Calculation of Baud Rate

So,

$$BRMClockFrequency > \frac{2}{MPD}$$

So, knowing BRM_clock frequency = 16 * Baud Rate, we get:

$$BaudRate > \frac{1}{8 \times MinPulseDuration}$$

So, the user needs to set IRSC = 0 when:

- If Minimum Pulse Duration = 2.5 us and Baud Rate > 50 Kbit/s.
- If Minimum Pulse Duration = 2.0 us and Baud Rate > 62.5 Kbit/s.
- If Minimum Pulse Duration = 1.41 us and Baud Rate > 88.6 Kbit/s.

NOTE

For baud rates lower than the limit, IRSC must be set to1.

53.6.6 Programming IrDA Interface

53.6.6.1 High Speed

As an example, the following sequence can be used to program the IrDA interface in order to send and receive characters at 115.2 Kbit/s.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 115.2 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR)

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UTS = 0x0000
UFCR = 0x0981
TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
RXTL[5:0] = 0x01: Default value
UBIR = 0x0202
UBMR = 0x20BE Baud rate = 115.2 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000

         UCR4 = 0x8201
CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is ready to send a character as soon as there is a write into UTXD register. And an interrupt will be sent to ARM platform when a character is received.

53.6.6.2 Low Speed

This time, we keep the same assumptions but the speed is now 9.6 Kbit/s. So, this baud rate is below the limit (even with a Min. Pulse Duration of 2.5 us) and thus IRSC must be set to 1.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 9.6 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR).

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UFCR = 0x0981
UFCR[15:10] = TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
UFCR[5:0] = RXTL[5:0] = 0x01: Default value
UBIR = 0x00FF
UBMR = 0xC354 Baud rate = 9.6 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000
UCR3[1] = INVT = 0: Positive pulse represents 0.
UCR4 = 0x8221
UCR4[15:10] = CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[5] = IRSC = 1: Because data rate is below the limit and thus the UART internal clock is
used to measure the pulse duration.
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is now ready to send a character as soon as there is a write into UTXD register. An interrupt will be sent to ARM platform when a character is received.

53.7 9-bit RS-485 Mode

53.7.1 Generalities

The UART provides a 9-bit mode to facilitate multidrop (RS-485) network communication. To enable this mode, set MDEN bit in the UMCR register to 1. When 9-bit RS-485 mode is enabled, UART transmitter can transmit the ninth bit (9^{th} bit) set by TXB8, and UART receiver can differentiate between data frames (9^{th} bit = 0) and address frames (9^{th} bit = 1).

The CTS_B pin can be used to control RS-485 output driver outside the chip.

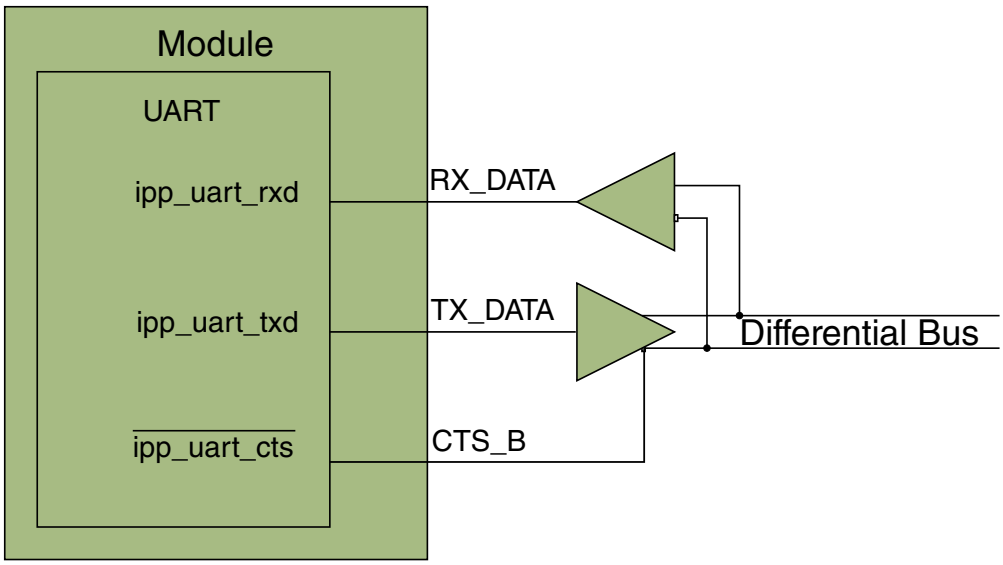


Figure 53-12. RS-485 driver connection (UART in DCE mode)

53.7.2 Transmit 9-bit RS-485 frames

To transmit 9-bit RS-485 frames, user need to enable parity (PREN=1) to enable trasmitting the ninth data bit, set 8-bit data word size (WS=1), and write TXB8 (UMCR[2]) as the 9th bit (bit [8]) to be transmitted (write '0' to TXB8 to transmit a data frame, write '1' to transmit a address frame). The other data bit [7:0] is written to TxFIFO by writing to the UTXD same as normal RS-232 operation.

53.7.3 Receive 9-bit RS-485 frames

To receive 9-bit RS-485 frames, user need to enable parity (PREN=1) to enable receiving the ninth data bit, set 8-bit data word size (WS=1). The receiver will save the 9-bit data to RxFIFO, and user should read the 9th databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

There are two slave address detect modes, normal detect mode and automatic detect mode, and can be selected by SLAM (UMCR[1]).

53.7.3.1 RS-485 Slave Address Normal Detect Mode

To enable Normal Detect mode, clear SLAM (UMCR[1] to 0). The receiver ignores all data frames (9^{th} bit = 0) until an address frame is received (9^{th} bit = 1). At that time, the slave address detected (SAD = USR1[3]) bit is asserted and the *interrupt_uart* interrupt is generated (if SADEN = UMCR[3] = 1). The address byte and sebsequent bytes are all put into RxFIFO along with their 9^{th} bit. The UART will also generate DMA request *dma_req_rx* when the RxFIFO reaches the selected threshold (controlled by RXTL) if receive ready DMA (RXDMAEN = UCR1[8]) request is enabled.

User should read the 9^{th} databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

In this mode, once the UART has detected a 9^{th} bit is equal to '1', it will always save the subsequent frames to RxFIFO. So the software must decide whether the address and data in RxFIFO are needed or not.

53.7.3.2 RS-485 Slave Address Automatic Detect Mode

To enable Automatic Detect Mode, set SLAM (UMCR[1]) to 1. The receiver tries to detect an address byte (frame 9^{th} bit = 1) that matches the programmed SLADDR (UMCR[15:8]) character. If the received byte is a data or an address byte that does not match the programmed SLADDR character, the receiver will discard these data.

Once the UART receives a matching address byte, it will assert the slave address detected (SAD = USR1[3]) bit and the *interrupt_uart* interrupt will be generated (if SADEN = UMCR[3] = 1). The address byte and sebsequent bytes are all put into RxFIFO along with their 9^{th} bit. If receive ready DMA(RXDMAEN = UCR1[8]) request is enabled, the UART will also generate DMA request *dma_req_rx* when the RxFIFO reaches the selected threshold (controlled by RXTL).

If another address byte is received and this address byte does not match SLADDR character, the receiver will discard the address byte and subsequent data byte. If the address byte again matches SLADDR character, the receiver will put this address byte and subsequent data byte in the RxFIFO along with their 9^{th} bit.

User should read the 9^{th} databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

See Initialization for 9-bit RS-485 programming guide.

53.8 Low Power Modes

These modes are controlled by the signals *doze_req* and *stop_req*. The control/status/data registers won't change when getting in/out of low power modes.

Table 53-10. UART Low Power State Operation

	Normal State (*doze_req* = 1'b0 & *stop_req* = 1'b0)	Doze State (*doze_req* = 1'b1)		Stop State (*stop_req* = 1'b1)
		DOZE bit = 0	DOZE bit = 1	
UART-Clock	ON	ON	ON	OFF
UART Serial / IrDA	ON	ON	OFF	OFF

53.8.1 UART Operation in System Doze Mode

While in Doze State (when *doze_req* input pin is set to 1'b1), the UART behavior depends on the DOZE (UCR1[1]) control bit.

While the DOZE bit is negated, the UART serial interface is enabled. While the system is in the Doze State, and the DOZE bit is asserted, the UART is disabled. If the Doze State is entered with the DOZE bit asserted while the UART serial interface was receiving or transmitting data, it will complete the receive/transmit of the current character and signal to the far-end transmitter/receiver to stop sending/receiving.

53.8.2 UART Operation in System Stop Mode

The internal baud rate clocks of the transmitter and receiver are gated off if the *stop_req* signal to UART is asserted. Even though the clocks at the input of the UART continue to run during system Stop mode, the UART will not do any transmission or reception.

The following UART interrupts wake the ARM platform processor from STOP mode:

- RTS (RTSD)
- IrDA Asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)
- RI (RIDELT in DTE mode only)
- DCD (DCDDELT in DTE mode only)
- DTR (DTRD in DCE mode only)
- DSR (DTRD in DTE mode only)

When an asynchronous WAKE (awake) interrupt exits the ARM platform from STOP mode, make sure that a dummy character is sent first because the first character may not be received correctly.

53.8.3 Power Saving Method in UART

The RXEN (UCR2[1]), TXEN (UCR2[2]) and UARTEN (UCR1[0]) bits are set by the user and provide software control of low-power modes.

Setting the UARTEN (UCR1[0]) bit to 0 shuts off the receiver and transmitter logic and the associated clocks.

If the UART is used only in transmit mode, UARTEN and TXEN must be set to 1. If the UART is used only in receive mode, UARTEN and RXEN must be set to 1. Setting TXEN or RXEN to 0 allows to save a lot of power.

53.9 UART Operation in System Debug State

The bit UTS [11] controls whether the UART will respond to the input signal *debug_req*, or whether it will continue to run as normal.

If the UART is programmed to respond to *debug_req*:

1. The UART will halt all operations upon detecting the *debug_req* input.
2. A transfer in progress, either to/from a core (using the IP Bus interface) or to/from an external device, will be completed before halting. This means a single byte/word transfer, not an entire FIFO. Reception of any further data from an external device will be disabled.
3. Internal registers will continue to be writable and readable using the IP Bus interface. A read will leave the contents unaffected.
4. The RX FIFO is affected in debug mode in the following way:
   - All writes into the RX FIFO are prevented.
   - The bit RXDBG (UTS[9]) is used to select the readability of the RX FIFO during debug mode:

     RXDBG = 0: hold the read pointer at the location it had upon entering debug mode, and URXD register returns only the data value at that location, no matter how many reads attempted.

RXDBG = 1, selectable at any time: Allow to read the characters received in Rx FIFO. It will not be possible to re-read previously read locations, nor will it be possible to readjust the read pointer to the value it had prior to entering debug mode.

53.10 Reset

This section describes how to reset the block and explains special requirements related to reset.

53.10.1 Hardware reset

All of registers, FIFOs, state machines and sequential elements can be reset to their initial values by hardware reset or power on reset.

53.10.2 Software reset

The status registers USR1 and USR2, BRM registers UBIR and UBMR, TxFIFO and RxFIFO, and transmitter and receiver state machines can be reset by software reset. Internal logic will keep the software reset asserted for about 4 *module_clock* cycles.

Programmer can follow the following software reset sequence:

1. Clear the SRST_B bit (UCR2[0])
2. Wait for software reset complete: poll SOFTRST bit (UTS[0]) until it is 0.
3. Re-program baud rate registers: Re-write UBIR and UBMR.

53.11 Transfer Error

The UART can generate a transfer error on the peripheral bus in the following cases:

- Core is writing into a read-only register.
- Core is accessing (read or write) an unused location within the assigned address space reserved to UART.

- Core is writing into UTXD register with transmit interface disabled (TXEN=0 or UARTEN=0)
- Core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0)

53.12 Functional Timing

This section includes timing diagrams for functional signaling.

53.12.1 IrDA Mode

According to IrDA specification, the low speed (115.2Kbit/s and below) IR frame format is compatible with UART frame.

In this figure, an example data 0x65 is used.

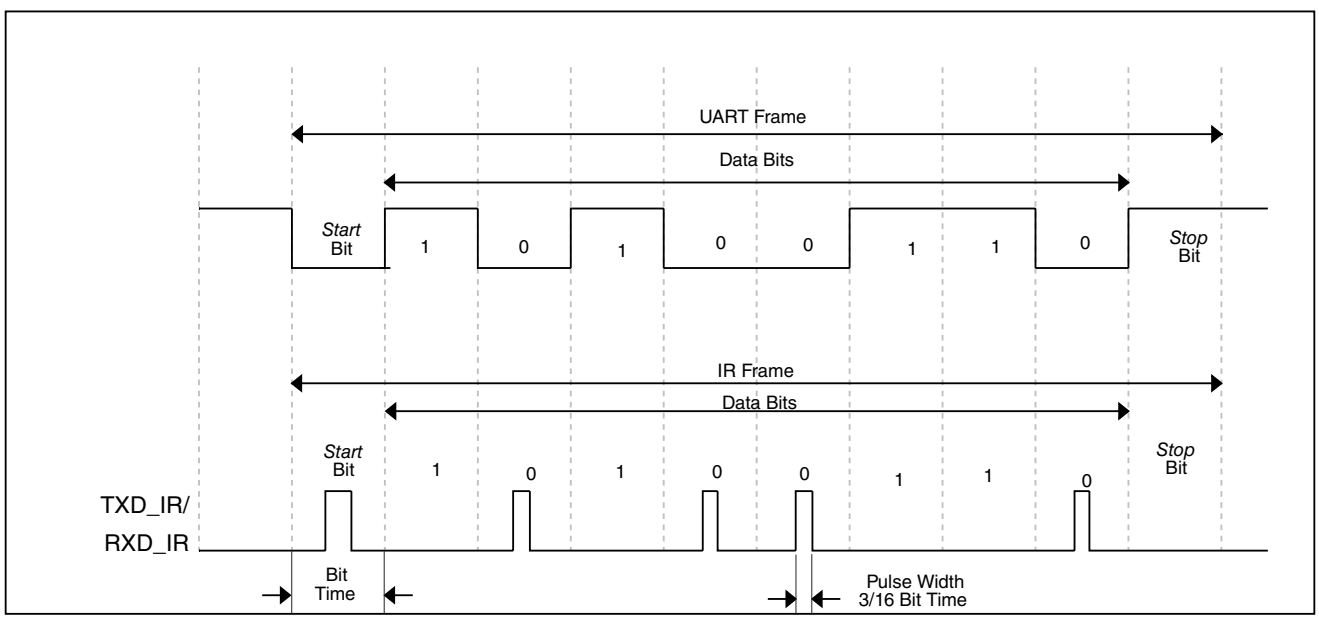


Figure 53-13. Timing diagram of Low Speed IR (<=115.2 Kbit/s) Data Line

53.13 Initialization

53.13.1 Programming the UART in RS-232 mode

As an example, the following sequence can be used to program the UART in order to send and receive characters in RS-232 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 921.6 Kbps
- Data bits = 8 bits
- Parity = Even
- Stop bits = 1 bit
- Flow control = Hardware

Main program:

1. UCR1 = 0x0001

   Enable the UART.

2. UCR2 = 0x2127

   Set hardware flow control, data format and enable transmitter and receiver.

3. UCR3 = 0x0704

   Set UCR3[RXDMUXSEL] = 1.

4. UCR4 = 0x7C00

   Set CTS trigger level to 31,

5. UFCR = 0x089E

   Set internal clock divider = 5 (divide input uart clock by 5). So the reference clock is 100 MHz/5 = 20 MHz.

   Set TXTL = 2 and RXTL = 30.

6. UBIR = 0x08FF
7. UBMR = 0x0C34

   In the above two steps, set baud rate to 921.6Kbps based on the 20MHz reference clock.

8. UCR1 = 0x2201

   Enable the TRDY and RRDY interrupts.

9. UMCR = 0x0000

UMCR stay at default value 0x0000

Interrupt service routine for the transmitter:

- Write characters into UTXD

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2. Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Read characters from URXD

The RRDY interrupt will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

53.13.2 Programming the UART in 9-bit RS-485 mode

As an example, the following sequence can be used to program the UART in order to send and receive frames in RS-485 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 5 Mbps

Main program:

1. UCR1 = 0x0001

   Enable the UART.

2. UCR2 = 0x4127

   Set software flow control ($\overline{\text{CTS}}$ pin is controlled by UCR2[12]), enable parity(enable 9^{th} bit rxd/txd), 8-bit word size , and enable transmitter and receiver.

3. UCR4 = 0x7C00

   Set CTS trigger level to 31,

4. UFCR = 0x0A9E

   Set RFDIV = 5 (divide input uart clock by 1), so the reference clock is 100 MHz. Set UART in DCE mode (RS-485 driver connection outside the chip is the same as Figure 53-12)

Set TXTL = 2 and RXTL = 30.

5. UBIR = 0x0003
6. UBMR = 0x0004

   In the above two steps, set baud rate to 5 Mbps based on the 100 MHz reference clock.

7. UCR1 = 0x2001 when UART as a master ,

   or UCR1 = 0x0201 (or 0x0101) when UART as a slave.

   Enable TRDY interrupt when UART as a master, enable RRDY interrupt or DMA request when UART as a slave.

8. UMCR = 0xA50B

   Enable 9-bit RS-485 mode, enable SAD interrupt, set automatic slave address detect mode, set slave address is 0xA5.

Interrupt service routine for the transmitter:

- Transmit data: write its ninth bit (bit[8]) to UMCR[2], write its bit [7:0] into UTXD[7:0]

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2.

Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Receive data: read its ninth bit (bit[8]) from URXD[10] , read its bit [7:0] from URXD[7:0].

Note: in RS-485 mode, URXD[10] bit is not the parity error, instead it holds the ninth bit (bit[8]) of the received data.

The SAD interrupt can not de-assert automatically, it needs MCU write 1 to USR1[3] to clear it . The RRDY interrupt or DMA request will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

53.14 References

- EIA/TIA-232-F Interface Standard

http://www.eia.org, http://www.tiaonline.org/standards

- IrDA Standard

  http://www.irda.org

53.15 UART Memory Map/Register Definition

UART supports 8-bit, 16-bit and 32-bit accesses to 32-bit memory-mapped addresses. Any access to unmapped memory location will yield a transfer error.

All registers except the ONEMS described in this section are 16-bit registers. The ONEMS register is a 24-bit register.

- For 32-bit write accesses, the upper two bytes will not be taken into account.
- For 32-bit read accesses the upper two bytes will return 0.

The ONEMS register is expanded from 16 bits to 24 bits in order to support the high frequency of the BRM internal clock *ref_clk* (*module_clock* after divider). The ONEMS register can be accessed as 8 bits, 16 bits or 32 bits.

- For 32-bit write accesses, the most significant byte of the ONEMS will be discarded.
- For 32-bit read accesses, the most significant byte of the ONEMS will be read as 0.

UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
201_8000	UART Receiver Register (UART7_URXD)	32	R	0000_0000h	53.15.1/ 3171
201_8040	UART Transmitter Register (UART7_UTXD)	32	W	0000_0000h	53.15.2/ 3173
201_8080	UART Control Register 1 (UART7_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
201_8084	UART Control Register 2 (UART7_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
201_8088	UART Control Register 3 (UART7_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179
201_808C	UART Control Register 4 (UART7_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
201_8090	UART FIFO Control Register (UART7_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
201_8094	UART Status Register 1 (UART7_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
201_8098	UART Status Register 2 (UART7_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
201_809C	UART Escape Character Register (UART7_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190
201_80A0	UART Escape Timer Register (UART7_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
201_80A4	UART BRM Incremental Register (UART7_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191
201_80A8	UART BRM Modulator Register (UART7_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
201_80AC	UART Baud Rate Count Register (UART7_UBRC)	32	R	0000_0004h	53.15.14/ 3192
201_80B0	UART One Millisecond Register (UART7_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193
201_80B4	UART Test Register (UART7_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
201_80B8	UART RS-485 Mode Control Register (UART7_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195
202_0000	UART Receiver Register (UART1_URXD)	32	R	0000_0000h	53.15.1/ 3171
202_0040	UART Transmitter Register (UART1_UTXD)	32	W	0000_0000h	53.15.2/ 3173
202_0080	UART Control Register 1 (UART1_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
202_0084	UART Control Register 2 (UART1_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
202_0088	UART Control Register 3 (UART1_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179
202_008C	UART Control Register 4 (UART1_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
202_0090	UART FIFO Control Register (UART1_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183
202_0094	UART Status Register 1 (UART1_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
202_0098	UART Status Register 2 (UART1_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
202_009C	UART Escape Character Register (UART1_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190
202_00A0	UART Escape Timer Register (UART1_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
202_00A4	UART BRM Incremental Register (UART1_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_00A8	UART BRM Modulator Register (UART1_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
202_00AC	UART Baud Rate Count Register (UART1_UBRC)	32	R	0000_0004h	53.15.14/ 3192
202_00B0	UART One Millisecond Register (UART1_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193
202_00B4	UART Test Register (UART1_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
202_00B8	UART RS-485 Mode Control Register (UART1_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195
202_4000	UART Receiver Register (UART8_URXD)	32	R	0000_0000h	53.15.1/ 3171
202_4040	UART Transmitter Register (UART8_UTXD)	32	W	0000_0000h	53.15.2/ 3173
202_4080	UART Control Register 1 (UART8_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
202_4084	UART Control Register 2 (UART8_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
202_4088	UART Control Register 3 (UART8_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179
202_408C	UART Control Register 4 (UART8_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
202_4090	UART FIFO Control Register (UART8_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183
202_4094	UART Status Register 1 (UART8_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
202_4098	UART Status Register 2 (UART8_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
202_409C	UART Escape Character Register (UART8_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190
202_40A0	UART Escape Timer Register (UART8_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
202_40A4	UART BRM Incremental Register (UART8_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191
202_40A8	UART BRM Modulator Register (UART8_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
202_40AC	UART Baud Rate Count Register (UART8_UBRC)	32	R	0000_0004h	53.15.14/ 3192
202_40B0	UART One Millisecond Register (UART8_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193
202_40B4	UART Test Register (UART8_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
202_40B8	UART RS-485 Mode Control Register (UART8_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_8000	UART Receiver Register (UART2_URXD)	32	R	0000_0000h	53.15.1/ 3171
21E_8040	UART Transmitter Register (UART2_UTXD)	32	W	0000_0000h	53.15.2/ 3173
21E_8080	UART Control Register 1 (UART2_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
21E_8084	UART Control Register 2 (UART2_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
21E_8088	UART Control Register 3 (UART2_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179
21E_808C	UART Control Register 4 (UART2_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
21E_8090	UART FIFO Control Register (UART2_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183
21E_8094	UART Status Register 1 (UART2_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
21E_8098	UART Status Register 2 (UART2_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
21E_809C	UART Escape Character Register (UART2_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190
21E_80A0	UART Escape Timer Register (UART2_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
21E_80A4	UART BRM Incremental Register (UART2_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191
21E_80A8	UART BRM Modulator Register (UART2_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
21E_80AC	UART Baud Rate Count Register (UART2_UBRC)	32	R	0000_0004h	53.15.14/ 3192
21E_80B0	UART One Millisecond Register (UART2_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193
21E_80B4	UART Test Register (UART2_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
21E_80B8	UART RS-485 Mode Control Register (UART2_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195
21E_C000	UART Receiver Register (UART3_URXD)	32	R	0000_0000h	53.15.1/ 3171
21E_C040	UART Transmitter Register (UART3_UTXD)	32	W	0000_0000h	53.15.2/ 3173
21E_C080	UART Control Register 1 (UART3_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
21E_C084	UART Control Register 2 (UART3_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
21E_C088	UART Control Register 3 (UART3_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_C08C	UART Control Register 4 (UART3_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
21E_C090	UART FIFO Control Register (UART3_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183
21E_C094	UART Status Register 1 (UART3_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
21E_C098	UART Status Register 2 (UART3_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
21E_C09C	UART Escape Character Register (UART3_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190
21E_C0A0	UART Escape Timer Register (UART3_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
21E_C0A4	UART BRM Incremental Register (UART3_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191
21E_C0A8	UART BRM Modulator Register (UART3_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
21E_C0AC	UART Baud Rate Count Register (UART3_UBRC)	32	R	0000_0004h	53.15.14/ 3192
21E_C0B0	UART One Millisecond Register (UART3_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193
21E_C0B4	UART Test Register (UART3_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
21E_C0B8	UART RS-485 Mode Control Register (UART3_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195
21F_0000	UART Receiver Register (UART4_URXD)	32	R	0000_0000h	53.15.1/ 3171
21F_0040	UART Transmitter Register (UART4_UTXD)	32	W	0000_0000h	53.15.2/ 3173
21F_0080	UART Control Register 1 (UART4_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
21F_0084	UART Control Register 2 (UART4_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
21F_0088	UART Control Register 3 (UART4_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179
21F_008C	UART Control Register 4 (UART4_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
21F_0090	UART FIFO Control Register (UART4_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183
21F_0094	UART Status Register 1 (UART4_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
21F_0098	UART Status Register 2 (UART4_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
21F_009C	UART Escape Character Register (UART4_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21F_00A0	UART Escape Timer Register (UART4_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
21F_00A4	UART BRM Incremental Register (UART4_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191
21F_00A8	UART BRM Modulator Register (UART4_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
21F_00AC	UART Baud Rate Count Register (UART4_UBRC)	32	R	0000_0004h	53.15.14/ 3192
21F_00B0	UART One Millisecond Register (UART4_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193
21F_00B4	UART Test Register (UART4_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
21F_00B8	UART RS-485 Mode Control Register (UART4_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195
21F_4000	UART Receiver Register (UART5_URXD)	32	R	0000_0000h	53.15.1/ 3171
21F_4040	UART Transmitter Register (UART5_UTXD)	32	W	0000_0000h	53.15.2/ 3173
21F_4080	UART Control Register 1 (UART5_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
21F_4084	UART Control Register 2 (UART5_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
21F_4088	UART Control Register 3 (UART5_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179
21F_408C	UART Control Register 4 (UART5_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
21F_4090	UART FIFO Control Register (UART5_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183
21F_4094	UART Status Register 1 (UART5_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
21F_4098	UART Status Register 2 (UART5_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
21F_409C	UART Escape Character Register (UART5_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190
21F_40A0	UART Escape Timer Register (UART5_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
21F_40A4	UART BRM Incremental Register (UART5_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191
21F_40A8	UART BRM Modulator Register (UART5_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
21F_40AC	UART Baud Rate Count Register (UART5_UBRC)	32	R	0000_0004h	53.15.14/ 3192
21F_40B0	UART One Millisecond Register (UART5_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193

Table continues on the next page...

UART memory map (continued)

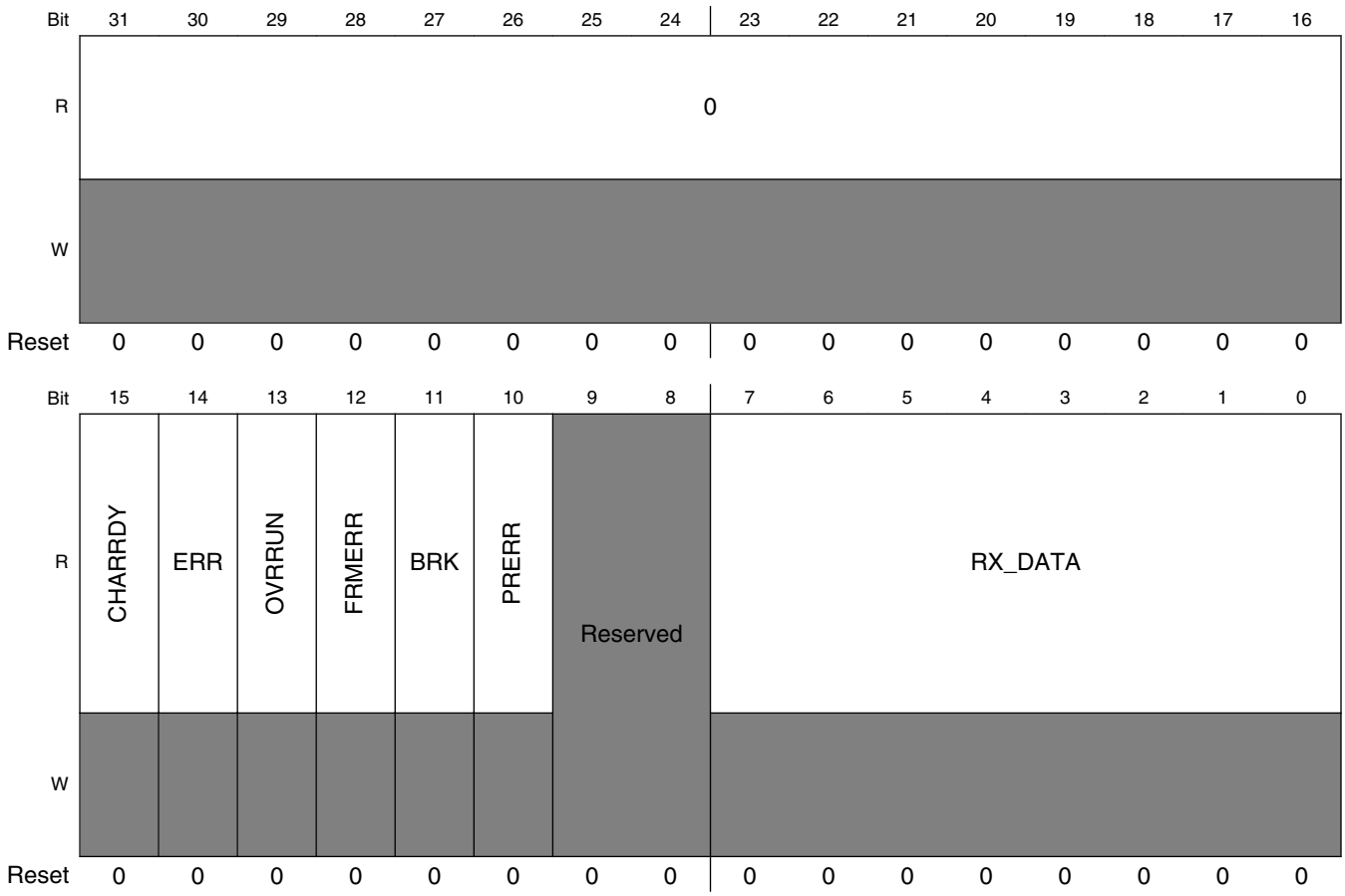
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21F_40B4	UART Test Register (UART5_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
21F_40B8	UART RS-485 Mode Control Register (UART5_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195
21F_C000	UART Receiver Register (UART6_URXD)	32	R	0000_0000h	53.15.1/ 3171
21F_C040	UART Transmitter Register (UART6_UTXD)	32	W	0000_0000h	53.15.2/ 3173
21F_C080	UART Control Register 1 (UART6_UCR1)	32	R/W	0000_0000h	53.15.3/ 3174
21F_C084	UART Control Register 2 (UART6_UCR2)	32	R/W	0000_0001h	53.15.4/ 3176
21F_C088	UART Control Register 3 (UART6_UCR3)	32	R/W	0000_0700h	53.15.5/ 3179
21F_C08C	UART Control Register 4 (UART6_UCR4)	32	R/W	0000_8000h	53.15.6/ 3181
21F_C090	UART FIFO Control Register (UART6_UFCR)	32	R/W	0000_0801h	53.15.7/ 3183
21F_C094	UART Status Register 1 (UART6_USR1)	32	R/W	0000_2040h	53.15.8/ 3185
21F_C098	UART Status Register 2 (UART6_USR2)	32	R/W	0000_4028h	53.15.9/ 3188
21F_C09C	UART Escape Character Register (UART6_UESC)	32	R/W	0000_002Bh	53.15.10/ 3190
21F_C0A0	UART Escape Timer Register (UART6_UTIM)	32	R/W	0000_0000h	53.15.11/ 3190
21F_C0A4	UART BRM Incremental Register (UART6_UBIR)	32	R/W	0000_0000h	53.15.12/ 3191
21F_C0A8	UART BRM Modulator Register (UART6_UBMR)	32	R/W	0000_0000h	53.15.13/ 3191
21F_C0AC	UART Baud Rate Count Register (UART6_UBRC)	32	R	0000_0004h	53.15.14/ 3192
21F_C0B0	UART One Millisecond Register (UART6_ONEMS)	32	R/W	0000_0000h	53.15.15/ 3193
21F_C0B4	UART Test Register (UART6_UTS)	32	R/W	0000_0060h	53.15.16/ 3194
21F_C0B8	UART RS-485 Mode Control Register (UART6_UMCR)	32	R/W	0000_0000h	53.15.17/ 3195

53.15.1 UART Receiver Register (UARTx_URXD)

NOTE

The UART will yield a transfer error on the peripheral bus when core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0).

Address: Base address + 0h offset



UARTx_URXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 CHARRDY	Character Ready. This read-only bit indicates an invalid read when the FIFO becomes empty and software tries to read the same old data. This bit should not be used for polling for data written to the RX FIFO. 0 Character in RX_DATA field and associated flags are invalid. 1 Character in RX_DATA field and associated flags valid and ready for reading.

Table continues on the next page...

UARTx_URXD field descriptions (continued)

Field	Description
14 ERR	**Error Detect.** Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character. 0 No error status was detected 1 An error status was detected
13 OVRRUN	**Receiver Overrun.** This read-only bit, when HIGH, indicates that the corresponding character was stored in the last position (32nd) of the Rx FIFO. Even if a 33rd character has not been detected, this bit will be set to '1' for the 32nd character. 0 No RxFIFO overrun was detected 1 A RxFIFO overrun was detected
12 FRMERR	**Frame Error.** Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO. 0 The current character has no framing error 1 The current character has a framing error
11 BRK	**BREAK Detect.** Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO. 0 The current character is not a BREAK character 1 The current character is a BREAK character
10 PRERR	**In RS-485 mode, it holds the ninth data bit (bit [8]) of received 9-bit RS-485 data** **In RS232/IrDA mode, it is the Parity Error flag**. Indicates whether the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0. 0 = No parity error was detected for data in the RX_DATA field 1 = A parity error was detected for data in the RX_DATA field
9–8 -	This field is reserved. **Reserved**
RX_DATA	**Received Data**. Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.

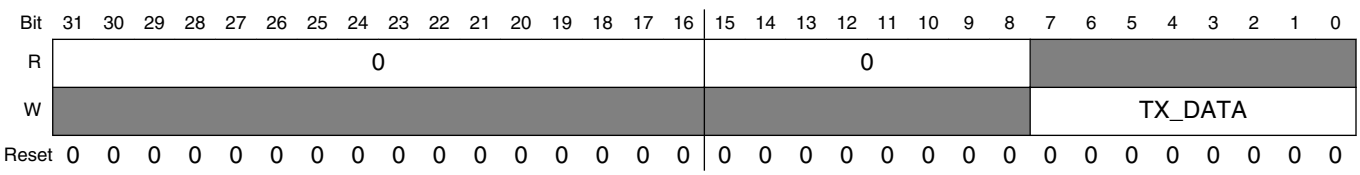
53.15.2 UART Transmitter Register (UARTx_UTXD)

NOTE

The UART will yield a transfer error on the peripheral bus when core is writing into UART_URXD register with transmit interface disabled (TXEN=0 or UARTEN=0).

Memory space between UART_URXD and UART_UTXD registers is reserved. Any read or write access to this space will be considered as an invalid access and yield a transfer error.

Address: Base address + 40h offset



UARTx_UTXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–8 Reserved	This read-only field is reserved and always has the value 0.
TX_DATA	**Transmit Data**. Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

53.15.3 UART Control Register 1 (UARTx_UCR1)

Address: Base address + 80h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADEN	ADBR	TRDYEN	IDEN	ICD		RRDYEN	RXDMAEN	IREN	TXMPTYEN	RTSDEN	SNDBRK	TXDMAEN	ATDMAEN	DOZE	UARTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UCR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ADEN	**Automatic Baud Rate Detection Interrupt Enable**. Enables/Disables the automatic baud rate detect complete (ADET) bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the automatic baud rate detection interrupt 1 Enable the automatic baud rate detection interrupt
14 ADBR	**Automatic Detection of Baud Rate**. Enables/Disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character "A" or "a" (0x41 or 0x61). 0 Disable automatic detection of baud rate 1 Enable automatic detection of baud rate
13 TRDYEN	**Transmitter Ready Interrupt Enable**. Enables/Disables the transmitter Ready Interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TXFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled. **NOTE:** An interrupt will be issued as long as TRDYEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TRDY interrupt. 0 Disable the transmitter ready interrupt 1 Enable the transmitter ready interrupt
12 IDEN	**Idle Condition Detected Interrupt Enable**. Enables/Disables the IDLE bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the IDLE interrupt 1 Enable the IDLE interrupt

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
11–10 ICD	**Idle Condition Detect**. Controls the number of frames RXD is allowed to be idle before an idle condition is reported. 00 Idle for more than 4 frames 01 Idle for more than 8 frames 10 Idle for more than 16 frames 11 Idle for more than 32 frames
9 RRDYEN	**Receiver Ready Interrupt Enable**. Enables/Disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RxFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled. 0 Disables the RRDY interrupt 1 Enables the RRDY interrupt
8 RXDMAEN	**Receive Ready DMA Enable**. Enables/Disables the receive DMA request *dma_req_rx* when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXTL bits. When negated, the receive DMA request is disabled. 0 Disable DMA request 1 Enable DMA request
7 IREN	**Infrared Interface Enable**. Enables/Disables the IR interface. See the IR interface description in Infrared Interface, for more information. Note: MDEN(UMCR[0]) must be cleared to 0 when using IrDA interface. See Table 53-1 0 Disable the IR interface 1 Enable the IR interface
6 TXMPTYEN	**Transmitter Empty Interrupt Enable**. Enables/Disables the transmitter FIFO empty (TXFE) interrupt. *interrupt_uart*. When negated, the TXFE interrupt is disabled. **NOTE:** An interrupt will be issued as long as TXMPTYEN and TXFE are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXFE interrupt. 0 Disable the transmitter FIFO empty interrupt 1 Enable the transmitter FIFO empty interrupt
5 RTSDEN	**RTS Delta Interrupt Enable**. Enables/Disables the RTSD interrupt. The current status of the RTS_B pin is read in the RTSS bit. 0 Disable RTSD interrupt 1 Enable RTSD interrupt
4 SNDBRK	**Send BREAK**. Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO and any characters remaining are transmitted when the BREAK is terminated. 0 Do not send a BREAK character 1 Send a BREAK character (continuous 0s)
3 TXDMAEN	**Transmitter Ready DMA Enable**. Enables/Disables the transmit DMA request *dma_req_tx* when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the *dma_req_tx* is controlled by the TXTL bits.

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
	NOTE: A DMA request will be issued as long as TXDMAEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the transmit DMA request. 0 Disable transmit DMA request 1 Enable transmit DMA request
2 ATDMAEN	**Aging DMA Timer Enable**. Enables/Disables the receive DMA request *dma_req_rx* for the aging timer interrupt (triggered with AGTIM flag in USR1[8]). 0 Disable AGTIM DMA request 1 Enable AGTIM DMA request
1 DOZE	**DOZE**. Determines the UART enable condition in the DOZE state. When *doze_req* input pin is at '1', (the ARM Platform executes a doze instruction and the system is placed in the Doze State), the DOZE bit affects operation of the UART. While in the Doze State, if this bit is asserted, the UART is disabled. See the description in Low Power Modes. 0 The UART is enabled when in DOZE state 1 The UART is disabled when in DOZE state
0 UARTEN	**UART Enable**. Enables/Disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1. UARTEN must be set to 1 before any access to UTXD and URXD registers, otherwise a transfer error is returned. This bit can be set to 1 along with other bits in this register. There is no restriction to the sequence of programing this bit and other control registers. 0 Disable the UART 1 Enable the UART

53.15.4 UART Control Register 2 (UARTx_UCR2)

Address: Base address + 84h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

UARTx_UCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ESCI	**Escape Sequence Interrupt Enable**. Enables/Disables the ESCF bit to generate an interrupt. 0 Disable the escape sequence interrupt 1 Enable the escape sequence interrupt
14 IRTS	**Ignore RTS Pin**. Forces the RTS input signal presented to the transmitter to always be asserted (set to low), effectively ignoring the external pin. When in this mode, the RTS pin serves as a general purpose input. 0 Transmit only when the RTS pin is asserted 1 Ignore the RTS pin
13 CTSC	CTS **Pin Control**. Controls the operation of the CTS_B module output. When CTSC is asserted, the CTS_B module output is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the CTS_B module output is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the CTS_B module output is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the CTS_B pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset the CTS_B signal is negated. 0 The CTS_B pin is controlled by the CTS bit 1 The CTS_B pin is controlled by the receiver
12 CTS	**Clear to Send**. Controls the CTS_B pin when the CTSC bit is negated. CTS has no function when CTSC is asserted. 0 The CTS_B pin is high (inactive) 1 The CTS_B pin is low (active)
11 ESCEN	**Escape Enable**. Enables/Disables the escape sequence detection logic. 0 Disable escape sequence detection 1 Enable escape sequence detection
10–9 RTEC	**Request to Send Edge Control**. Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1 (see Table 53-5). 00 Trigger interrupt on a rising edge 01 Trigger interrupt on a falling edge 1X Trigger interrupt on any edge
8 PREN	**Parity Enable**. Enables/Disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated. 0 Disable parity generator and checker 1 Enable parity generator and checker
7 PROE	**Parity Odd/Even**. Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low. 0 Even parity 1 Odd parity

Table continues on the next page...

UARTx_UCR2 field descriptions (continued)

Field	Description
6 STPB	**Stop**. Controls the number of stop bits after a character. When STPB is low, 1 stop bit is sent. When STPB is high, 2 stop bits are sent. STPB also affects the receiver. 0 The transmitter sends 1 stop bit. The receiver expects 1 or more stop bits. 1 The transmitter sends 2 stop bits. The receiver expects 2 or more stop bits.
5 WS	**Word Size**. Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, however not when a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable. 0 7-bit transmit and receive character length (not including START, STOP or PARITY bits) 1 8-bit transmit and receive character length (not including START, STOP or PARITY bits)
4 RTSEN	**Request to Send Interrupt Enable.** Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the RTS_B pin (the RTSF bit is asserted), an interrupt will be generated on the *interrupt_uart* pin. (See Table 53-5.) 0 Disable request to send interrupt 1 Enable request to send interrupt
3 ATEN	**Aging Timer Enable**. This bit is used to enable the aging timer interrupt (triggered with AGTIM) 0 AGTIM interrupt disabled 1 AGTIM interrupt enabled
2 TXEN	**Transmitter Enable**. Enables/Disables the transmitter. When TXEN is negated the transmitter is disabled and idle. When the UARTEN and TXEN bits are set the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking 1s. The transmitter FIFO cannot be written when this bit is cleared. 0 Disable the transmitter 1 Enable the transmitter
1 RXEN	**Receiver Enable**. Enables/Disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters, because it requires a valid 1-to-0 transition before it can accept any character. 0 Disable the receiver 1 Enable the receiver
0 SRST	**Software Reset**. Once the software writes 0 to SRST_B, the software reset remains active for 4 *module_clock* cycles before the hardware deasserts SRST_B. The software can only write 0 to SRST_B. Writing 1 to SRST_B is ignored. 0 Reset the transmit and receive state machines, all FIFOs and register USR1, USR2, UBIR, UBMR, UBRC , URXD, UTXD and UTS[6-3]. 1 No reset

53.15.5 UART Control Register 3 (UARTx_UCR3)

Address: Base address + 88h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DPEC		DTREN	PARERREN	FRAERREN	DSR	DCD	RI	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	DTRDEN	RXDMUXSEL	INVT	ACIEN
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

UARTx_UCR3 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–14 DPEC	This bit is not used in this chip.
13 DTREN	This bit is not used in this chip.
12 PARERREN	**Parity Error Interrupt Enable. Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt.** 0 Disable the parity error interrupt 1 Enable the parity error interrupt
11 FRAERREN	**Frame Error Interrupt Enable. Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt.** 0 Disable the frame error interrupt 1 Enable the frame error interrupt
10 DSR	This bit is not used in this chip.
9 DCD	This bit is not used in this chip.
8 RI	This bit is not used in this chip.
7 ADNIMP	**Autobaud Detection Not Improved-. Disables new features of autobaud detection (See Baud Rate Automatic Detection Protocol, for more details).** 0 Autobaud detection new features selected 1 Keep old autobaud detection mechanism

Table continues on the next page...

UARTx_UCR3 field descriptions (continued)

Field	Description
6 RXDSEN	Receive Status Interrupt Enable. Controls the receive status interrupt (*interrupt_uart*). When this bit is enabled and RXDS status bit is set, the interrupt *interrupt_uart* will be generated. 0 Disable the RXDS interrupt 1 Enable the RXDS interrupt
5 AIRINTEN	Asynchronous IR WAKE Interrupt Enable. Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the RXD pin. 0 Disable the AIRINT interrupt 1 Enable the AIRINT interrupt
4 AWAKEN	Asynchronous WAKE Interrupt Enable. Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin. 0 Disable the AWAKE interrupt 1 Enable the AWAKE interrupt
3 DTRDEN	This bit is not used in this chip.
2 RXDMUXSEL	RXD Muxed Input Selected. Selects proper input pins for serial and Infrared input signal. **NOTE:** In this chip, UARTs are used in MUXED mode, so that this bit should always be set.
1 INVT	Invert TXD output in RS-232/RS-485 mode, set TXD active level in IrDA mode. In RS232/RS-485 mode(UMCR[0] = 1), if this bit is set to 1, the TXD output is inverted before transmitted. In **IrDA mode**, when INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s. 0 **TXD is not inverted** 1 **TXD is inverted** 0 **TXD** Active low transmission 1 **TXD** Active high transmission
0 ACIEN	**Autobaud Counter Interrupt Enable.** This bit is used to enable the autobaud counter stopped interrupt (triggered with ACST (USR2[11]). 0 ACST interrupt disabled 1 ACST interrupt enabled

53.15.6 UART Control Register 4 (UARTx_UCR4)

Address: Base address + 8Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CTSTL						INVR	ENIRI	WKEN	IDDMAEN	IRSC	LPBYP	TCEN	BKEN	OREN	DREN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UCR4 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–10 CTSTL	**CTS Trigger Level**. Controls the threshold at which the CTS_B pin is deasserted by the RxFIFO. After the trigger level is reached and the CTS_B pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved. 000000 0 characters received 000001 1 characters in the RxFIFO ... — ... — 100000 32 characters in the RxFIFO (maximum)
9 INVR	**Invert RXD input in RS-232/RS-485 Mode, d**etermine **RXD input** logic level being sampled in **In IrDA mode.** **In RS232/RS-485 Mode(UMCR[0] = 1), if this bit is set to 1, the RXD input is inverted before sampled.** **In IrDA mode**,when cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR 1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s. 0 **RXD input is not inverted** 1 **RXD input is inverted** 0 **RXD** active low detection 1 **RXD** active high detection
8 ENIRI	**Serial Infrared Interrupt Enable**. Enables/Disables the serial infrared interrupt. 0 Serial infrared Interrupt disabled 1 Serial infrared Interrupt enabled

Table continues on the next page...

UARTx_UCR4 field descriptions (continued)

Field	Description
7 WKEN	**WAKE Interrupt Enable**. Enables/Disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver. 0 Disable the WAKE interrupt 1 Enable the WAKE interrupt
6 IDDMAEN	**DMA IDLE Condition Detected Interrupt Enable** Enables/Disables the receive DMA request *dma_req_rx* for the IDLE interrupt (triggered with IDLE flag in USR2[12]). 0 DMA IDLE interrupt disabled 1 DMA IDLE interrupt enabled
5 IRSC	**IR Special Case**. Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency. See InfraRed Special Case (IRSC) Bit. 0 The vote logic uses the sampling clock (16x baud rate) for normal operation 1 The vote logic uses the UART reference clock
4 LPBYP	**Low Power Bypass**. Allows to bypass the low power new features in UART. To use during debug phase. 0 Low power features enabled 1 Low power features disabled
3 TCEN	**TransmitComplete Interrupt Enable**. Enables/Disables the TXDC bit to generate an interrupt (*interrupt_uart* = 0) **NOTE:** An interrupt will be issued as long as TCEN and TXDC are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXDC interrupt. 0 Disable TXDC interrupt 1 Enable TXDC interrupt
2 BKEN	**BREAK Condition Detected Interrupt Enable**. Enables/Disables the BRCD bit to generate an interrupt. 0 Disable the BRCD interrupt 1 Enable the BRCD interrupt
1 OREN	**Receiver Overrun Interrupt Enable**. Enables/Disables the ORE bit to generate an interrupt. 0 Disable ORE interrupt 1 Enable ORE interrupt
0 DREN	**Receive Data Ready Interrupt Enable**. Enables/Disables the RDR bit to generate an interrupt. 0 Disable RDR interrupt 1 Enable RDR interrupt

53.15.7 UART FIFO Control Register (UARTx_UFCR)

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

UARTx_UFCR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–10 TXTL	**Transmitter Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved. 000000 Reserved 000001 Reserved 000010 TxFIFO has 2 or fewer characters ... — ... — 011111 TxFIFO has 31 or fewer characters 100000 TxFIFO has 32 characters (maximum)
9–7 RFDIV	Reference Frequency Divider. Controls the divide ratio for the reference clock. The input clock is *module_clock*. The output from the divider is *ref_clk* which is used by BRM to create the 16x baud rate oversampling clock (*brm_clk*). 000 Divide input clock by 6 001 Divide input clock by 5 010 Divide input clock by 4 011 Divide input clock by 3 100 Divide input clock by 2 101 Divide input clock by 1 110 Divide input clock by 7 111 Reserved
6 DCEDTE	**DCE/DTE mode select**. Select UART as data communication equipment (DCE mode) or as data terminal equipment (DTE mode).

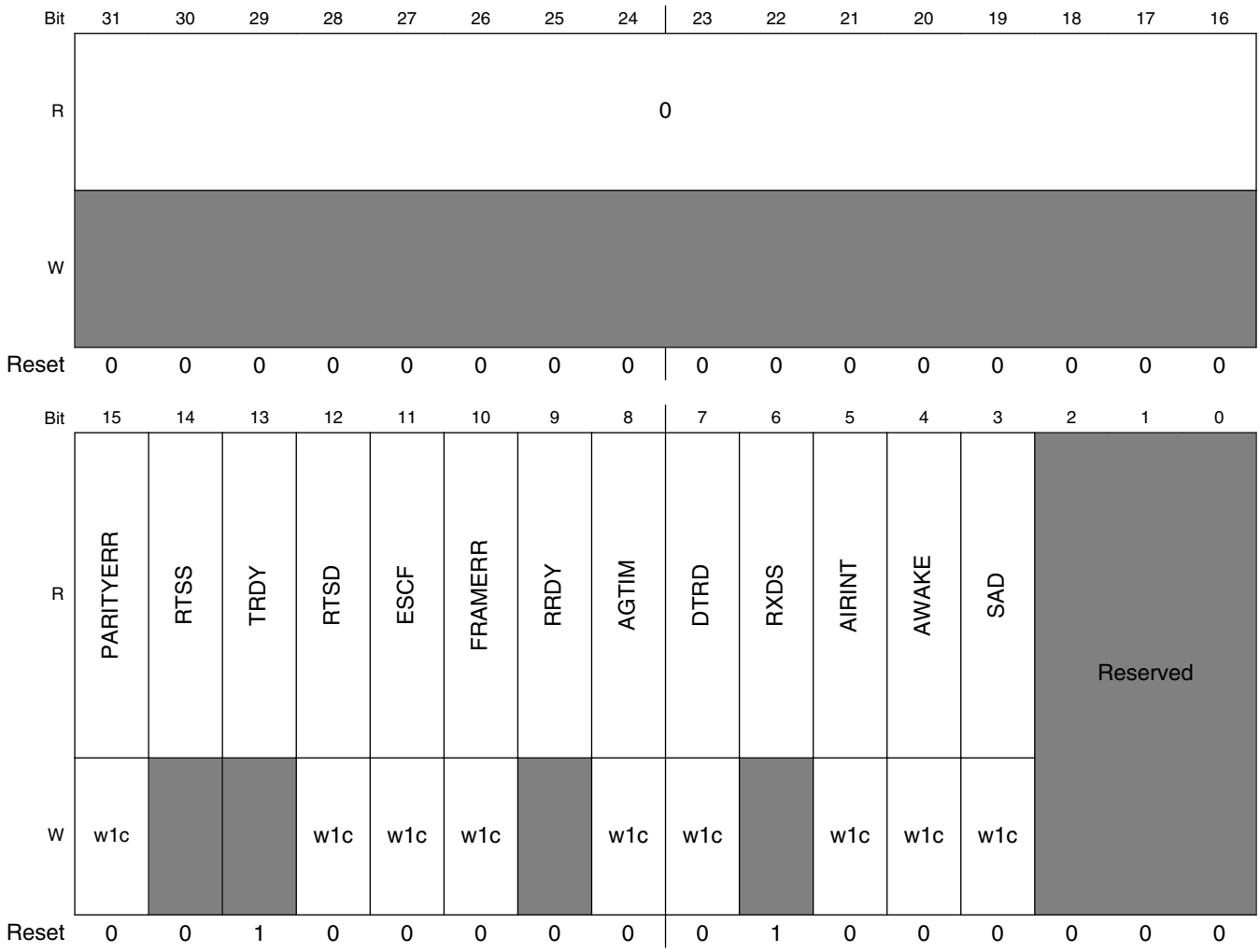
Table continues on the next page...

UARTx_UFCR field descriptions (continued)

Field	Description
	0 DCE mode selected 1 DTE mode selected
RXTL	**Receiver Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the Settings column. Setting 0 to 32 are in use. All other settings are Reserved. 000000 0 characters received 000001 RxFIFO has 1 character ... — ... — 011111 RxFIFO has 31 characters 100000 RxFIFO has 32 characters (maximum)

53.15.8 UART Status Register 1 (UARTx_USR1)

Address: Base address + 94h offset



UARTx_USR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 PARITYERR	**Parity Error Interrupt Flag**. Indicates a parity error is detected. PARITYERR is cleared by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0. 0 No parity error detected 1 Parity error detected (write 1 to clear)
14 RTSS	RTS_B **Pin Status**. Indicates the current status of the RTS_B pin. A "snapshot" of RTS_B is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0.

Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
	0 The RTS_B module input is high (inactive) 1 The RTS_B module input is low (active)
13 TRDY	**Transmitter Ready Interrupt / DMA Flag**. Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO exceeds the threshold set by TXTL bits. At reset, TRDY is set to 1. 0 The transmitter does not require data 1 The transmitter requires data (interrupt posted)
12 RTSD	**RTS Delta.** Indicates whether the RTS_B pin changed state. It (RTSD) generates a maskable interrupt. When in STOP mode, RTS assertion sets RTSD and can be used to wake the processor. The current state of the RTS_B pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0. 0 RTS_B pin did not change state since last cleared 1 RTS_B pin changed state (write 1 to clear)
11 ESCF	**Escape Sequence Interrupt Flag**. Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect. 0 No escape sequence detected 1 Escape sequence detected (write 1 to clear).
10 FRAMERR	**Frame Error Interrupt Flag**. Indicates that a frame error is detected. The *interrupt_uart* interrupt will be generated if a frame error is detected and the interrupt is enabled. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect. 0 No frame error detected 1 Frame error detected (write 1 to clear)
9 RRDY	**Receiver Ready Interrupt / DMA Flag**. Indicates that the RxFIFO data level is above the threshold set by the RXTL bits. (See the RXTL bits description in UART FIFO Control Register (UART_UFCR) for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0. 0 No character ready 1 Character(s) ready (interrupt posted)
8 AGTIM	Ageing Timer Interrupt Flag. Indicates that data in the RxFIFO has been idle for a time of 8 character lengths (where a character length consists of 7 or 8 bits, depending on the setting of the WS bit in UCR2, with the bit time corresponding to the baud rate setting) and FIFO data level is less than RxFIFO threshold level (RXTL in the UFCR). Clear by writing a 1 to it. 0 AGTIM is not active 1 AGTIM is active (write 1 to clear)
7 DTRD	This bit is not used in this chip.
6 RXDS	Receiver IDLE Interrupt Flag. Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled. 0 Receive in progress 1 Receiver is IDLE

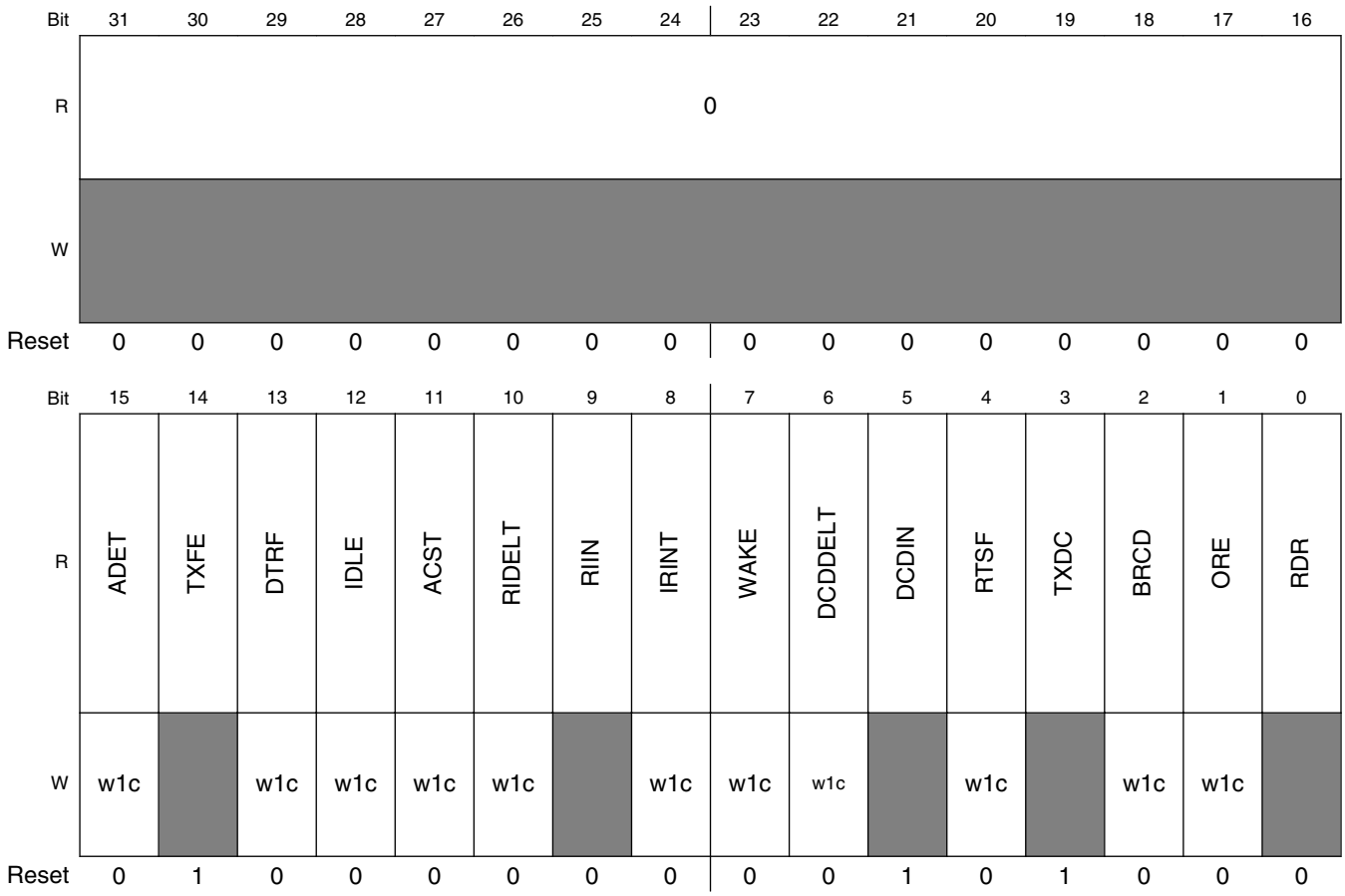
Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
5 AIRINT	Asynchronous IR WAKE Interrupt Flag. Indicates that the IR WAKE pulse was detected on the RXD pin. Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect. 0 No pulse was detected on the RXD IrDA pin 1 A pulse was detected on the RXD IrDA pin
4 AWAKE	Asynchronous WAKE Interrupt Flag. Indicates that a falling edge was detected on the RXD pin. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect. 0 No falling edge was detected on the RXD Serial pin 1 A falling edge was detected on the RXD Serial pin
3 SAD	RS-485 Slave Address Detected Interrupt Flag. Indicates if RS-485 Slave Address was detected . SAD was asserted in RS-485 mode when the SADEN bit is set and Slave Address is detected in RxFIFO (in Nomal Address Detect Mode, the 9th data bit = 1; in Automatic Address Detect Mode, the received charater matches the programmed SLADDR). 0 No slave address detected 1 Slave address detected
-	This field is reserved. Reserved

53.15.9 UART Status Register 2 (UARTx_USR2)

Address: Base address + 98h offset



UARTx_USR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ADET	**Automatic Baud Rate Detect Complete**. Indicates that an "A" or "a" was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect. 0 ASCII "A" or "a" was not received 1 ASCII "A" or "a" was received (write 1 to clear)
14 TXFE	**Transmit Buffer FIFO Empty**. Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress. 0 The transmit buffer (TxFIFO) is not empty 1 The transmit buffer (TxFIFO) is empty
13 DTRF	This bit is not used in this chip.

Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
12 IDLE	**Idle Condition**. Indicates that an idle condition has existed for more than a programmed amount frame (see Idle Line Detect. An interrupt can be generated by this IDLE bit if IDEN (UCR1[12]) is enabled. IDLE is cleared by writing 1 to it. Writing 0 to IDLE has no effect. 0 No idle condition detected 1 Idle condition detected (write 1 to clear)
11 ACST	**Autobaud Counter Stopped**. In autobaud detection (ADBR=1), indicates the counter which determines the baud rate was running and is now stopped. This means either START bit is finished (if ADNIMP=1), or Bit 0 is finished (if ADNIMP=0). See New Autobaud Counter Stopped bit and Interrupt, for more details. An interrupt can be flagged on *interrupt_uart* if ACIEN=1. 0 Measurement of bit length not finished (in autobaud) 1 Measurement of bit length finished (in autobaud). (write 1 to clear)
10 RIDELT	This bit is not used in this chip.
9 RIIN	This bit is not used in this chip.
8 IRINT	**Serial Infrared Interrupt Flag.** When an edge is detected on the RXD pin during SIR Mode, this flag will be asserted. This flag can cause an interrupt which can be masked using the control bit ENIRI: UCR4 [8]. 0 no edge detected 1 valid edge detected (write 1 to clear)
7 WAKE	**Wake**. Indicates the start bit is detected. WAKE can generate an interrupt that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect. 0 start bit not detected 1 start bit detected (write 1 to clear)
6 DCDDELT	This bit is not used in this chip.
5 DCDIN	This bit is not used in this chip.
4 RTSF	**RTS Edge Triggered Interrupt Flag. Indicates if** a programmed edge is detected on the RTS_B pin. The RTEC bits select the edge that generates an interrupt (see Table 53-5). RTSF can generate an interrupt that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect. 0 Programmed edge not detected on RTS_B 1 Programmed edge detected on RTS_B (write 1 to clear)
3 TXDC	**Transmitter Complete**. Indicates that the transmit buffer (TxFIFO) and Shift Register is empty; therefore the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO. 0 Transmit is incomplete 1 Transmit is complete
2 BRCD	**BREAK Condition Detected**. Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect. 0 No BREAK condition was detected 1 A BREAK condition was detected (write 1 to clear)
1 ORE	**Overrun Error**. When set to 1, ORE indicates that the receive buffer (RxFIFO) was full (32 chars inside), and a 33rd character has been fully received. This 33rd character has been discarded. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect.

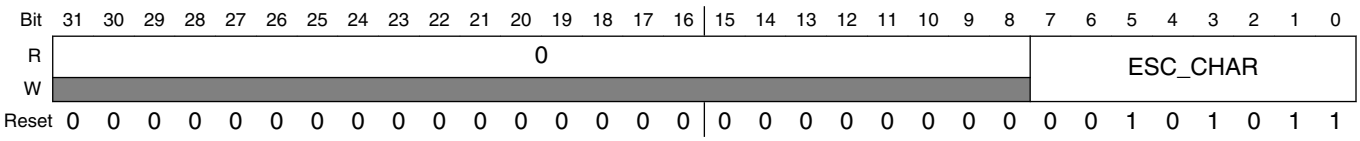
Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
	0 No overrun error 1 Overrun error (write 1 to clear)
0 RDR	**Receive Data Ready**-Indicates that at least 1 character is received and written to the RxFIFO. If the URXD register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared. 0 No receive data ready 1 Receive data ready

53.15.10 UART Escape Character Register (UARTx_UESC)

Address: Base address + 9Ch offset

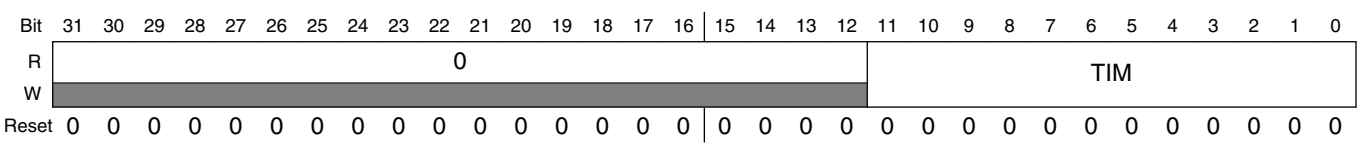


UARTx_UESC field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
ESC_CHAR	**UART Escape Character**. Holds the selected escape character that all received characters are compared against to detect an escape sequence.

53.15.11 UART Escape Timer Register (UARTx_UTIM)

Address: Base address + A0h offset



UARTx_UTIM field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value 0.
TIM	UART Escape Timer. Holds the maximum time interval (in ms) allowed between escape characters. The escape timer register is programmable in intervals of 2 ms. See Escape Sequence Detection and Table 53-9 for more information on the UART escape sequence detection. Reset value 0x000 = 2 ms up to 0xFFF = 8.192 s.

53.15.12 UART BRM Incremental Register (UARTx_UBIR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write 0x000F value into the UBIR after finishing detecting baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[3].

Please note software reset will reset the register to its reset value.

Address: Base address + A4h offset

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

UARTx_UBIR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
INC	Incremental Numerator. Holds the numerator value minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this field using byte accesses is not recommended and is undefined.

53.15.13 UART BRM Modulator Register (UARTx_UBMR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write a proper value into the UBMR based on detected baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[4].

Please note software reset will reset the register to its reset value.

Address: Base address + A8h offset

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

3. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priority than hardware when writing this register at the same time.
4. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priority than hardware when writing this register at the same time.

UARTx_UBMR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
MOD	**Modulator Denominator.** Holds the value of the denominator minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.

53.15.14 UART Baud Rate Count Register (UARTx_UBRC)

Address: Base address + ACh offset

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

UARTx_UBRC field descriptions

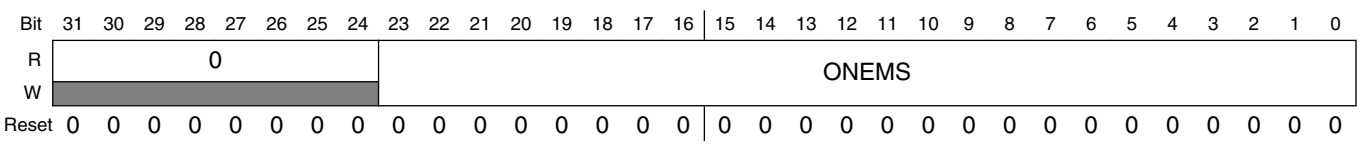
Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
BCNT	**Baud Rate Count Register.** This read only register is used to count the start bit of the incoming baud rate (if ADNIMP=1), or start bit + bit0 (if ADNIMP=0). When the measurement is done, the Baud Rate Count Register contains the number of UART internal clock cycles (clock after divider) present in an incoming bit. BCNT retains its value until the next Automatic Baud Rate Detection sequence has been initiated. The 16 bit Baud Rate Count register is reset to 4 and stays at hex FFFF in the case of an overflow.

53.15.15 UART One Millisecond Register (UARTx_ONEMS)

NOTE

This register has been expanded from 16 bits to 24 bits. In previous versions, the 16-bit ONEMS can only support the maximum 65.535MHz (0xFFFFx1000) *ref_clk*. To support 4Mbps Bluetooth application with 66.5MHz *module_clock*, the value 0x103C4 (66.5M/1000) should be written into this register. In this case, the 16 bits are not enough to contain the 0x103C4. So this register was expanded to 24 bits to support high frequency of the *ref_clk*.

Address: Base address + B0h offset

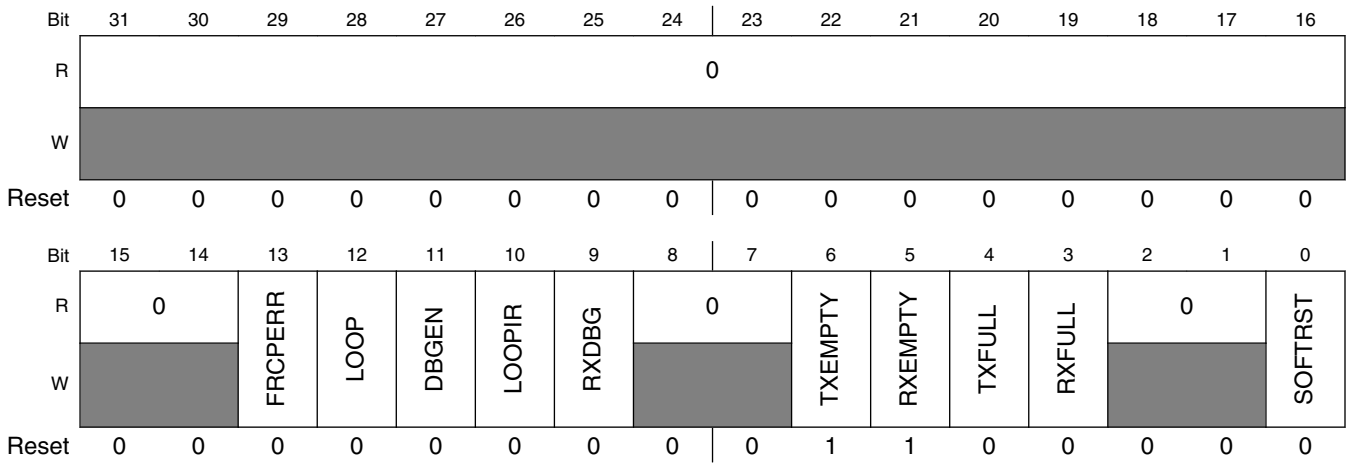


UARTx_ONEMS field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.
ONEMS	**One Millisecond Register.** This 24-bit register must contain the value of the UART internal frequency (*ref_clk* in Figure 53-1) divided by 1000. The internal frequency is obtained after the UART BRM internal divider (F (*ref_clk*) = F(*module_clock*) / RFDIV). In fact this register contains the value corresponding to the number of UART BRM internal clock cycles present in one millisecond. The ONEMS (and UTIM) registers value are used in the escape character detection feature (Escape Sequence Detection) to count the number of clock cycles left between two escape characters. The ONEMS register is also used in infrared special case mode (IRSC = UCR4[5] = 1'b1), see InfraRed Special Case (IRSC) Bit.

53.15.16 UART Test Register (UARTx_UTS)

Address: Base address + B4h offset



UARTx_UTS field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13 FRCPERR	Force Parity Error. Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging. 0 Generate normal parity 1 Generate inverted parity (error)
12 LOOP	Loop TX and RX for Test. Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP. If RXDMUXSEL (UCR3[2]) is set to 1, the loopback is applied on serial and IrDA signals. If RXDMUXSEL is set to 0, the loopback is only applied on serial signals. 0 Normal receiver operation 1 Internally connect the transmitter output to the receiver input
11 DBGEN	This bit is not used in this chip. debug_enable_B**.** This bit controls whether to respond to the *debug_req* input signal. 0 UART will go into debug mode when *debug_req* is HIGH 1 UART will not go into debug mode even if *debug_req* is HIGH
10 LOOPIR	**Loop TX and RX for IR Test (LOOPIR)**. This bit controls loopback from transmitter to receiver in the InfraRed interface. 0 No IR loop 1 Connect IR transmitter to IR receiver
9 RXDBG	This bit is not used in this chip. **RX_fifo_debug_mode.** This bit controls the operation of the RX fifo read counter when in debug mode. 0 rx fifo read pointer does not increment 1 rx_fifo read pointer increments as normal

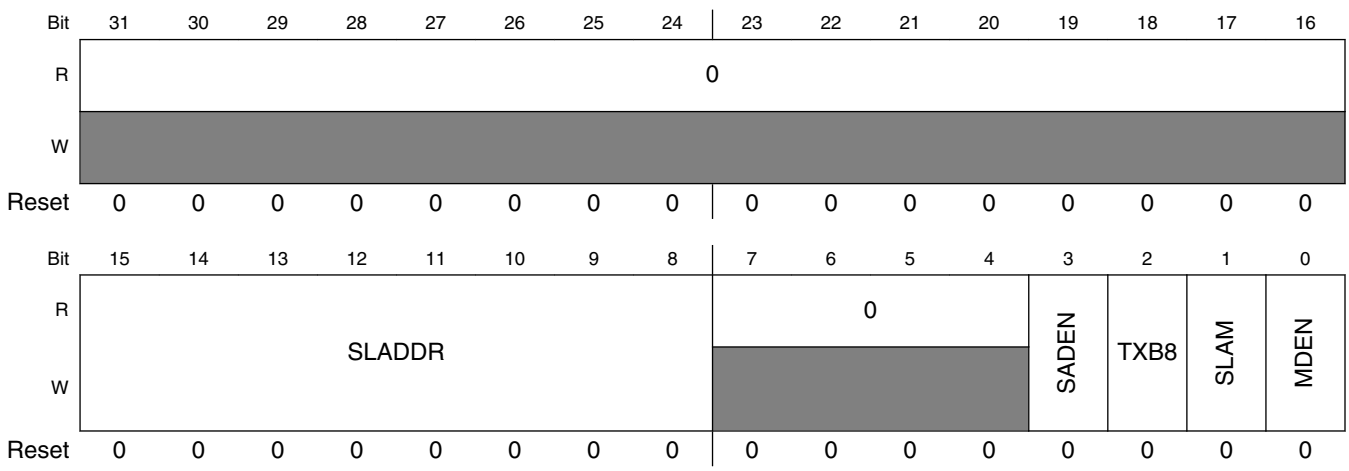
Table continues on the next page...

UARTx_UTS field descriptions (continued)

Field	Description
8–7 Reserved	This read-only field is reserved and always has the value 0.
6 TXEMPTY	TxFIFO Empty. Indicates that the TxFIFO is empty. 0 The TxFIFO is not empty 1 The TxFIFO is empty
5 RXEMPTY	RxFIFO Empty. Indicates the RxFIFO is empty. 0 The RxFIFO is not empty 1 The RxFIFO is empty
4 TXFULL	TxFIFO FULL. Indicates the TxFIFO is full. 0 The TxFIFO is not full 1 The TxFIFO is full
3 RXFULL	RxFIFO FULL. Indicates the RxFIFO is full. 0 The RxFIFO is not full 1 The RxFIFO is full
2–1 Reserved	This read-only field is reserved and always has the value 0.
0 SOFTRST	Software Reset. Indicates the status of the software reset (SRST_B bit of UCR2). 0 Software reset inactive 1 Software reset active

53.15.17 UART RS-485 Mode Control Register (UARTx_UMCR)

Address: Base address + B8h offset



UARTx_UMCR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–8 SLADDR	RS-485 Slave Address Character. Holds the selected slave adress character that the receiver wil try to detect.
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 SADEN	RS-485 Slave Address Detected Interrupt Enable. 0 Disable RS-485 Slave Address Detected Interrupt 1 Enable RS-485 Slave Address Detected Interrupt
2 TXB8	Transmit RS-485 bit 8 (the ninth bit or 9th bit). In RS-485 mode, software writes TXB8 bit as the 9th data bit to be transmitted. 0 0 will be transmitted as the RS485 9th data bit 1 1 will be transmitted as the RS485 9th data bit
1 SLAM	RS-485 Slave Address Detect Mode Selection. 0 Select Normal Address Detect mode 1 Select Automatic Address Detect mode
0 MDEN	9-bit data or Multidrop Mode (RS-485) Enable. 0 Normal RS-232 or IrDA mode, see Table 53-1 for detail. 1 Enable RS-485 mode, see Table 53-1 for detail

Chapter 54
Universal Serial Bus Controller (USB)

54.1 Overview

The USB controller block provides high performance USB functionality that conforms to the *Universal Serial Bus Specification*, Rev. 2.0 (Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips; 2000), and the *On-The-Go and Embedded Host Supplement to the USB Revision* 2.0 Specification (Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, Renesas Electronics Corporation, ST-Ericsson; 2012).

The USB controller consists of two independent USB controller cores: two On-The-Go (OTG) controller cores. Each controller core supports UTMI interface. See Features for more details.Both the controller cores are single-port cores. For the OTG cores, there is only one port. The port can be used as either a downstream or an upstream port.

The following figure is a block diagram of USB.

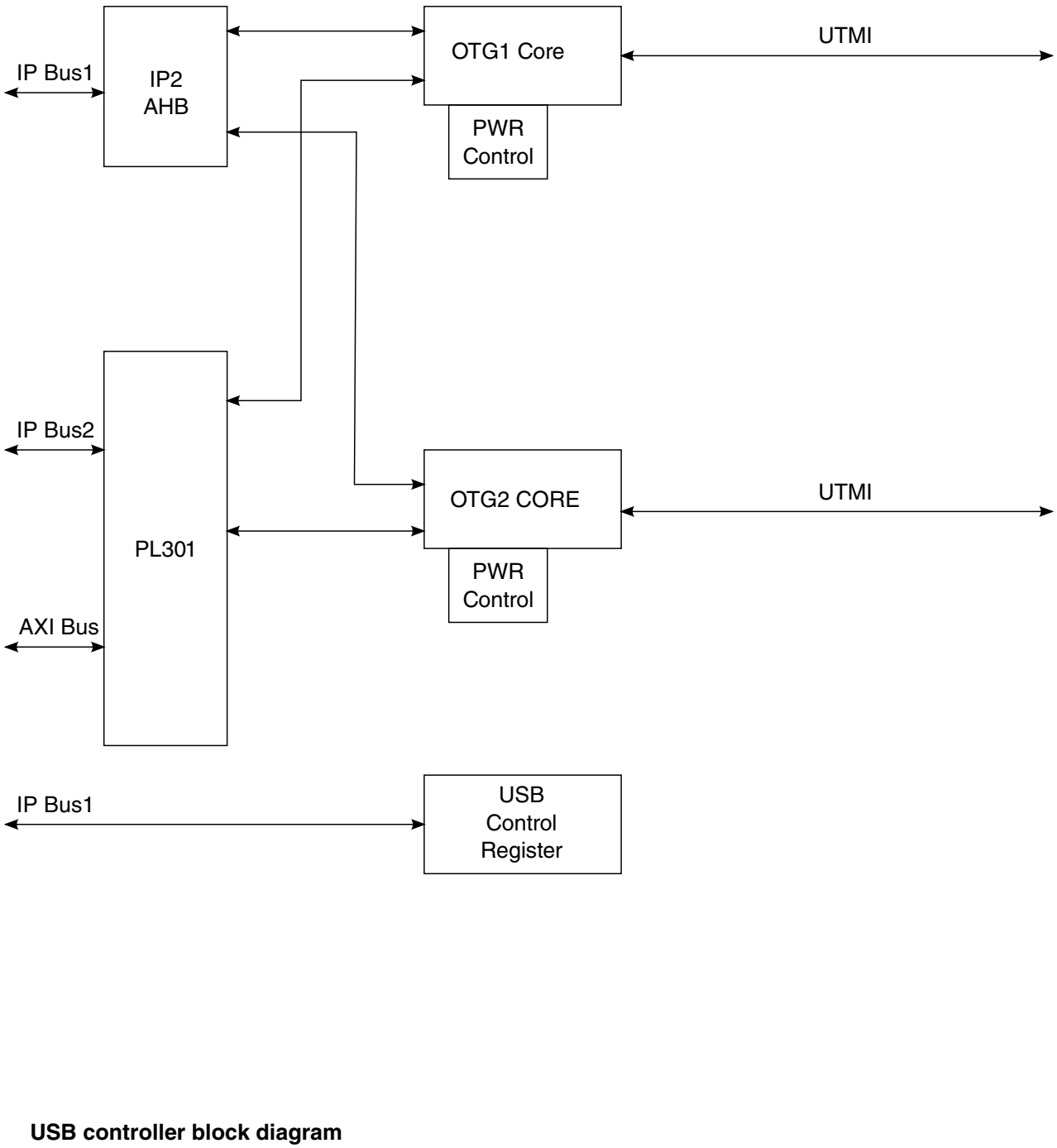


Figure 54-1. USB block diagram

54.1.1 Features

There are two USB 2.0 controller cores in this chip:

- Controller Core 0 is also named 'OTG1 Core'; its connected port is named 'OTG1 port'.
- Controller Core 1 is also named 'OTG2 Core'; its connected port is named 'OTG2 port'.

The following list provides features of each of the controller cores.

- USB 2.0 Controller Core 0
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface
  - High Speed, Full Speed and Low Speed operation in Host mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, session request protocol, and host negotiation protocol
  - Up to 8 bidirectional endpoints
  - Support charger detection
- USB 2.0 Controller Core 1
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface
  - High Speed, Full Speed and Low Speed operation in Host mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, session request protocol, and host negotiation protocol
  - Up to 8 bidirectional endpoints
- Low-power mode with local and remote wake-up capability
- Serial PHY interfaces configurable for bidirectional/unidirectional and differential/single ended
- Embedded DMA controller in each core

54.1.2 Modes of Operation

The USB has two main modes of operation: normal mode and low power mode.

Each USB OTG controller core can operate in High Speed operation (480 Mbps), Full Speed operation (12 Mbps) and Low Speed operation (1.5 Mbps).

This chapter explains the operation modes.

54.1.2.1 Normal Mode

The OTG controller core can operate in Host mode and Device (Peripheral) mode. The Host-only controller core can operate in Host mode only.

Each USB controller core has its corresponding port, which can work in one or more interface modes.

NOTE

Each controller supports only the interface type listed below. Selecting a different interface type in the PORTSC.PTS field results in unpredictable behavior and may cause the system to hang.

- OTG1 port
  - This port supports on-chip UTMI transceiver only.
- OTG2 port
  - This port supports on-chip UTMI transceiver only.

NOTE

HSIC is an inter-chip interface that is optimized for circuit board layouts.

54.1.2.2 Low-Power Mode

Each USB controller core has a low-power mode (Suspend mode) to save power consumption.

As described in the USB 2.0 specification, the device can go into the Suspend state after it sees a constant Idle state on the upstream facing port. The OTG controller core enters Suspend mode after 3 ms of inactivity on the port when it is in Device Operation mode. Host controllers, including the OTG controller in Host mode, do not suspend automatically but can be placed in Suspend mode by software.

Either the local ARM platform or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication. For details about Suspend/Resume, see USB Power Control.

54.2 External Signals

The table found here describes the external signals of USB.

Table 54-1. USB External Signals

Signal	Description	Pad	Mode	Direction
USB_OTG1_ID	ID signal	GPIO1_IO00	ALT2	I
		SD1_DATA0	ALT8	
		UART3_TX_DATA	ALT8	
USB_OTG1_OC	OTG External input for VBUS overcurrent detection	ENET2_RX_DATA1	ALT8	I
		GPIO1_IO01	ALT2	
		SD1_CLK	ALT8	
USB_OTG1_PWR	To control PMIC to supply VBUS voltage	ENET2_RX_DATA0	ALT8	O
		GPIO1_IO04	ALT2	
		SD1_CMD	ALT8	
USB_OTG2_ID	ID signal	ENET2_TX_CLK	ALT8	I
		GPIO1_IO05	ALT2	
		SD1_DATA3	ALT8	
USB_OTG2_OC	OTG External input for VBUS overcurrent detection	ENET2_TX_EN	ALT8	I
		GPIO1_IO03	ALT2	
		SD1_DATA2	ALT8	
USB_OTG2_PWR	To control PMIC to supply VBUS voltage	ENET2_TX_DATA1	ALT8	O
		GPIO1_IO02	ALT2	
		SD1_DATA1	ALT8	

54.3 Functional Description

These sections describe the functionality of the various building blocks of the USB.

54.3.1 USB 2.0 Controller Core 0

The USB 2.0 Controller 0 is an instantiation of an EHCI-compatible core which supports high-, full-, and low-speed operation.

In Host mode, this controller core supports high-, full-, and low-speed operation. In Device mode, it supports high- and full-speed operation.

54.3.1.1 Host Mode

The controller supports direct connection of .

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and protocol engine blocks to support connection to full and low speed devices.

54.3.1.2 Peripheral (Device) Mode

- Up to eight bidirectional endpoints
- High/full-speed operation
- Support of HNP and SRP
- Remote wake-up capability

54.3.2 USB 2.0 Controller Core 1

USB 2.0 Controller Core 1 is an instantiation of EHCI-compatible core which supports High Speed / Full Speed / Low Speed operation.

54.3.3 USB Power Control

The USB controller supports suspend and wake-up functionality.

The power control block allows for placing the transceiver in USB low power mode when USB bus is IDLE, and supports local and remote wake-up to bring the transceiver out of USB low power mode when needed. Additionally, the power control block can wake-up the ARM platform from core sleep mode by generating an interrupt.

54.3.3.1 Entering Low Power Suspend Mode

In Host operation mode, low power suspend mode is entered as follows:

1. Clear the ASE and PSE bits in USB_USBCMD, and wait until the AS and PS bits in USB_USBSTS become "0".
2. Set the "SUSPEND" bit in USB_PORTSC1.
3. Set the "PHCD" bit in USB_PORTSC1.

For device operation mode, low power suspend mode is entered as follows:

1. After Host drive is IDLE for 3ms, an SLI interrupt is issued (the "DCSUSPEND" or "SLI" bit in USB_USBSTS).
2. Set the "PHCD" bit on USB_PORTSC1

54.3.3.2 Wake-Up Events

The power control block monitors the USB bus when the USB core is in the USB suspend state.

Depending on whether the core is on Host or Device mode, a number of wake-up conditions are monitored. Upon detection of a wake-up condition, an interrupt (asynchronous) will be generated to ARM platform if the related wake-up interrupt enable bit is set.

USB wake-up interrupt also re-activates the ARM platform clocks if they were stopped during the suspend.

54.3.3.2.1 Host Mode Events

The host controller wakes up on the following events:

- Remote Wake-up Request

A peripheral can request the host to reactivate the bus by driving wake-up signaling on the DM/DP lines. The power control block sends a wake-up request to the USB core when a J-K transition on DM/DP line is detected.

- Wake-Up On Overcurrent

If Wake-Up On Overcurrent is enabled (WKOC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when an overcurrent event is detected.

- Wake-Up On Disconnect

If Wake-Up On Disconnect is enabled (WKDC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when a disconnection event is detected (J-SE0/K-SE0 transition on DM/DP line).

- Wake-Up On Connect

If a Wake-Up On Connect is enabled (WKCN bit in the USB core register PORTSC1 is set '1'), the power control sub-block sends a wake-up request to the USB core when the connection event is detected (SE0-J/SE0-K transition on DM/DP line).

For a detailed description of register bits WKOC, WKDC, WKCN, please see Port Status & Control (USB_*n*PORTSC1).

54.3.4 Interrupts

54.3.4.1 USB Core Interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB Cores. Refer to the Interrupt Enable Register (USB_*n*USBINTR) for details.

54.3.4.2 USB Wake-Up Interrupts

Each USB Core has an associated wake-up interrupt. The wake-up interrupts are generated outside of the USB controller cores, but using the same vector as the corresponding USB controller cores interrupt.

These interrupts are generated by the Power Control blocks which run on the 32 KHz standby clock. The wake-up interrupt is designed to work even when the USB and ARM platform clocks are disabled, such that a wake-up condition on the USB bus can re-activate the ARM platform clocks.

Because the wake-up interrupt is generated and cleared on a 32 KHz clock, this interrupt request responds very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt masks the request instantaneously as this is clocked by the ARM platform clock. The software should wait for at least three 32 KHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. Because this interrupt is only used during low power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to entering the USB suspend mode.

54.4 USB Operation Model

54.4.1 Register Interface

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

NOTE

USB controller registers support only DWORD (32-bit) access.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are dynamic control or status registers that may be read only, read/write, or read/write-to-clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

The following table describes the Interface register sets.

Table 54-2. Interface Register Sets

Offset	Register Set	Explanation
000h-07Ch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h-124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
080h-0FCh 140h-1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

54.4.1.1 Configuration, Control and Status Register Set

The following table describes the Device/Host capability registers.

NOTE

Depending on implementation, "*x*" can have the following values: UOG1, UOG2.

.

Table 54-3. Device/Host Capability Registers

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
000h	4	USB_*x*_ID	Identification Register	O	O
004h	4	USB_*x*_HWGENERAL	General Hardware Parameters	O	O
008h	4	USB_*x*_HWHOST	Host Hardware Parameters	X	O
00Ch	4	USB_*x*_HWDEVICE	Device Hardware Parameters	O	X
010h	4	USB_*x*_HWTXBUF	TX Buffer Hardware Parameters	O	O
014h	4	USB_*x*_HWRXBUF	RX Buffer Hardware Parameters	O	O
018-07Fh		-	Reserved		
080h	4	USB_*x*_GPTIMER0LD	General Purpose Timer #0 Load Register	O	O
084h	4	USB_*x*_GPTIMER0CTRL	General Purpose Timer #0 Control Register	O	O
088h	4	USB_*x*_GPTIMER1LD	General Purpose Timer #1 Load Register	O	O
08Ch	4	USB_*x*_GPTIMER1CTRL	General Purpose Timer #1 Control Register	O	O
090h	4	USB_*x*_SBUSCFG	System Bus Interface Configuration Register	O	O
094-09Fh		-	Reserved		
100h	1	USB_*x*_CAPLENGTH	Capability Register Length	O	O
101h		-	Reserved		
102h	2	USB_*x*_HCIVERSION	Host Controller Interface Version Number	X	O
104h	4	USB_*x*_HCSPARAMS	Host Controller Structural Parameters	X	O
108h	4	USB_*x*_HCCPARAMS	Host Controller Capability Parameters	X	O
10C-11Fh		-	Reserved		
120h	2	USB_*x*_DCIVERSION	Device Controller Interface Version Number	O	X
122h	2	-	Reserved		
124h	4	USB_*x*_DCCPARAMS	Device Controller Capability Parameters	O	X
128-13Fh		-	Reserved		
140h	4	USB_*x*_USBCMD	USB Command Register	O	O
144h	4	USB_*x*_USBSTS	USB Status Register	O	O
148h	4	USB_*x*_USBINTR	USB Interrupt Enable Register	O	O
14Ch	4	USB_*x*_FRINDEX	USB Frame Index	O	O
150h	4	-	Reserved		
154h	4	USB_*x*_PERIODICLISTBASE	Frame List Base Address	X	O
		USB_*x*_DEVICEADDR	USB Device Address	O	X
158h	4	USB_*x*_ASYNCLISTADDR	Next Asynchronous List Address	X	O
	4	USB_*x*_ENDPOINTLISTADDR	Address at Endpoint list in memory	O	X
15Ch	4	-	Reserved		
160h	4	USB_*x*_BURSTSIZE	Programmable Burst Size	O	O
164h	4	USB_*x*_TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning	X	O

Table continues on the next page...

Table 54-3. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
168 - 174h		-	Reserved		
178h	4	USB_*x*_ENDPTNAK	Endpoint NAK register	O	X
17Ch	4	USB_*x*_ENDPTNAKEN	Endpoint NAK Enable register	O	X
180h	4	USB_*x*_CONFIGFLAG	Configured Flag Register	X	O
184h	4	USB_*x*_PORTSC1	Port Status/Control Register 1	O	O
188-1A3h		-	Reserved		
1A4h	4	USB_*x*_OTGSC	On-The-Go Status/Control Register	O	O
1A8h	4	USB_*x*_USBMODE	USB Controller Operating Mode	O	O
1ACh	4	USB_*x*_ENDPTSETUPSTAT	Endpoint Setup Status	O	X
1B0h	4	USB_*x*_ENDPTPRIME	Endpoint Initialization	O	X
1B4h	4	USB_*x*_ENDPTFLUSH	Endpoint De-Initialization	O	X
1B8h	4	USB_*x*_ENDPTSTATUS	Endpoint Status	O	X
1BCh	4	USB_*x*_ENDPTCOMPLETE	Endpoint Complete	O	X
1C0 1C4 ... 1DCh	64	USB_*x*_ENDPTCTRL0 USB_*x*_ENDPTCTRL1 .... USB_*x*_ENDPTCTRL7	Endpoint Control Register 0-7	O	X

NOTE

"O" means the register is available in host/device operation mode;

"X" means the register is reserved in host/device operation mode

54.4.1.2 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

54.4.1.3 OTG Operations

54.4.1.3.1 Register Bits

In the previous section, the Register interface has behaviors described for device mode and behaviors described for host mode. However, for OTG operations it is necessary to perform tasks independent of the controller mode.

The following figure shows the controller mode.

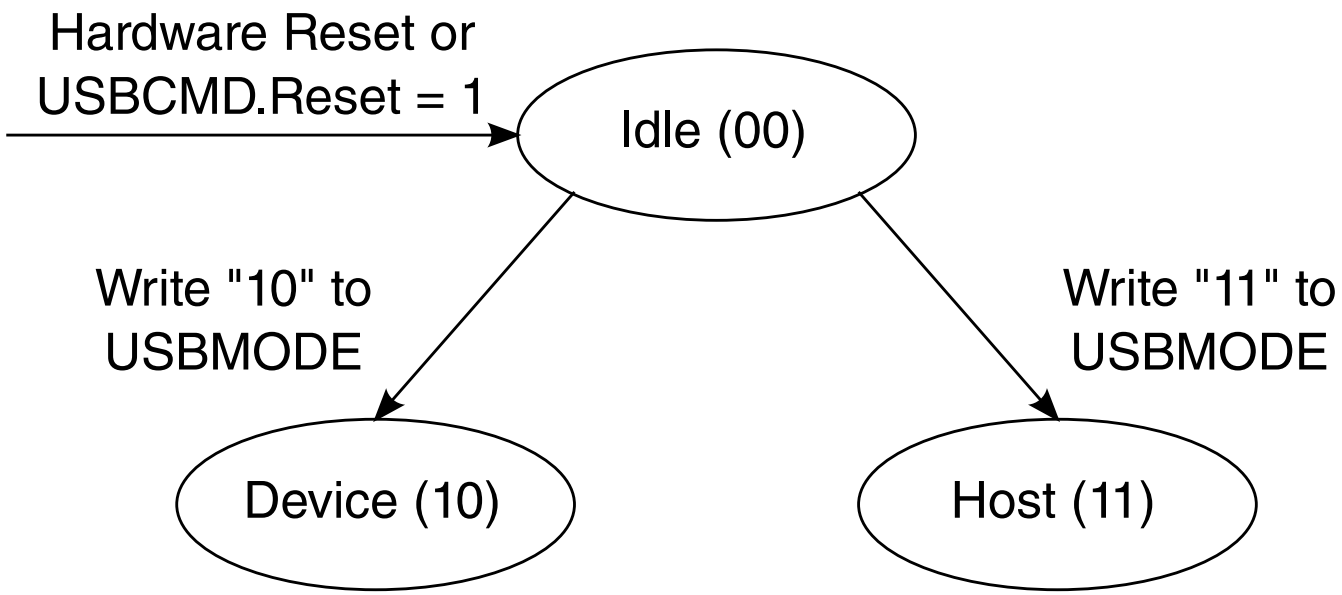


Figure 54-2. Controller Mode

To this end, listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

All Identification Registers

All Device/Host Capability Registers

OTGSC: All bits

PORTSC1:

Physical Interface Select

Physical Interface Serial Select

Physical Interface Data Width

Physical Interface Low Power

Physical Interface Wake Signals

Port Indicators

Port Power

54.4.2 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware).

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a Periodic Schedule, Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split-transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) transfers for the host controller. The asynchronous list is the root for all the bulk and control transfers. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split-transaction data streams are managed with Split-transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4 K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writeable fields. The host controller must preserve the read-only fields on all data structure writes.

54.4.2.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the USB_PERIODICLISTBASE address register and the USB_FRINDEX register.

The periodic schedule is based on an array of pointers called the Periodic Frame List.

The USB_PERIODICLISTBASE address register is combined with the USB_FRINDEX register to produce a memory pointer into the frame list. The Periodic Frame List implements a sliding window of work over time.

The following figure shows the organization of periodic schedule.

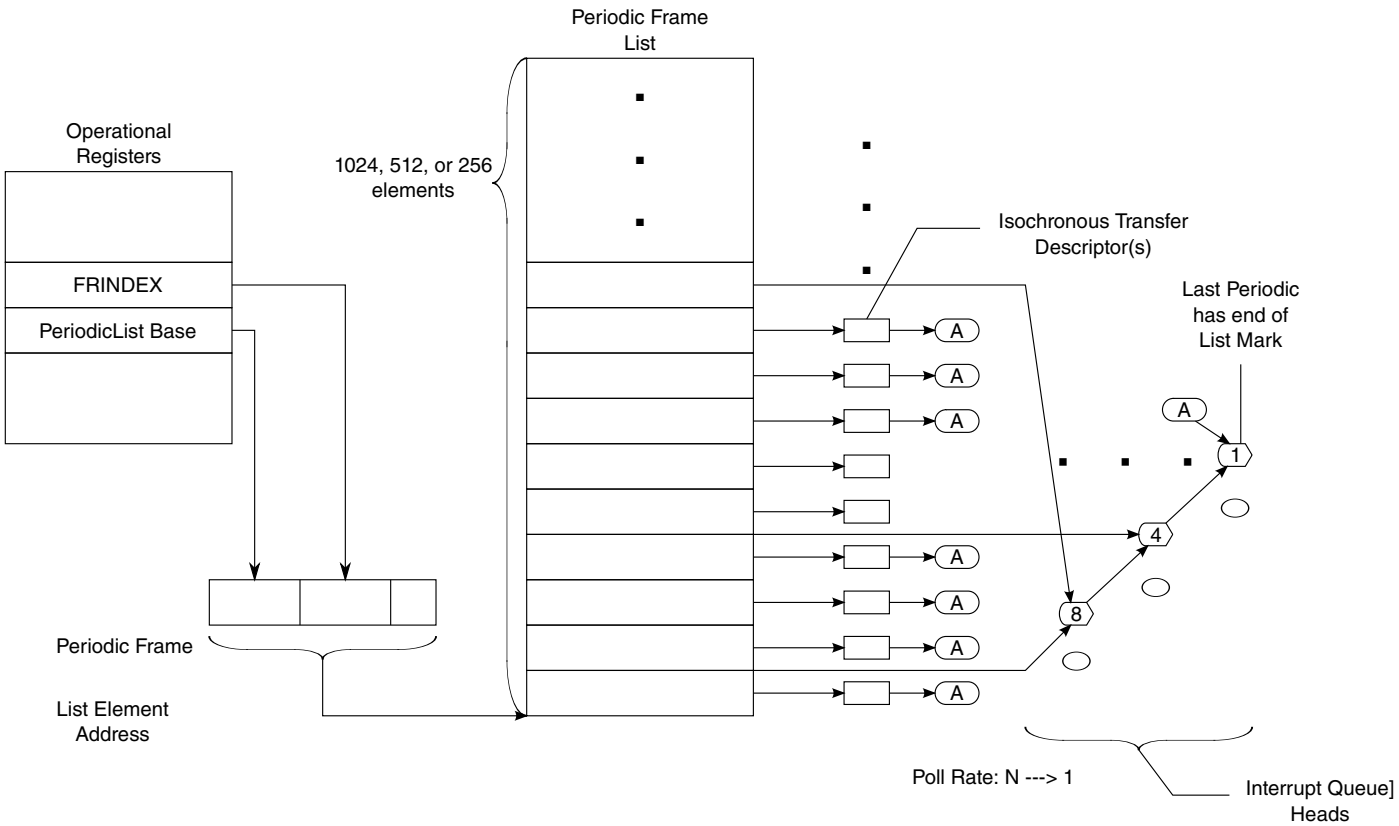


Figure 54-3. Periodic Schedule Organization

Split transaction Interrupt, Bulk and Control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4 K-page aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the USB_HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must support all three sizes. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into Frame List Size field in the USB_USBCMD register.

Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on DWord boundaries within the Frame List.

The table below illustrates the format of the Frame list element pointer.

Table 54-4. Format of Frame List Element Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Frame List Link Pointer																											0		Typ			03-00H

Frame List Link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing.

The least significant bit is the T-Bit (bit 0). When this bit is set to a one, the host controller never uses the value of the frame list pointer as a physical memory pointer. The Typ field is used to indicate the exact type of data structure being referenced by this pointer. The value encodings are.

Table 54-5. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

54.4.2.2 Asynchronous List Queue Head Pointer

The Asynchronous Transfer List (based at the USB_ASYNCLISTADDR register) is where all of the control and bulk transfers are managed.

Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

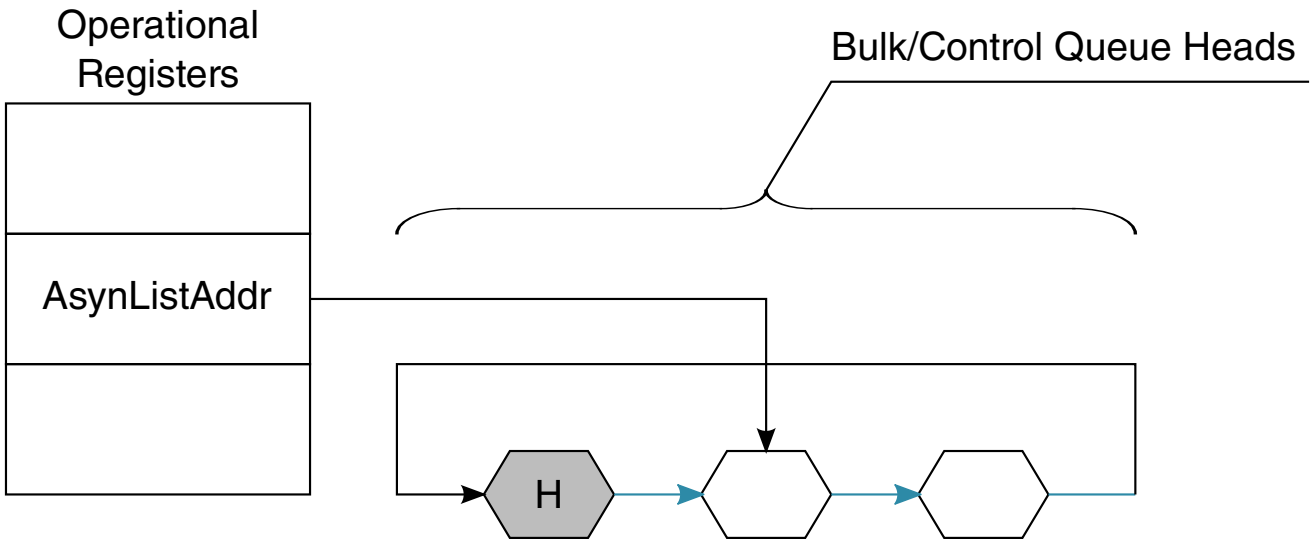


Figure 54-4. Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The USB_ASYNCLISTADDR register is simply a pointer to the next queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

54.4.2.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is shown in the table below.

This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

Table 54-6. Isochronous Transaction Descriptor (iTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Next Link Pointer																											0		Typ		T	03-00 H
Status				Transaction 0 Length												IOC	PG*			Transaction 0 Offset*												07-04 H
Status				Transaction 1 Length												IOC	PG*			Transaction 1 Offset*												0B-08H
Status				Transaction 2 Length												IOC	PG*			Transaction 2 Offset*												0F-0CH
Status				Transaction 3 Length												IOC	PG*			Transaction 3 Offset*												13-10 H

Table continues on the next page...

Table 54-6. Isochronous Transaction Descriptor (iTD) (continued)

| | | |

Table 54-7. Next Schedule Element Pointer (continued)

Reserved	
2-1 QH/(s)iTD Select (Typ)	This field indicates to the Host Controller whether the item referenced is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0 Terminate (T)	1= Link Pointer field is not valid. 0= Link Pointer field is valid.

54.4.2.3.2 iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status.

Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and Transaction X Offset fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB.

The following table describes iTD Transaction Status and Control fields.

Table 54-8. iTD Transaction Status and Control

Bit	Description	
31-28 Status	This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	Definition
	31	Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.
	30	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.
	29	Babble Detected. Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.

Table continues on the next page...

Table 54-8. iTD Transaction Status and Control (continued)

Bit	Description	
	28	Transaction Error (XactErr). Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27-16 Transaction X Length	For an OUT, this field is the number of data bytes the host controller sends during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (0‡zero length data, 1‡one byte, 2‡two bytes, etc.). The maximum value this field may contain is 0xC00 (3072).	
15 Interrupt On Complete (IOC)	If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.	
14-12 Page Select (PG)	These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.	
11-0 Transaction X Offset	This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.	

54.4.2.3.3 iTD Buffer Page Pointer List (Plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4 K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous.

Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Because each pointer is a 4 K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

The tables below illustrate the field descriptions.

Table 54-9. iTD Buffer Pointer Page 0 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 0)	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-8 Endpoint Number (Endpt)	This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.

Table continues on the next page...

Table 54-9. iTD Buffer Pointer Page 0 (Plus) (continued)

Bit	Description
7 Reserved	Bit reserved for future use and should be initialized by software to zero.
6-0 Device Address	This field selects the specific device serving as the data source or sink.

Table 54-10. iTD Buffer Pointer Page 1 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 1)	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11 Direction (I/O)	0 = OUT; 1 = IN. This field encodes whether the high-speed transaction should use an IN or OUT PID.
10-0 Maximum Packet Size	This directly corresponds to the maximum packet size of the associated endpoint *(wMaxPacketSize)*. This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (per micro-frame). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 54-11. iTD Buffer Pointer Page 2 (Plus)

Bit	Description
31-12 Buffer Pointer	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-2 Reserved	This bit reserved for future use and should be set to zero.
1-0 Multi	This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (per micro-frame). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro- frame. 10b Two transactions to be issued for this endpoint per micro- frame. 11b Three transactions to be issued for this endpoint per micro- frame.

Table 54-12. iTD Buffer Pointer Page 3-6

Bit	Description
31-12 Buffer Pointer	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-0 Reserved	These bits reserved for future use and should be set to zero.

54.4.2.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

The following table shows the Split Transaction Isochronous Transfer Descriptor (siTD).

Table 54-13. Split Transaction Isochronous Transfer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next Link Pointer																											0		Typ		T	03-00
I/O	Port Number							-	Hub Addr							Reserved				EndPt				-	Device Address							07-04[1]
Reserved																µFrame C-mask								µFrame S-mask								0B-08[1]
ioc	P	Reserved				Total Bytes to Transfer										µFrame C-prog-mask								Status								0F-0C[2]
Buffer Pointer (Page 0)																				Current Offset												13-10[2]
Buffer Pointer (Page 1)																				Reserved							TP		T-count			17-14[2]
Back Pointer																											0				T	1B-18

1. 04-0B: Static Endpoint State
2. 0C-13: Transfer results

Host Controller Read/Write

Host Controller Read Only

54.4.2.4.1 Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure.

The following table describes the Next Link Pointer fields.

Table 54-14. Next Link Pointer

Bit	Description
31-5	Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.

Table continues on the next page...

Table 54-14. Next Link Pointer (continued)

Bit	Description
4-3	Reserved. These bits must be written as zeros.
2-1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1 = Link Pointer field is not valid. 0 = Link Pointer is valid.

54.4.2.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

The tables below describe the Endpoint and transaction translator characteristics and micro-frame schedule control fields.

Table 54-15. Endpoint and Transaction Translator Characteristics

Bit	Description
31	Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the full-speed transaction should be an IN or OUT.
30-24	Port Number. This field is the port number of the recipient Transaction Translator.
23	Reserved. Bit reserved and should be set to zero.
22-16	Hub Address. This field holds the device address of the Companion Controllers' hub.
15-12	Reserved. Field reserved and should be set to zero.
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit is reserved for future use. It should be set to zero.
6-0	Device Address. This field selects the specific device serving as the data source or sink.

Table 54-16. Micro-frame Schedule Control

Bit	Description
31-16	Reserved. This field reserved for future use. It should be set to zero.
15-8	Split Completion Mask (mFrame C-Mask). This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host

Table continues on the next page...

Table 54-16. Micro-frame Schedule Control (continued)

Bit	Description
	controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Split Start Mask (mFrame S-mask). This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

54.4.2.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer.

The following table describes siTD transfer state fields.

Table 54-17. siTD Transfer Status and Control

Bit	Description
31	Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it asserts a hardware interrupt at the next interrupt threshold.
30	Page Select (P). Used to indicate which data page pointer should be concatenated with the *CurrentOffset* field to construct a data buffer pointer (0 selects *Page 0* pointer and 1 selects *Page 1*). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
29-26	Reserved. This field reserved for future use and should be set to zero.
25-16	Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)
15-8	µFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.
7-0: Status—This field records the status of the transaction executed by the host controller for this slot. It is a bit vector with the encoding shown in the following rows.	
7	Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.
6	ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.
5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller transmits an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.
4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit is set only for IN transactions.
2	Missed Micro-Frame. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.

Table continues on the next page...

Table 54-17. siTD Transfer Status and Control (continued)

Bit	Description
1	Split Transaction State (SplitXstate). The bit encodings are: Value Meaning 00b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 01b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
0	Reserved. Bit reserved for future use and should be set to zero.

54.4.2.4.4 siTD Buffer Pointer List (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4 K (page) aligned buffer pointers.

The least significant 12 bits of each DWord are used as additional transfer state. The following table describes the siTD buffer pointer fields.

Table 54-18. Buffer Page Pointer List (plus)

Bit	Description
31-12	Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4 K page aligned physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* (see siTD Transfer State) specifies the *current* active pointer.
Bits 11-0 (Page 0)	Current Offset—The 12 least significant bits of the Page 0 pointer are the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
Bits 11-0 (Page 1)—The least significant bits of the Page 1 pointer are split into three subfields as shown in the following rows.	
11-5 (Page 1)	Reserved
4-3 (Page 1)	Transaction position (TP). This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: Value Meaning 00b All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01b Begin. This is the first data payload for a full- speed that is greater than 188 bytes. 10B Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes.

Table continues on the next page...

Table 54-18. Buffer Page Pointer List (plus) (continued)

Bit	Description
	11b End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.
2-0 (Page 1)	Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

54.4.2.4.5 siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD, and it cannot reference any other schedule data structure.

The following table describes the siTD back link pointer fields.

Table 54-19. siTD Back Link Pointer

Bit	Description
31-5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4-1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

54.4.2.5 Queue element transfer descriptor (qTD)

This data structure is only used with a queue head. It describes one or more USB transactions to transfer up to 20480 (5*4096) bytes.

The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers.

It is 32 bytes and must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary; however, for optimal utilization of on-chip busses it is recommended to align the buffers on a 32-byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

The following table shows the queue element transfer descriptor data structure.

Table 54-20. Queue element transfer descriptor data structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next qTD Pointer																											0				T	03-00
Alternate Next qTD Pointer																											0				T	07-04
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								0B-08[1]
Buffer Pointer (page 0)																				Current Offset												0F-0C[1]
Buffer Pointer (page 1)																				Reserved												13-10
Buffer Pointer (page 2)																				Reserved												17-14
Buffer Pointer (page 3)																				Reserved												1B-18
Buffer Pointer (page 4)																				Reserved												1F-1C

1. 08-0F: Transfer Results

Host Controller Read/Write

Host Controller Read Only

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

54.4.2.5.1 Next qTD Pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

The following table describes Next qTD pointer fields.

Table 54-21. qTD Next Element Transfer Pointer (DWord 0)

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4-1	Reserved

Table continues on the next page...

Table 54-21. qTD Next Element Transfer Pointer (DWord 0) (continued)

0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

54.4.2.5.2 Alternate Next qTD Pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next transfer descriptor on short packet. To be more explicit the host controller always uses this pointer when the current qTD is retired due to short packet.

The following table describes the TD Alternate Next Element Transfer Pointer field descriptions.

Table 54-22. TD Alternate Next Element Transfer Pointer (DWord 1)

Bit	Description
31-5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

54.4.2.5.3 qTD Token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE

The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

The following table describes the TD Token fields.

Table 54-23. TD Token (DWord 2)

Bit	Description
31 Data Toggle	This is the data toggle sequence bit. The use of this bit depends on the setting of the *Data Toggle Control* bit in the queue head.

Table continues on the next page...

Table 54-23. TD Token (DWord 2) (continued)

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>30-16
Total Bytes to Transfer</td><td>This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that Total Bytes To Transfer be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction is always less than QHD.Maximum Packet Length.

Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16 K(4000H).</td></tr>
<tr><td>15
Interrupt On Complete (IOC)</td><td>If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.</td></tr>
<tr><td>14-12
Current Page (C_Page)</td><td>This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.</td></tr>
<tr><td>11-10
Error Counter (CERR)</td><td>This field is a 2-bit down counter that keeps track of the number of consecutive Errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the Host Controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD inactive, sets the Halted bit to a one, and error status bit for the error that caused CERR to decrement to zero. An interrupt is generated if the USB Error Interrupt Enable bit in the USBINTR register is set to a one. If HCD programs this field to zero during set-up, the Host Controller does not count errors for this qTD and there is no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD.

Transaction Error - Decrement

Data Buffer Error - No Decrement[3]

Stalled - No Decrement[1]

Babble Detected - No Decrement[1]

No Error - No Decrement[2]
<table>
<tr><th>Error</th><th>Decrement Counter</th></tr>
<tr><td>1</td><td>Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented</td></tr>
<tr><td>2</td><td>If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset CERR to extend the total number of errors for this transaction. For example, CERR should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 00b.

See Split Transaction Interrupt for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See Asynchronous - Do Complete Split for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the PIDCode indicates a SETUP.</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

Table 54-23. TD Token (DWord 2) (continued)

Bit	Description	
	3	Data buffer errors are host problems. They don't count against the device's retries.
	NOTE: Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.	
9-8 PID Code	This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:	
	00b	OUT Token generates token (E1H)
	01b	IN Token generates token (69H)
	10b	SETUP Token generates token (2DH) (undefined if endpoint is an interrupt, the queue head is non-zero) transfer type, for example, µ*Frame S-mask* field in.
	11b	Reserved
7-0 Status	This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are:	
	Bit	Status Field Description
	7	Active. Set to one by software to enable the execution of transactions by the Host Controller.
	6	Halted. Set to one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero.
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller forces a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Because "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	2	Missed Micro-Frame. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	1	Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split-

Table continues on the next page...

Table 54-23. TD Token (DWord 2) (continued)

Bit	Description	
		transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are: Value Meaning 0b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint.
	0	Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: Value Meaning 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

54.4.2.5.4 qTD Buffer Page Pointer List

The last five DWords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes Current Offset field to the starting offset into the current page, where current page is selected through the value in the C_Page field.

The following table describes the qTD Buffer Pointer(s) (DWords 3-7) fields.

Table 54-24. qTD Buffer Pointer(s) (DWords 3-7)

Bit	Description
31-12	Buffer Pointer List. Each element in the list is a 4 K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4 K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction through the new buffer pointer.
11-0	Current Offset (Reserved). This field is reserved in all pointers except the first one (for example Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zero.

54.4.2.6 Queue Head

The following table shows the queue head structure layout.

Table 54-25. Queue Head Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Queue Head Horizontal Link Pointer																											0		Typ		T	03-00
RL				C	Maximum Packet Length											H	dtc	EP		EndPt				I	Device Address							07-04[1]
Mult		Port Number[2]							Hub Addr[2]							μFrame C-mask[2]								μFrame S-mask								0B-08[1]
Current qTD Pointer																											0					0F-0C
Next qTD Pointer																											0				T	13-10[3]
Alternate Next qTD pointer																											NakCnt				T	17-14[4]
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								1B-18
Buffer Pointer (Page 0)																				Current Offset												1F-1C
Buffer Pointer (Page 1)																				Reserved				C-prog-mask[2]								23-20
Buffer Pointer (Page 2)																				S-bytes[2]									FrameTag[2]			27-24[4]
Buffer Pointer (Page 3)																				Reserved												2B-28
Buffer Pointer (Page 4)																				Reserved												2F-2C[3]

1. 04-0B: Static endpoint state.
2. These fields are used exclusively to support split transactions to USB 2.0 hubs
3. 10-2F: Transfer overlay.
4. 14-27: Transfer results.

Host Controller Read/Write

Host Controller Read Only

54.4.2.6.1 Queue Head Horizontal Link Pointer

The first DWord of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

The following table describes the Queue head DWord 0 fields.

Table 54-26. Queue Head DWord 0

Bit	Description
31-5	Queue Head Horizontal Link Pointer (QHLP). This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD Select (Typ). This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

54.4.2.6.2 Queue Head Endpoint Capabilities/Characteristics

The second and third DWords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint.

There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.
- Endpoint Capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

The following table describes the Endpoint characteristics: Queue head DWord 1 fields.

Table 54-27. Endpoint Characteristics: Queue Head DWord 1

Bit	Description	
31-28	Nak Count Reload (RL). This field contains a value, which is used by the host controller to reload the Nak Counter field.	
27	Control Endpoint Flag (C). If the *QH.EPS* field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to zero.	
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). The maximum value this field may contain is 0x400 (1024).	
15	Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.	
14	Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0b Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1b Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.	
13-12	Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are:	
	Value	Meaning
	00b	Full-Speed (12 Mbits/sec)
	01b	Low-Speed (1.5 Mbits/sec)
	10b	High-Speed (480 Mbits/sec)
	11b	Reserved
	This field must not be modified by the host controller.	
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.	
7	Inactivate on Next Transaction (I). This bit is used by system software to request that the host controller set the Active bit to zero. See Rebalancing the periodic schedule , for full operational details. This field is only valid when the queue head is in the Periodic Schedule and the *EPS* field indicates a Full or Low-speed endpoint. Setting this bit to one when the queue head is in the Asynchronous Schedule or the *EPS* field indicates a high-speed device yields undefined results.	
6-0	Device Address. This field selects the specific device serving as the data source or sink.	

The table below describes the Endpoint capabilities: Queue head DWord 2 field descriptions.

Table 54-28. Endpoint Capabilities: Queue Head DWord 2

Bit	Description
31-30	High-Bandwidth Pipe Multiplier (Mult). This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). The valid values are: Value Meaning

Table continues on the next page...

Table 54-28. Endpoint Capabilities: Queue Head DWord 2 (continued)

	00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame. 10b Two transactions to be issued for this endpoint per micro-frame. 11b Three transactions to be issued for this endpoint per micro-frame.
29-23	Port Number. This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22-16	Hub Addr. This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.
15-8	Split Completion Mask (µFrame C-Mask). This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the µ*Frame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Interrupt Schedule Mask (µFrame S-mask). This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the µ*Frame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

54.4.2.6.3 Transfer Overlay-Queue Head

The nine DWords in this area represent a transaction working space for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the Queue Head Horizontal Link Pointer to the next queue head. The host controller will never follow the Next Transfer Queue Element or Alternate Queue Element pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

The following table describes the current qTD link pointer field descriptions.

Table 54-29. Current qTD Link Pointer

Bit	Description
31-5	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4-0	Reserved (R). These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an overlay because when the queue is advanced to the next queue element, the source queue element is merged onto this area. This area serves as execution cache for the transfer.

The table below describes the Host-controller rules for bits in overlay.

Table 54-30. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8 and 9)

DWord	Bit	Description
5	4-1	Nak Counter (NakCnt)µRW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	Data Toggle. The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.
6	15	Interrupt On Complete (IOC). The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11-10	Error Counter (C_ERR). This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	Ping State (P)/ERR. If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7-0	Split-transaction Complete-split Progress (C-prog-mask). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4-0	Split-transaction Frame Tag (Frame Tag). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11-5	S-bytes. Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

54.4.2.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary.

See Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose USB_HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use yields undefined results.

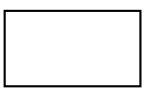
Table 54-31. Frame Span Traversal Node Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Normal Path Link Pointer																											0		Typ		T	03-00
Back Path Link Pointer																											0		Typ[1]		T	07-04

1. Must be set to indicate a queue head



Host Controller Read/Write



Host Controller Read Only

54.4.2.7.1 FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

The following table describes the FSTN normal path pointer fields.

Table 54-32. FSTN Normal Path Pointer Field Descriptions

Bit	Description
31-5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD/FSTN Select (Typ). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (Frame Span Traversal Node)
0	Terminate (T). 1 = Link Pointer field is not valid. 0 = Link Pointer is valid.

54.4.2.7.2 FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head.

If the T-bit in this pointer is zero, then this FSTN is a Save-Place indicator. Its Typ field must be set by software to indicate the target data structure is a queue head. If the T-bit in this pointer is set to one, then this FSTN is the Restore indicator. When the T-bit is one, the host controller ignores the Typ field.

The following table describes the FSTN back path link pointer fields.

Table 54-33. FSTN Back Path Link Pointer Field Descriptions

Bit	Description
31-5	Back Path Link Pointer (BPLP). This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	Typ. Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate (T). 1=Link Pointer field is not valid (that is the host controller must not use bits [31:5] as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (that is the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

54.4.3 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software).

Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

54.4.3.1 Host Controller Initialization

After initial power-on or HCReset (hardware or through HCReset bit in the USB_USBCMD register), all of the operational registers are at their default values. After a hardware reset, only the operational registers not contained in the Auxiliary power well are at their default values.

The following table describes the default values of operational registers.

.

Table 54-34. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USB_USBCMD	00080000h (00080B00h, if *Asynchronous Schedule Park Capability is one*)
USB_USBSTS	00001000h
USB_USBINTR	00000000h
USB_FRINDEX	00000000h
USB_CTRLDSSEGMENT	00000000h
USB_PERIODICLISTBASE	Undefined
USB_ASYNCLISTADDR	Undefined
USB_CONFIGFLAG	00000000h
USB_PORTSC1	00002000h (w/*PPC* set to one); 00003000h (w/*PPC* set to zero)

To initialize the host controller, software should perform the following steps:

- Write the appropriate value to the USB_USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the USB_PERIODICLIST BASE register. If no work items are in the periodic schedule, all elements of the Periodic Frame List should have their T-Bits set to one.
- Write the USB_USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller ON through setting the Run/Stop bit.

At this point, the host controller is up and running and the port registers begin reporting device connects, and so on. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled ports, but the schedules have not enabled. To communicate with devices through the asynchronous schedule, system software must write the USB_ASYNCLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing one to the Asynchronous Schedule Enable bit in the USB_USBCMD register. To communicate with devices through the periodic schedule, system software must enable the periodic schedule by writing one to the Periodic Schedule Enable bit in the USB_USBCMD register.

NOTE

The schedules can be turned on before the first port is reset (and enabled).

When the USB_USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

54.4.3.2 Port Routing and Control

The EHCI specification defines that a USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers.

Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; for example, Low-, Full-, and High-speed capability for every port.

NOTE

The USB controllers on i.MX parts do not require nor support companion controllers to support Full and Low Speed device. Full and Low Speed devices are supported within the USB controller by emulating the functionality of a high-speed HUB. Therefore, no port routing is present in the controller. Please refer to Embedded Transaction Translator Function for detail!

The following figure illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

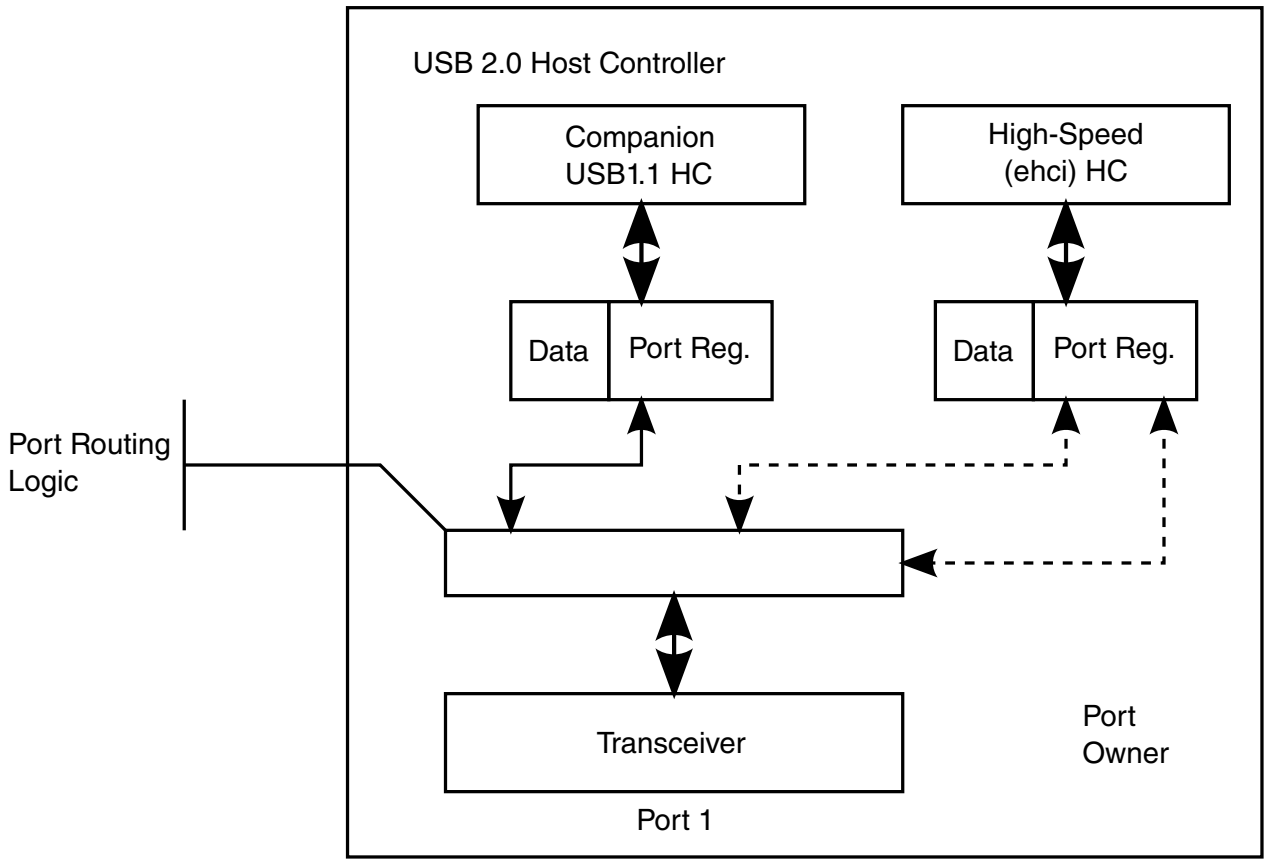


Figure 54-5. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller block has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Either the EHCI host controller or one companion host controller controls each transceiver. Routing logic lies between the transceiver, the port status and control registers.[1]

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a global routing policy control field and per-port ownership control fields. The Configured Flag (CF) bit is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports still work in Full- and Low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication

1. The routing logic should not be implemented in the 480 MHz clock domain of the transceiver.

from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (that is no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The N_CC field in the Structural Parameter register (HCSPARAMS) indicates whether the controller implementation includes companion host controllers. When N_CC has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports always fails the high-speed chirp during reset and the ports are not enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See Host Controller Structural Parameters (USB_*n*HCSPARAMS).

54.4.3.2.1 Port Routing Control through EHCI Configured (CF) Bit

Each port in the USB 2.0 host controller are routed either to a single companion host controller or to the EHCI host controller.

The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The Configured Flag (CF) bit, is used to globally set the policy of the routing logic. Each port register has a Port Owner control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the CF bit transitions from zero to one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all Port Owner bits go to zero). While the CF-bit is one, the EHCI Driver controls individual ports' routing through the Port Owner control bit. Likewise, whenever the CF bit transitions from one to zero (as a result of Aux power application, HCRESET, or software writing zero to CF-bit), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's CF bit (after Aux power application or HCRESET) is zero.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

The following table summarizes the default routing for all the ports, based on the value of the EHCI HC's CF bit.

Table 54-35. Default Port Routing Depending on EHCI HC CF Bit

HS CF Bit	Default Port Ownership	Explanation
0B	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1B	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this specification (see Port Status & Control (USB_*n*PORTSC1)). The EHCI HC can temporarily release control of the port to a companion HC by setting the *PortOwner* bit in the PORTSC1 register to one.

54.4.3.2.2 Port Routing Control through PortOwner and Disconnect Event

Manipulating the port routing through the CF-bit is an extreme process and not intended to be used during normal operation.

The normal mode of port ownership transferal is on the granularity of individual ports using the Port Owner bit in the EHCI HC's USB_PORTSC1 register (for hand-offs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI *CF-bit* is set to one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and

identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.

- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the LineStatus bits in the USB_PORTSC1 register. If they indicate the attached device is a full-speed device (for example, D+ is asserted), then the EHCI Driver sets the PortReset control bit to one (and sets the PortEnable bit to zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing zero to the port reset bit. The reset process is actually complete when software reads zero in the PortReset bit. The EHCI Driver checks the PortOwner bit in the USB_PORTSC1 register. If set to one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the LineStatus bits might indicate a low-speed device. Additionally, when the port reset process is complete, the PortEnable field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the USB_PORTSC1 register to one to release port ownership to a companion host controller.
- When the EHCI Driver sets PortOwner bit to one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI USB_PORTSC1 register observes and reports a disconnect event through the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to one, a connect change set to one and a connect status set to zero. This information is derived directly from the EHCI port register. This allows the hub driver to assume the device was disconnected during reset. It acknowledges the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected. The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's CF-bit transitions from1b to 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects is detected by the EHCI port register and the process repeats.

54.4.3.2.3 Example Port Routing State Machine

The following figure illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

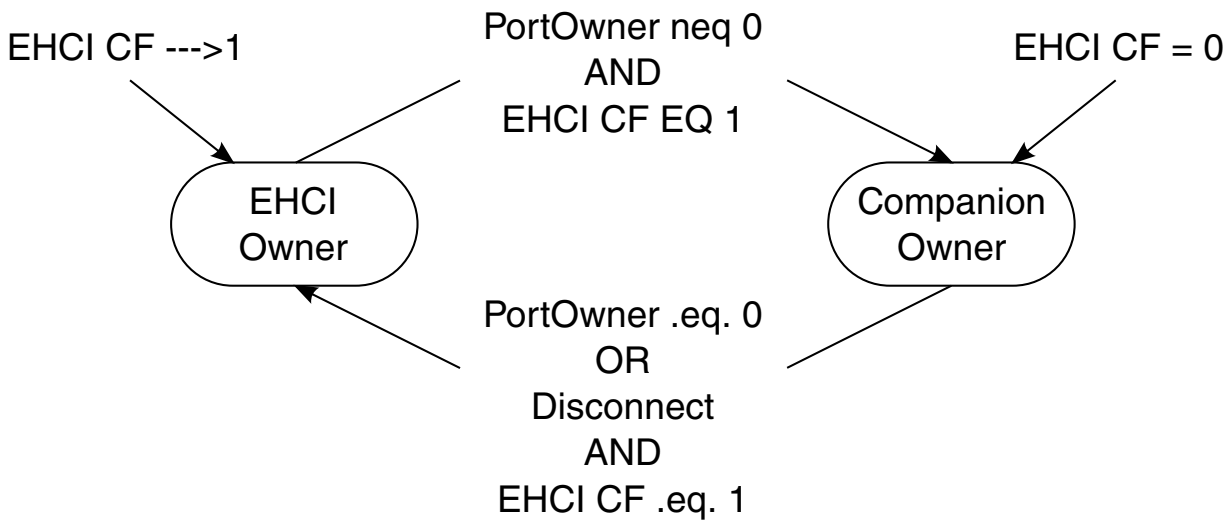


Figure 54-6. Port Owner Handoff State Machine

54.4.3.2.3.1 EHCI HC Owner

Entry to this state occurs when one of the following events occur:

- When the EHCI HC's Configure Flag (CF) bit in the USB_CONFIGFLAG register transitions from zero to one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.
- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver acknowledges the disconnect by setting the connect status change bit to zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes zero to the PortOwner bit in the USB_PORTSC1 register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

54.4.3.2.3.2 Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the PortOwner field transitions from zero to one.
- When the HS-mode HC's Configure Flag (CF) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

54.4.3.2.4 Port Power

The Port Power Control (PPC) bit in the USB_HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (see Host Controller Structural Parameters (USB_*n*HCSPARAMS)).

When this bit is zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the *PPC* bit is one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as PortPowerOutputEnable (PPE). PPE is controlled based on the state of the combination bits PPC bit, EHCI Configured (CF)-bit and individual Port Power (PP) bits.

The following table describes the summary behavioral model.

Table 54-36. Port Power Enable Control Rules

CF	CHC[1] (PP)	EHC[2] (PP)	Owner	PPE[3]	Description
0	0	X	CHC	0	When the EHCI controller is not configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	X	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.

Table continues on the next page...

Table 54-36. Port Power Enable Control Rules (continued)

1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

1. CHC (Companion Host Controller).
2. EHC (EHCI Host Controller).
3. PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

54.4.3.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI USB_PORTSC1 register has an over-current status and over-current change bit.

The functionality of these bits are specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document.

The over-current condition effects the following bits in the USB_PORTSC1 register on the EHCI port:

- Over-current Active bits are set to one. When the over-current condition goes away, the Over-current Active bit transitions from one to zero.
- Over-current Change bits are set to one. On every transition of the Over-current Active bit the host controller sets the Over-current Change bit to one. Software sets the Over-current Change bit to zero by writing one to this bit.
- Port Enabled/Disabled bit is set to zero. When this change bit gets set to one, then the Port Change Detect bit in the USB_USBSTS register is set to one.
- Port Power (PP) bits may optionally be set to zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to limit the current and leave power applied. When the Over-current Change bit transitions from zero to one, the host controller also sets the Port Change Detect bit in the USB_USBSTS register to one. In addition, if the Port Change Interrupt Enable bit in the USB_USBINTR register is one, then the host controller issues an interrupt to the system. Refer to Table 54-37 for summary behavior for over-current detection

when the host controller is halted (suspended from a device component point of view).

54.4.3.3 Suspend/Resume-Host Operational Model

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub.

Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely through software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wake-up events are:

- Remote-wake-up enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake-up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the USB_PORTSC1 registers.

Selective suspend is a feature supported by every USB_PORTSC1 register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the Run/Stop bit in the USB_USBCMD register to zero. The EHCI sub-block can then be placed into a lower device state through the PCI power management interface (see Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs, the system resumes operation and system software eventually set the Run/Stop bit to one and resume the suspended ports. Software must not set the Run/Stop bit to one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the ARM platform is restarted. So, by definition, if software is running, clocks in the system are stable and the Run/Stop bit in the USB_USBCMD register can be set to one. Minimum system software delays are also defined in the PCI Power Management Specification. Refer to PCI Power Management Specification for more information.

54.4.3.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing one into the appropriate USB_PORTSC1 Suspend bit. Software must only set the Suspend bit when the port is in the enabled state (Port Enabled bit is one) and the EHCI is the port owner (PortOwner bit is zero).

The host controller may evaluate the Suspend bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the Suspend bit. The host controller must evaluate the Suspend bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing one to the Force Port Resume bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see Port Status & Control (USB_*n*PORTSC1)). If system software sets Force Port Resume bit to one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 ms after a port indicates that it is suspended (Suspend bit is one) before initiating a port resume through the Force Port Resume bit. When Force Port Resume bit is one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 ms) then sets the Force Port Resume bit to zero. When the host controller receives the write to transition Force Port Resume to zero, it completes the resume sequence as defined in the USB specification, and sets both the Force Port Resume and Suspend bits to zero. Software-initiated port resumes do not affect the Port Change Detect bit in the USB_USBSTS register nor do they cause an interrupt if the Port Change Interrupt Enable bit in the USB_USBINTR register is one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 µsec. The port's Force Port Resume bit is set to one and the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit in the USB_USBINTR register is one the host controller issues a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 ms), then terminates the resume sequence by writing zero to the Force Port Resume bit in the port. The host controller receives the write of zero to Force Port Resume, terminates the resume sequence and sets Force Port Resume and Suspend port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the USB_PORTSC1 register and observing that the Suspend and Force Port Resume bits are zero. Software must ensure that the host controller is running (that is HCHalted bit in the USB_USBSTS register is zero), before terminating a resume by writing zero to a

port's Force Port Resume bit. If HCHalted is one when Force Port Resume is set to zero, then SOFs do not occur down the enabled port and the device returns to suspend mode in a maximum of 10 msec.

The table below summarizes the wake-up events. Whenever a resume event is detected, the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit is one in the USB_USBINTR register, the host controller generates an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the Port Change Detect status bit in the USB_USBSTS register.

Table 54-37. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	Not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in USB_PORTSC1 register is set to one. Port Change Detect bit in USB_USBSTS register set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is one. A disconnect is detected.	Depending in the initial port state, theUSB_ PORTSC1 Connected Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is zero. A disconnect is detected.	Depending on the initial port state, the USB_PORTSC1 Connect and Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is not connected and the port's WKCNNT_E bit is one. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is zero. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is connected and the port's WKOC_E bit is one. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is zero. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USB_USBINTR register is one.

[2] PME# asserted if enabled (Note: PME Status must always be set to one).

[3] PME# not asserted.

54.4.3.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule.

The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware / software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the USB_PERIODICLISTBASE register (see Frame List Base Address (USB_*n*PERIODICLISTBASE))/ Device Address (USB_*n*DEVICEADDR). The USB_PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structures. In each micro-frame, if the periodic schedule is enabled (see Periodic scheduling threshold) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It only executes from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the USB_PERIODICLISTBASE and the USB_FRINDEX registers (see the following figure). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a next link pointer of a schedule data structure having its T-bit set to one. When the host controller encounters a T-Bit set to one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. After the transition, the host controller executes from the asynchronous schedule until the end of the micro-frame.

The following figure illustrates the derivation of pointer into frame list array.

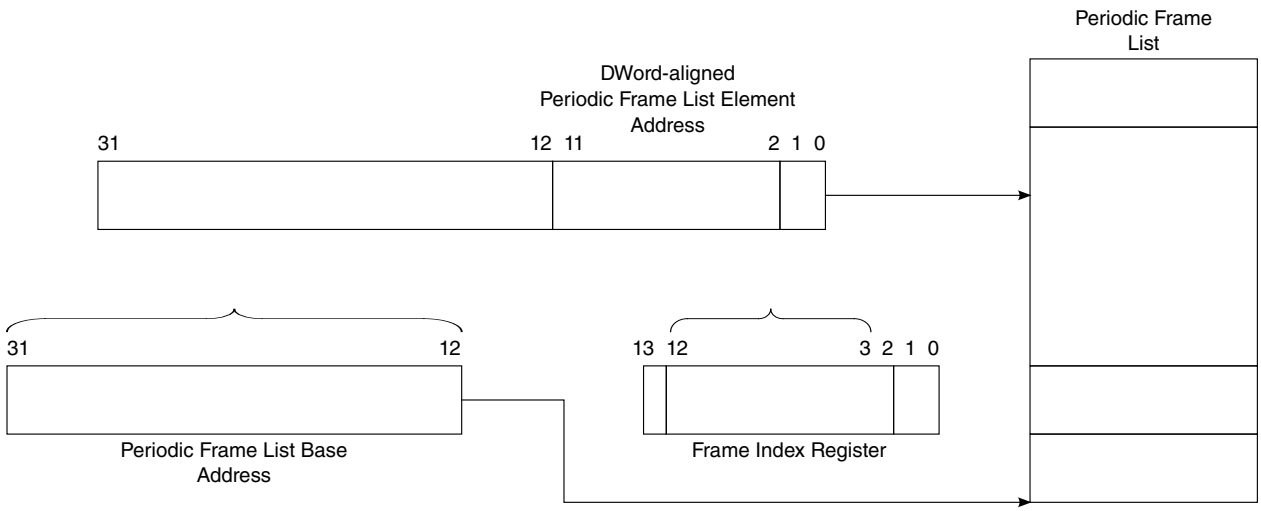


Figure 54-7. Derivation of Pointer into Frame List Array

When the host controller determines that it is the time to execute from the asynchronous list, it uses the operational register USB_ASYNCLISTADDR to access the asynchronous schedule, see the figure below.

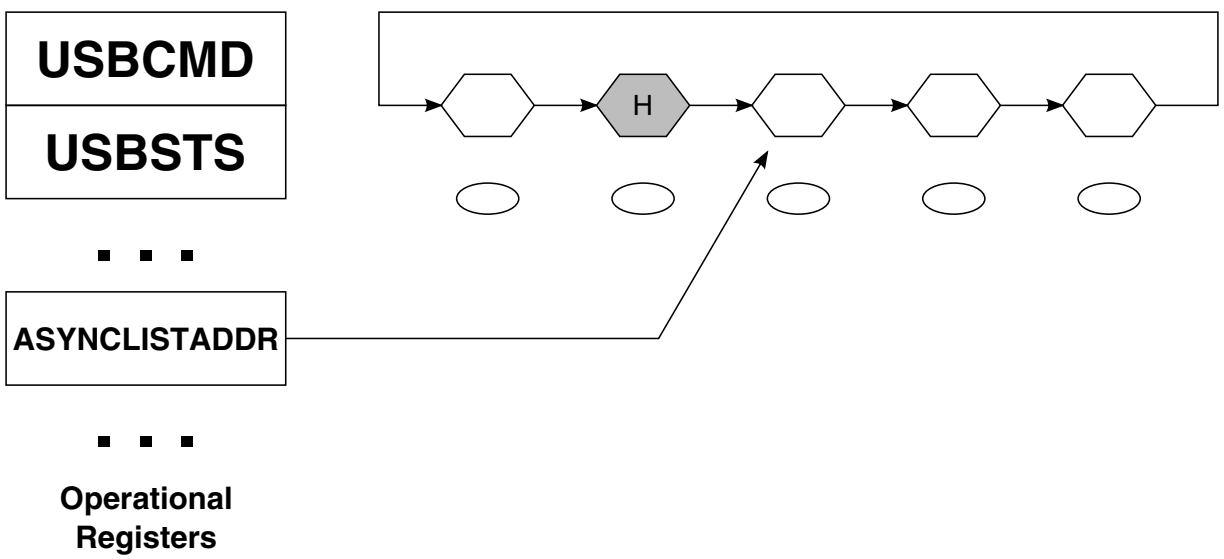


Figure 54-8. General Format of Asynchronous Schedule List

The USB_ASYNCLISTADDR register contains a physical memory pointer to the next queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the USB_ASYNCLISTADDR register. Software must set queue head horizontal pointer T-bits to zero for queue heads in the asynchronous schedule. See Asynchronous Schedule for complete operational details.

54.4.3.4.1 Example - Preserving Micro-Frame Integrity

One of the requirements of a USB host controller is to maintain Frame Integrity. This means that the HC must preserve the micro-frame boundaries.

For example, SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of micro-frame timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One implication of this responsibility is that the HC must ensure that it does not start transactions that do not complete before the end of the micro-frame. More precisely, no transactions should be started by the host controller, which do not complete in their entirety before the EOF1 point. In order to enforce this rule, the host controller must check each transaction before it starts to ensure that it completes before the end of the micro-frame.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction takes. Take OUTs for an example. The host controller must fetch all of the OUT data from memory in order to send it onto the USB bus. A host controller implementation could pre-fetch all of the OUT data, and pre-compute the actual number of bits in the token and data packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers can allow the host controller to know exactly whether there is enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the micro-frame. It is a reasonable assumption that software never over-commits the micro-frame to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction is not executed that could have been executed. However, under all circumstances, a transaction is never started unless there is enough time in the frame to complete the transaction.

54.4.3.4.1.1 Transaction Fit - A Best-Fit Approximation Algorithm

A curve is calculated which represents the latest start time for every packet size, at which software schedules the start of a periodic transaction.

This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the micro-frame. A plot of these two curves are illustrated in Figure 54-9. The plot Y-axis represents the number of byte-times left in a frame.

The space between the 80% and the Last Start plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and Last Start curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land above the function curve. The host controller will not start transactions whose results land below the function curve.

The following figure illustrates the Best-Fit Approximation.

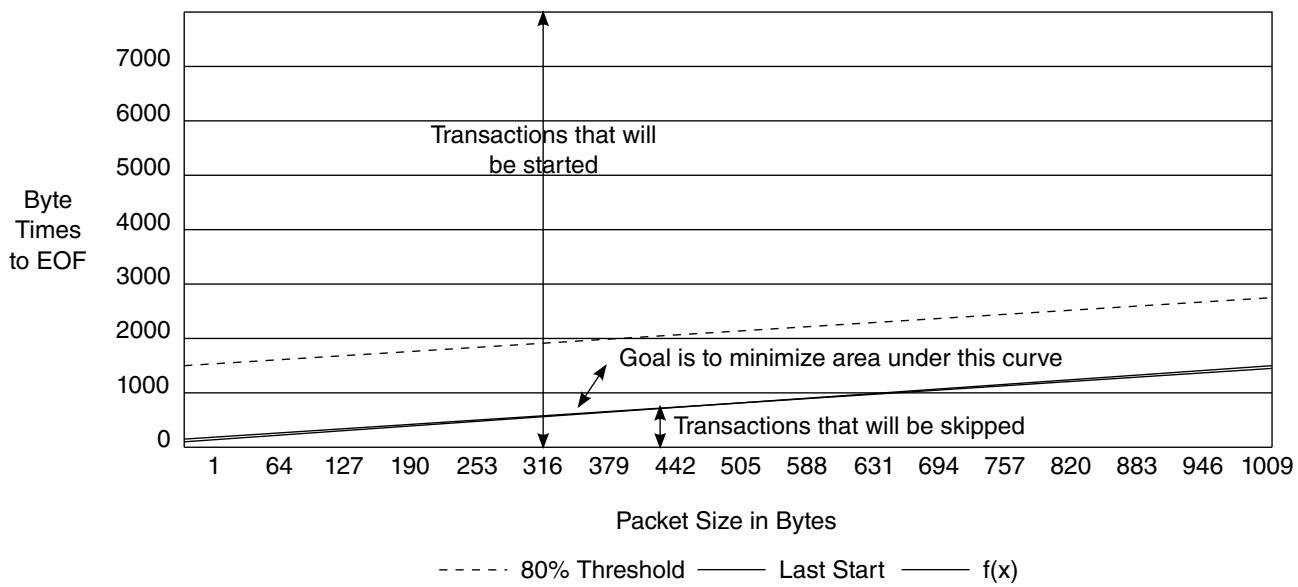


Figure 54-9. Best Fit Approximation

The LastStart line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used is a start-split, zero-length OUT transaction with a handshake. Summaries of the component parts are listed in the table below. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 54-38. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation

Table continues on the next page...

Table 54-38. Example Worse-case Transaction Timing Components (continued)

Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Token as defined in USB core specification. Includessync, token, eop, and so on.
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop, and so on.
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop, and so on.
		144	Total

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the Last Start curve, without dipping below the LastStart line, while at the same time keeping the check as simple as possible for hardware implementation. The f(x) in Figure 54-9 was constructed using the following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Alorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)
Begin
Local Temp = MaximumPacketSize + 192
Local rvalue = TRUE
If MaximumPacketSize >= 128 then
     Temp += 128
End If
If Temp > HC_BytesLeftInFrame then
     Rvalue = FALSE
End If
Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and the hardware counter of the number of bytes left in the current micro-frame. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the $f(x)$ plot was getting close to the LastStart line.

54.4.3.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned.

Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions through a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in the following figure). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

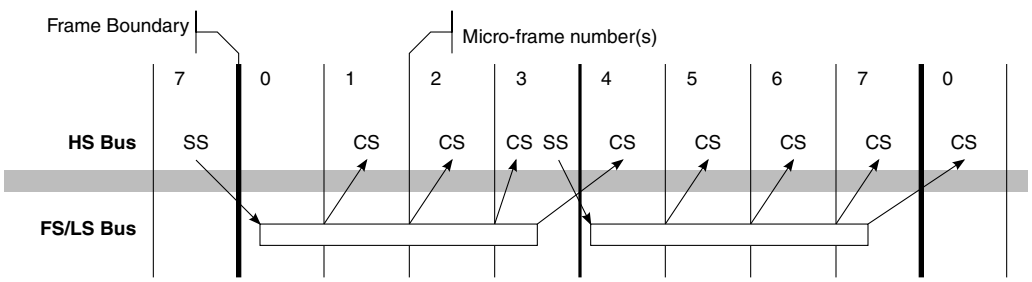


Figure 54-10. Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as the above figure illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed through the Frame List Index Register (USB_FRINDEX) documented in USB Frame Index (USB_*n*FRINDEX) and initially illustrated in Schedule Traversal Rules . Bits FRINDEX[2:0], represent the micro-frame number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are increment based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one micro-frame. The one micro-frame delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in the following figure. This adjustment allows software to trivially schedule the periodic start and

complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this specification.

The following figure illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined: The host controller's view of the 1 msec boundaries is called H-Frames. The high-speed bus's view of the 1 msec boundaries is called B-Frames.

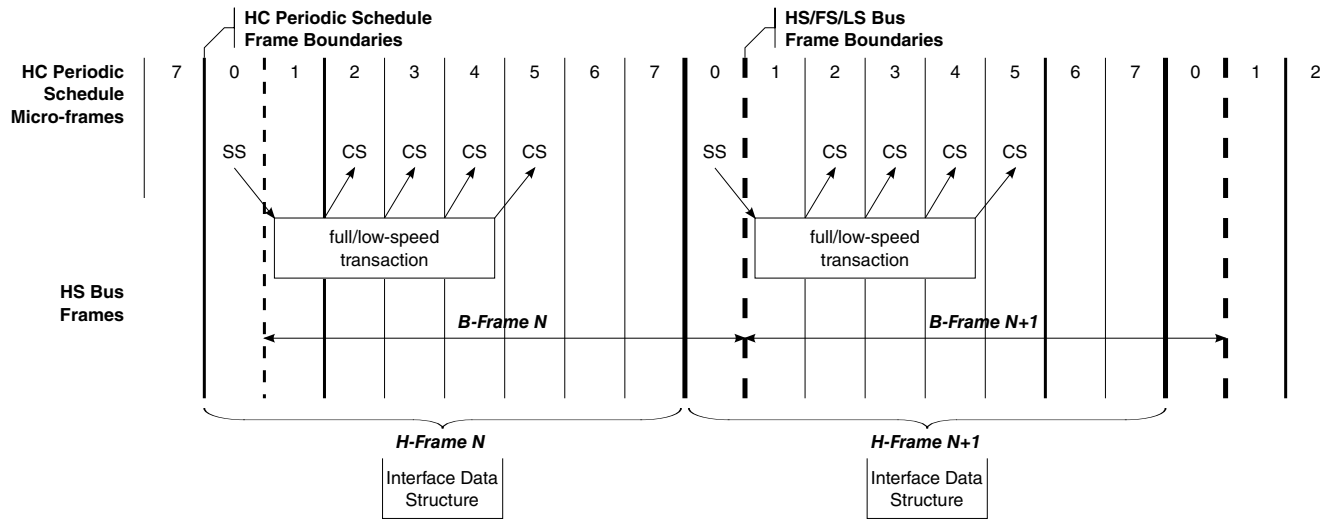


Figure 54-11. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Micro-frame numbers for the H-Frame are tracked by FRINDEX[2:0]. B-Frame boundaries are visible on the high-speed bus through changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (that is the high-speed bus sees eight SOFs with the same frame number value). H-Frames and B-Frames have the fixed relationship (that is B-Frames lag H-Frames by one micro-frame time) illustrated in the figure above. The host controller's periodic schedule is naturally aligned to H-Frames. Software schedules transactions for full- and low-speed periodic endpoints relative the H-Frames. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in USB Frame Index (USB_*n*FRINDEX), the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one micro-frame count. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. USB Frame Index (USB_*n*FRINDEX) provides the requirements that software should adhere when writing a new value in FRINDEX.

The table below illustrates the required relationship between the value of FRINDEX and the value of SOFV.

Table 54-39. Operation of FRINDEX and SOFV (SOF Value Register)

Current			Next		
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

NOTE

Where [F] = [13:3]; [μF] = [2:0]

54.4.3.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled through the Periodic Schedule Enable bit in the USB_USBCMD register.

If the Periodic Schedule Enable bit is set to zero, then the host controller simply does not try to access the periodic frame list through the USB_PERIODICLISTBASE register. Likewise, when the Periodic Schedule Enable bit is one, then the host controller does use the USB_PERIODICLISTBASE register to traverse the periodic schedule. The host controller will not react to modifications to the Periodic Schedule Enable immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the Periodic Schedule Enable bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b micro-frame. These work items must be removed from the schedule before the Periodic Schedule Enable bit is written to zero. The Periodic Schedule Status bit in the USB_USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing one (or zero) to the Periodic Schedule Enable bit in the USB_USBCMD register. Software then can poll the Periodic Schedule Status bit to determine when the periodic schedule has

made the desired transition. Software must not modify the Periodic Schedule Enable bit unless the value of the Periodic Schedule Enable bit equals that of the Periodic Schedule Status bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the

The following figure illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

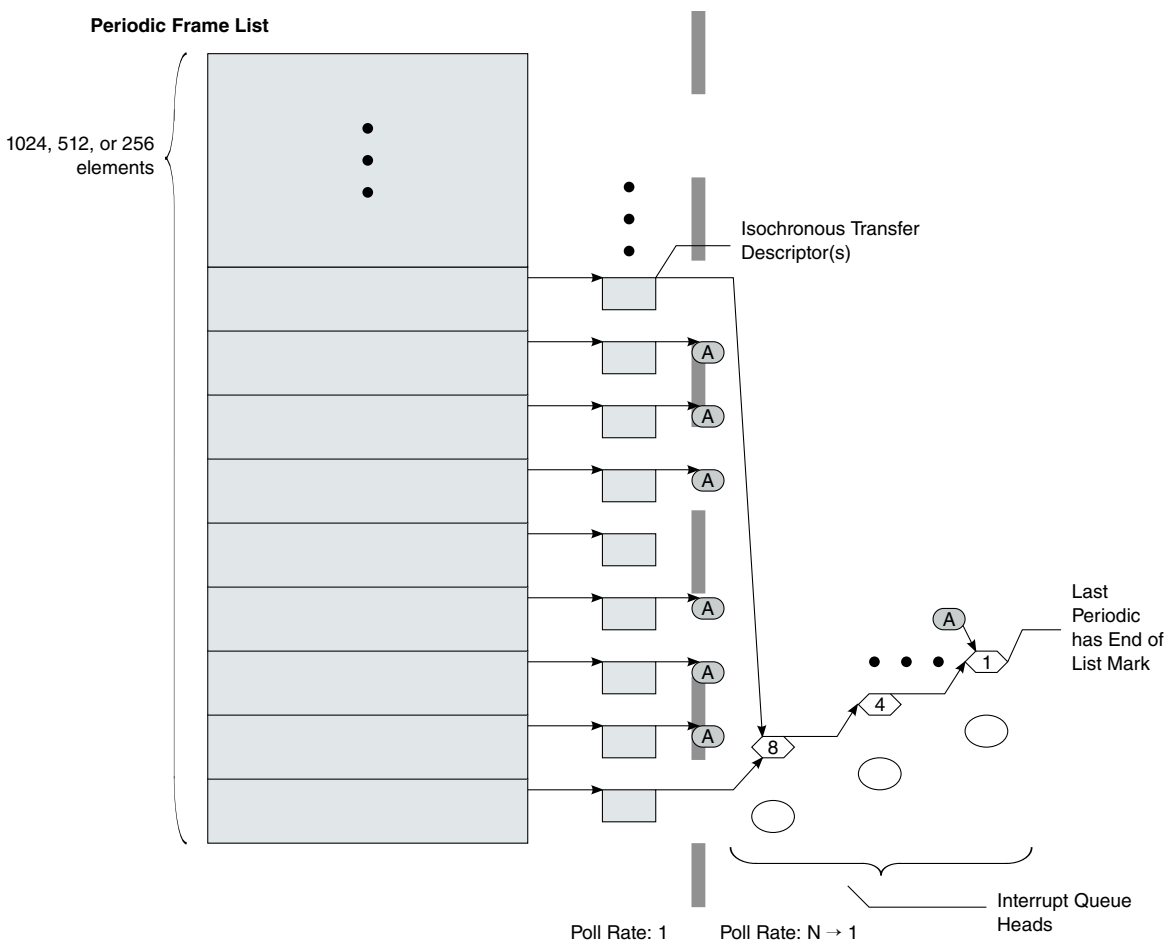


Figure 54-12. Example Periodic Schedule

54.4.3.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). The four distinct sections to an iTD:

- The first field is the Next Link Pointer. This field is for schedule linkage purposes only.
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4 K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

54.4.3.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Therefore, each transaction descriptor corresponds to one micro-frame. Each iTD can span 8 micro-frames worth of transactions.

When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array.

If the active bit in the Status field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the Next pointer to the next schedule data structure.

When the indexed active bit is one, the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, and so on.). It also uses the Page Select (PG) field to index the buffer pointer array, storing the selected buffer pointer and the next sequential buffer pointer. For example, if PG field is 0, then the host controller stores Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's PG field) and the transaction description's Transaction Offset field. The host controller uses the endpoint addressing information and I/O-bit to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the Status field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (for example, page 0 pointer) selected by the active transaction descriptions' PG (for example, value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer crosses a page boundary. When this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (for example, page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the Maximum Packet Size field. An iTD supports high-bandwidth pipes through the Mult (multiplier) field. When the Mult field is 1, 2, or 3, the host controller executes the specified number of Maximum Packet sized bus transactions for the endpoint in the current micro-frame. In other words, the Mult field represents a transaction count for the endpoint in the current micro-frame. If the Mult field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the Mult field.

For OUT transfers, the value of the Transaction X Length field represents the total bytes to be sent during the micro-frame. The Mult field must be set by software to be consistent with Transaction X Length and Maximum Packet Sixe. The host controller sends the bytes in Maximum Packet Size'd portions. After each transaction, the host controller decrements its local copy of Transaction X Length by Maximum Packet Size. The number of bytes the host controller sends is always Maximum Packet Size or Transaction X Length, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues Mult transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use Maximum Packet Size to detect packet babble errors. The host controller keeps the sum of bytes received in the Transaction X Length field. After all transactions for the endpoint have completed for the micro-frame, Transaction X Length contains the total bytes received. If the final value of Transaction X Length is less than the value of Maximum Packet Size, then less data than was allowed for was received from the associated endpoint. This short packet condition does not set

the USBINT bit in the USB_USBSTS register to one. The host controller will not detect this condition. If the device sends more than Transaction X Length or Maximum Packet Size bytes (whichever is less), then the host controller sets the Babble Detected bit to one and set the Active bit to zero. Note, that the host controller is not required to update the iTD field Transaction X Length in this error scenario. If the Mult field is greater than one, then the host controller automatically executes the value of Mult transactions. The host controller will not execute all Mult transactions if:

- The endpoint is an OUT and Transaction X Length goes to zero before all the Mult transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before Mult transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame. Refer to Appendix D for a table summary of the host controller required behavior for all the high-bandwidth transaction cases.

54.4.3.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

The following figure illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

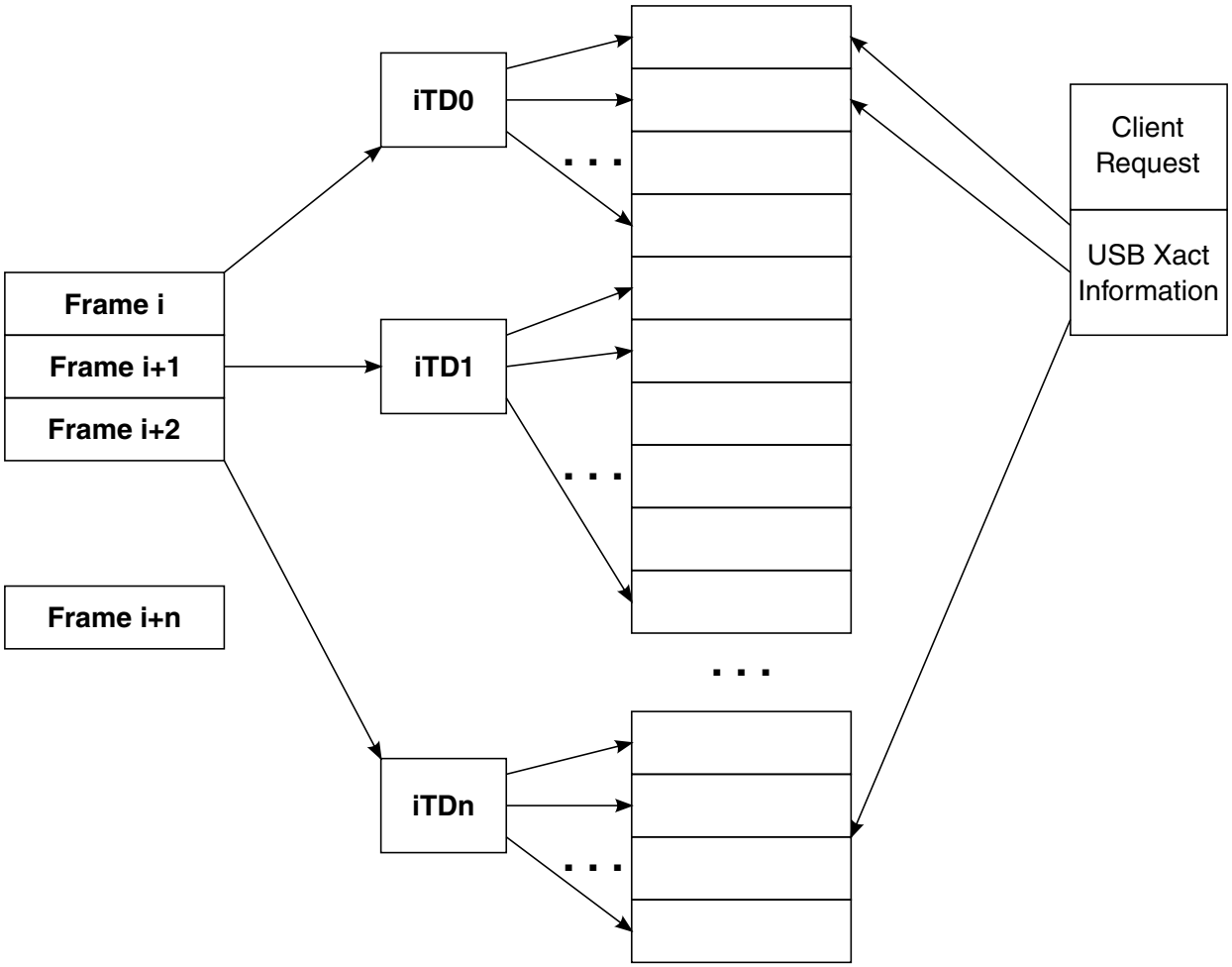


Figure 54-13. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the Transaction Offset field is set relative to the correct buffer pointer page (for example, the same one referenced by the PG field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current Page Buffer Pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available Page Buffer Pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer wraps a page boundary. Doing so yields undefined behavior. The host controller hardware is not required to 'alias' the page selector to Page zero. USB 2.0 isochronous

endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to one.

54.4.3.7.2.1 Periodic scheduling threshold

The Isochronous Scheduling Threshold field in the USB_HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures.

It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. Three basic caching models that account for the fact the isochronous data structures span 8 micro-frames. The three caching models are: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the USB_FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty-factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the Isochronous Scheduling Threshold field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame) but always dumps any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the USB_FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the Isochronous Scheduling Threshold field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the USB_FRINDEX register (plus the constant 1

uncertainty) to determine the current micro-frame/frame (assume modulo 8 arithmetic in adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, then software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant 3 bits of the Isochronous Scheduling Threshold field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the Isochronous Scheduling Threshold field. For example, if the count value were 2, then the host controller keeps a window of 2 micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

54.4.3.8 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

If the Asynchronous Schedule Enable bit is set to zero, then the host controller simply does not try to access the asynchronous schedule through the USB_ASYNCLISTADDR register. Likewise, when the Asynchronous Schedule Enable bit is one, then the host controller does use the USB_ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to the Asynchronous Schedule Enable bit are not necessarily immediate. Rather the new value of the bit is taken into consideration the next time the host controller needs to use the value of the USB_ASYNCLISTADDR register to get the next queue head.

The Asynchronous Schedule Status bit in the USB_USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing one (or zero) to the Asynchronous Schedule Enable bit in the USB_USBCMD register. Software then can poll the Asynchronous Schedule Status bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the Asynchronous Schedule Enable bit unless the value of the Asynchronous Schedule Enable bit equals that of the Asynchronous Schedule Status bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the USB_ASYNCLISTADDR register. The default value of the USB_ASYNCLISTADDR register after reset is undefined and the schedule is disabled when the Asynchronous Schedule Enable bit is zero.

Software may only write this register with defined results when the schedule is disabled. For example, Asynchronous Schedule Enable bit in the USB_USBCMD and the Asynchronous Schedule Status bit in the USB_USBSTS register are zero. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the Asynchronous Schedule Enable bit to one. The asynchronous schedule is actually enabled when the Asynchronous Schedule Status bit is one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the USB_ASYNCLISTADDR register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller completes processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the USB_ASYNCLISTADDR register. Next time the asynchronous schedule is accessed, this is the first data structure that is serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller completes processing the asynchronous schedule when one of the following events occur:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition (see Empty Asynchronous Schedule Detection)
- The schedule has been disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 54-8. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

54.4.3.8.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section.

There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the USB_ASYNCLISTADDR register, then enables the list by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example, qTD pointers have T-Bits set to one or reference valid qTDs and the Horizontal Pointer references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadCurrent is a pointer to a queue head that is
          -- already in the active list
          -- pQHeadNew is a pointer to the queue head to be added
          --
          -- This algorithm links a new queue head into a existing
          -- list
          --
          pQueueHeadNew.HorizontalPointer  =  pQueueHeadCurrent.HorizontalPointer
          pQueueHeadCurrent.HorizontalPointer =  physicalAddressOf(pQueueHeadNew)
End InsertQueueHead
```

54.4.3.8.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section.

There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list.

Software deactivates the asynchronous schedule by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to zero. Software can determine when the list is idle when the Asynchronous Schedule Status bit in the USB_USBSTS register is zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list through the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadPrevious is a pointer to a queue head that
          -- references the queue head to remove
          -- pQHeadToUnlink is a pointer to the queue head to be
```

```
        -- removed
        -- pQheadNext is a pointer to a queue head still in the
        -- schedule. Software provides this pointer with the
        -- following strict rules:
        --          if the host software is one queue head, then
        --          pQHeadNext must be the same as
        --          QueueheadToUnlink.HorizontalPointer. If the host
        --          software is unlinking a consecutive series of
        --          queue heads, QHeadNext must be set by software to
        --          the  queue head remaining in the schedule.
        --
        -- This algorithm unlinks a queue head from a circular list
        --
        pQueueHeadPrevious.HorizontalPointer =  pQueueHeadToUnlink.HorizontalPointer
                          pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
End UnlinkQueueHead
```

If software removes the queue head with the H-bit set to one, it must select another queue head still linked into the schedule and set its H-bit to one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its H-bit set to one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, and so on). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (Interrupt on Async Advance Doorbell bit in the USB_USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (Interrupt on Async Advance bit in the USB_USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to one, it also sets the command bit to zero. The third bit is an interrupt enable (Interrupt on Async Advance bit in the USB_USBINTR register) that is matched with the status bit. If the status bit is one and the interrupt enable bit is one, then the host controller asserts a hardware interrupt.

The figure below illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that remains in the asynchronous schedule.

When the host controller observes that doorbell bit being set to one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A and B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (that is traversed beyond queue head (B) in this example). The following figure illustrates the generic queue head unlink scenario.

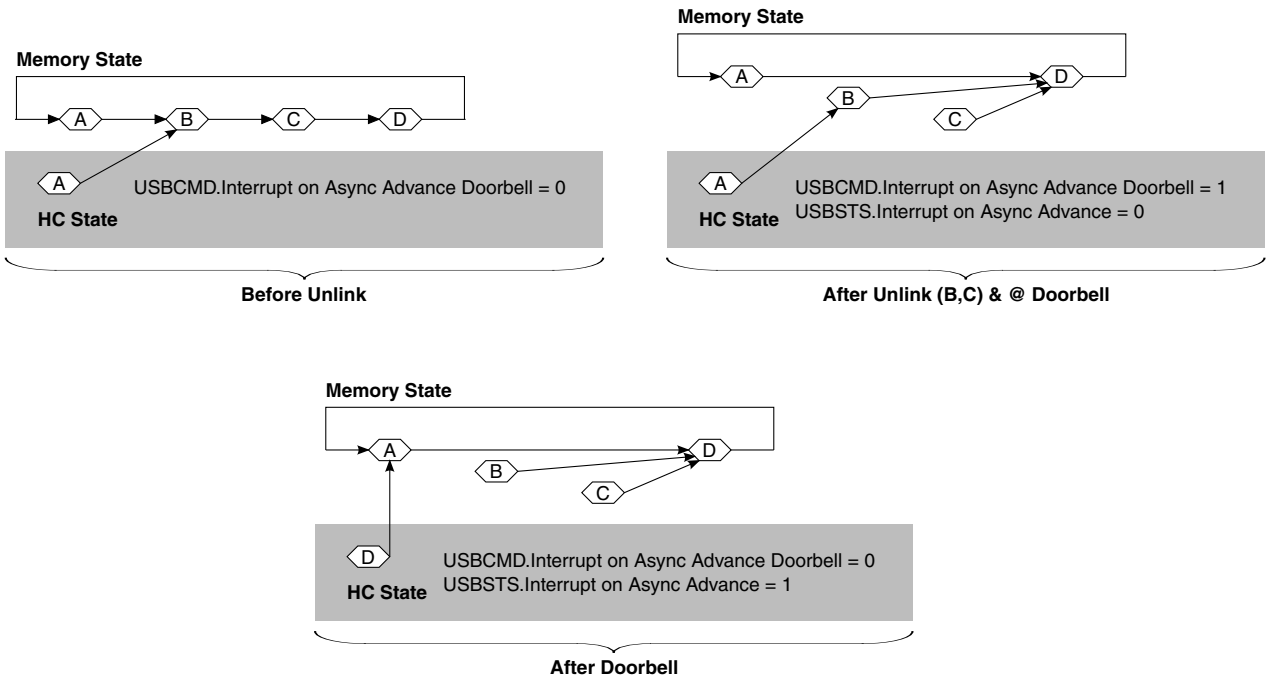


Figure 54-14. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue (twice)) before setting the Advance on Async status bit to one.

Software may re-use the memory associated with the removed queue heads after it observes the Interrupt on Async Advance status bit is set to one, following assertion of the doorbell. Software should acknowledge the Interrupt on Async Advance status as indicated in the USB_USBSTS register, before using the doorbell handshake again.

54.4.3.8.3 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty.

The queue head data structure (see Table 54-25) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USB_USBSTS register (*Reclamation*) that is set to zero when the Enhanced Interface Host Controller observes a queue head with the H-bit set to one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Asynchronous schedule traversal: *Start Event*).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is shown in the following figure.

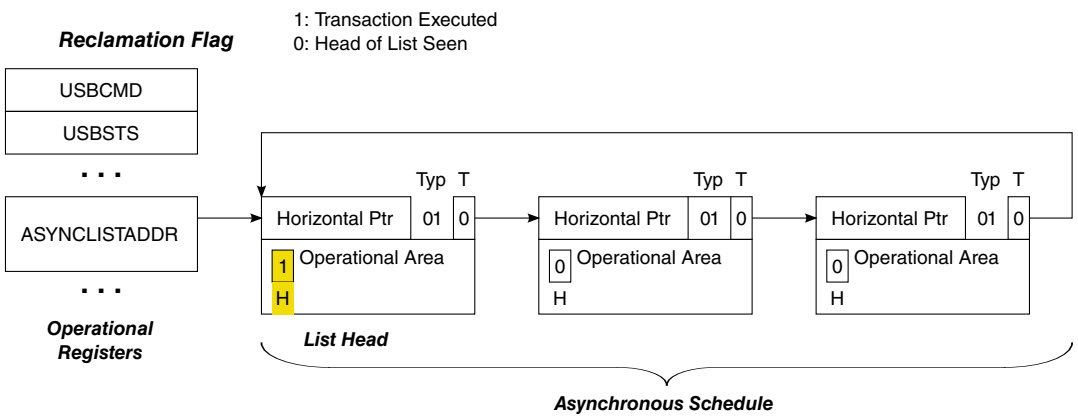


Figure 54-15. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to one, and that it is always coherent with respect to the schedule.

54.4.3.8.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the micro-frame.

It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out very quickly. If it is the only item in the schedule, then the host controller ceases traversal of the Asynchronous schedule very early in the micro-frame. In order to provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting

until the next micro-frame. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

54.4.3.8.4.1 Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is: *how long should the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10 µsec. The value is derived based on the analysis in Example Derivation for *AsyncSchedSleepTime*, The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *Nakcnt* fields, and so on. So the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. The figure below illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

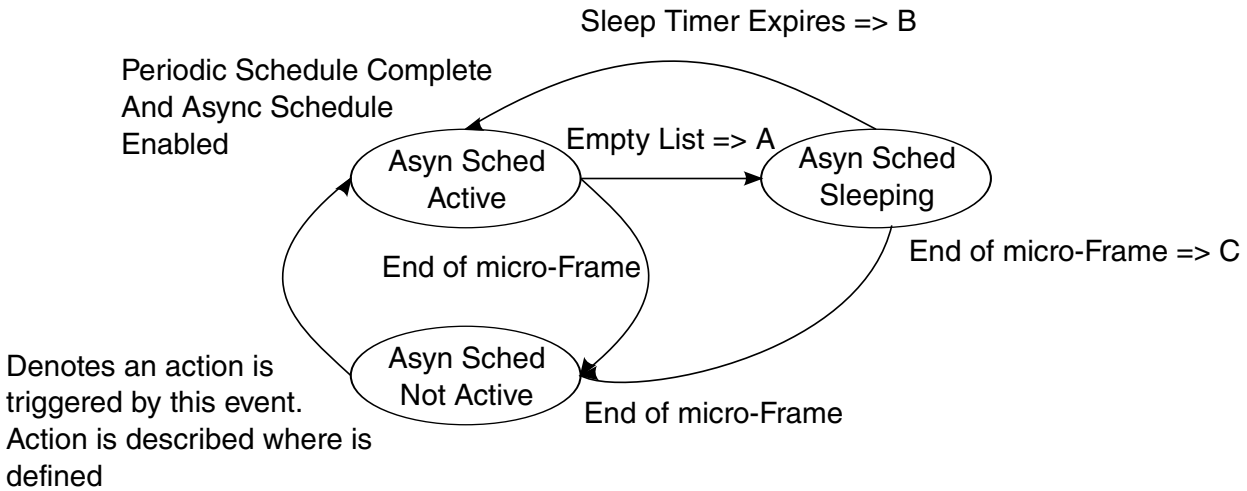


Figure 54-16. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in the figure above are defined in the following table.

Table 54-40. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to one and moves the Nak Counter reload state machine to WaitForListHead (see Nak Count Reload Control).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

54.4.3.8.4.2 Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine does not leave this state when the *Asynchronous Schedule Enable* bit in the USB_USBCMD register is zero.

This state is entered from Async Sched Active or Async Sched Sleeping states when the end-of-micro-frame event is detected.

54.4.3.8.4.3 Async Sched Active

This state is entered from the Async Sched Not Active state when the periodic schedule is not active. It is also entered from the Async Sched Sleeping states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USB_USBSTS register to one.

While in this state, the host controller continually traverses the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

54.4.3.8.4.4 Async Sched Sleeping

The state is entered from the Async Sched Active state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

54.4.3.8.4.5 Example Derivation for *AsyncSchedSleepTime*

The derivation is based on analysis of what work the host controller could be doing next.

It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests.

The table below summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (for example *footprint*, or *wall clock*) the transaction takes to complete.

Table 54-41. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk		
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64		
Type	Bulk		
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (that is setup)
Size	8		
Type	Cntrl		

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller gets the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10 µs sleep period would allow the host controller to get the NAK results on the first complete-split.

54.4.3.8.5 Asynchronous schedule traversal: *Start Event*

Once the HC has *idled* itself through the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each micro-frame.

In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame) the HC must occasionally *re-activate* during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Restarting Asynchronous Schedule Before EOF . Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Restarting Asynchronous Schedule Before EOF).

54.4.3.8.6 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (see Empty Asynchronous Schedule Detection) depends on the proper management of the *Reclamation* bit in the USB_USBSTS register.

The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (see Fetch Queue Head).

It is required that the host controller sets the *Reclamation* bit to one whenever an asynchronous schedule traversal *Start Event*, as documented inAsynchronous schedule traversal: *Start Event*, occurs. The *Reclamation* bit is also set to one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Execute Transaction). The host controller sets the *Reclamation* bit to zero whenever it finds a queue head with its *H-bit* set to one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to zero when executing from the periodic schedule.

54.4.3.9 Operational Model for Nak Counter

This section describes the operational model for the *NakCnt* field defined in a queue head.

See Queue Head Initialization for more information. Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control through the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced through the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (that is like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which does not complete until after the transaction on the classic bus completes.

The two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. The two operational modes associated with this counter:

- Not Used- This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- Nak Throttle Mode- This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of Nak or Nyet responses the host controller tolerates on each endpoint. In this mode, the HC decrements the *NakCnt* field based on the token/handshake criteria listed in the table below. The host controller must reload *NakCnt* when the endpoint successfully moves data (for example, policy to reward device for moving data).

The following table describes the NakCnt field adjustment rules.

Table 54-42. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake NAK	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1] Start
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

1. Recommended behavior on this response is to reload *NakCnt*

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller does not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Nak Count Reload Control .

NOTE

When all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller detects an empty Asynchronous Schedule and idle schedule traversal (see Empty Asynchronous Schedule Detection).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Asynchronous schedule traversal: *Start Event*. Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

54.4.3.9.1 Nak Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Execute Transaction). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters (*NakCnt* see Table 54-25) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Asynchronous schedule traversal: *Start Event* for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 54-15).

The following figure illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: Execute Transaction (see the figure below). The host controller does not perform the nak counter reload operation if the RL field (see Table 54-25) is set to zero.

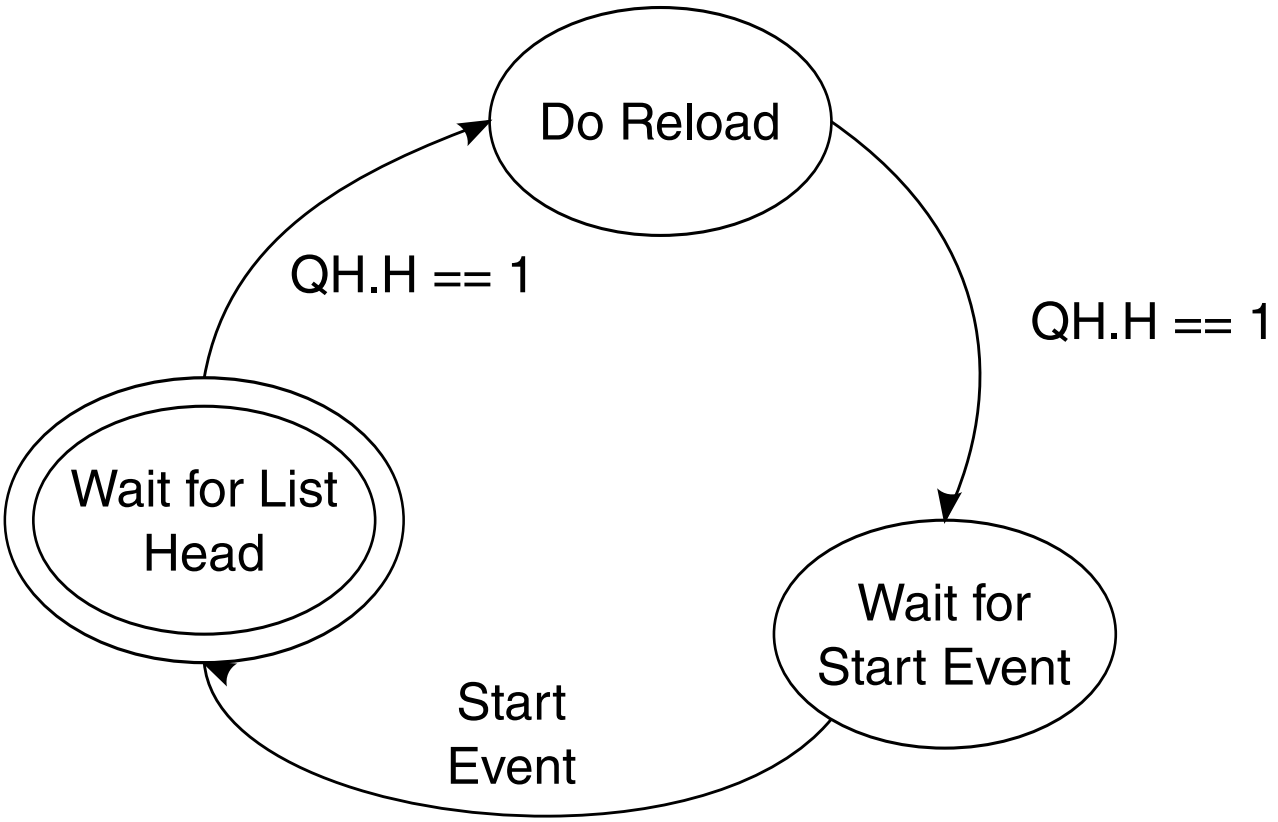


Figure 54-17. Example HC State Machine for Controlling Nak Counter Reloads

54.4.3.9.1.1 Wait for List Head

This is the initial state.

The state machine enters this state from Wait for Start Event when a start event as defined in Asynchronous schedule traversal: *Start Event*occurs.

The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule.

This occurs when the host controller fetches a queue head whose *H-bit* is set to one.

54.4.3.9.1.2 Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the *H-bit* set to one. While in this state, the host controller performs nak counter reloads for every queue head visited that has a non-zero nak reload value (*RL*) field.

54.4.3.9.1.3 Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to one is fetched. While in this state, the host controller does not perform nak counter reloads.

54.4.3.10 Managing Control/Bulk/Interrupt Transfers through Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure. One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Table 54-25). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer*.

The general processing model for the host controller's use of a queue head is simple:

- read a queue head,
- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area,
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller sets one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (for example, the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions occurs for the endpoint and the host controller does not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in the following figure. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller always *find* them if/when they are reachable. The figure below illustrates the Host Controller Queue Head Traversal State Machine.

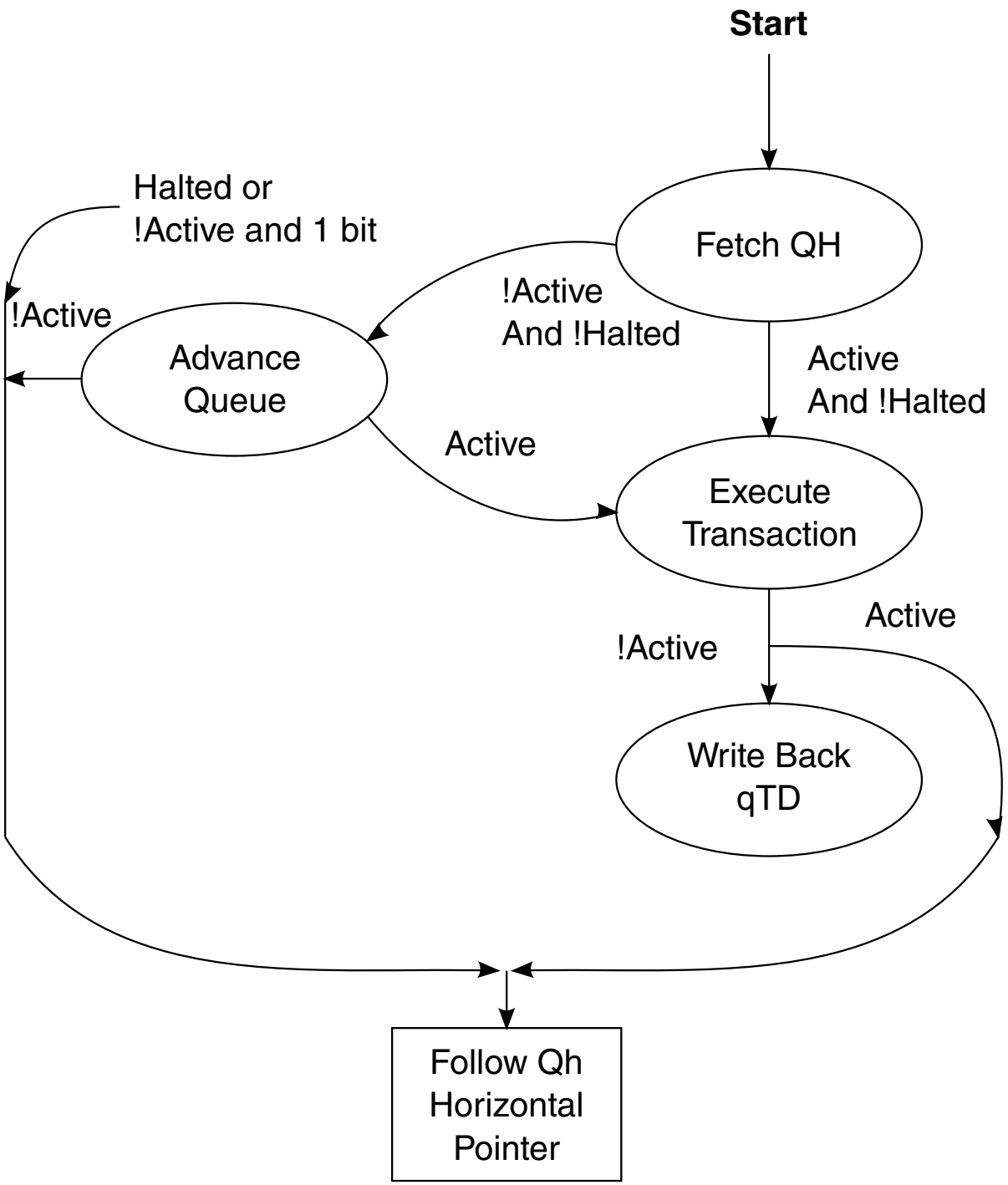


Figure 54-18. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (see Execute

Transaction) describes the basic requirements for all endpoints. Split Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions.

NOTE

Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section Queue Head for layout of a queue head):

Valid static endpoint state.

- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

54.4.3.10.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (see Next Asynch. Address (USB_*n*ASYNCLISTADDR))/ Endpoint List Address (USB_*n*ENDPTLISTADDR) Additionally, it may be referenced from the *Next LinkPointer* field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Table 54-25.

While in this state, the host controller performs operations to implement empty schedule detection (see Empty Asynchronous Schedule Detection) and Nak Counter reloads (see Operational Model for Nak Counter). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (that is *S-mask* is zero), and
- The *H-bit* is one, and
- The *Reclamation* bit in the USBSTS register is zero.

When these criteria are met, the host controller stops traversing the asynchronous list (as described in Empty Asynchronous Schedule Detection). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is one and the *Reclamation* bit is one, then the host controller sets the *Reclamation* bit in the USBSTS register to zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Operational Model for Nak Counter.

This state is complete when the queue head has been read on-chip.

54.4.3.10.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the FetchQHD state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay.

NOTE

If the *I-bit* is one and the *Active* bit is zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer*. Otherwise, the host controller uses the *NextqTD Pointer*. If *NextqTD Pointer's T-bit* is set to one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its *Active* bit set to one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure.

The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 54-27).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

54.4.3.10.3 Execute Transaction

The host controller enters this state from the Fetch Queue Head state only if the *Active* bit in *Status* field of the queue head is set to one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Split Transactions for Asynchronous Transfers and Split Transaction Interrupt .

54.4.3.10.3.1 Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (for example, non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has one in it.

For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

54.4.3.10.3.2 Asynchronous Transfer Pre-operations and Pre-condition Criteria

If the queue head is not for an interrupt endpoint (for example, zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria.

The pre-operation is:

Checks the Nak counter reload state (Operational Model for Nak Counter). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The pre-condition evaluated is:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

54.4.3.10.3.3 Transfer Type Independent Pre-operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are:

- The host controller determines whether there is enough time in the micro-frame to complete this transaction (see Transaction Fit - A Best-Fit Approximation Algorithm for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller exits this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to zero, or
- There is not enough time in the micro-frame left to execute the next transaction(see Transaction Fit - A Best-Fit Approximation Algorithm) for example method for implementing the frame boundary test).

NOTE

For a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (for example,

advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Buffer Pointer List Use for Data Streaming with qTDs .

NOTE

The *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller executes zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and the device delivers data, the host controller detects a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (for example not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (for example, decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory).

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *MaximumPacket Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Transaction Error .

The following events causes the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from one to zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to one and *Active* is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to one and the *Active* bit is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The *Total Bytes to Transfer* field is zero after the transaction completes.
  - For a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the Advance Queue state. Refer to Split Transactions for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (for example *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (for example a packet babble). This results in the host controller setting the *Halted* bit to one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

54.4.3.10.3.4 Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed through queue heads (control, bulk and interrupt).

The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USB_n_USBSTS register to one. To halt the queue head, the *Active* bit is set to zero and the *Halted* bit is set to one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller does not advance the transfer state on a transaction that results in a *Halt* condition (for example no updates necessary for *Total Bytes to Transfer*, *C_Page*, *Current Offset,* and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USB_n_USBSTS register is set to one. If the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is set to one, a hardware interrupt is generated at the next interrupt threshold.

54.4.3.10.3.5 Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule.

This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the USB_n_HCCPARAMs register to one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USB_n_USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (for example *Asynchronous Schedule Park Mode Enable* bit in the USB_n_USBCMD register is zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (for example only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the *Asynchronous Schedule Park Mode Count* field in the USB_n_USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to one and also set *Asynchronous Schedule Park Mode Count* field to zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Execute Transaction . It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake.

The following table summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 54-43. Actions for Park Mode, based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1, 2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.

Table continues on the next page...

Table 54-43. Actions for Park Mode, based on Endpoint Response and Residual Transfer State (continued)

	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction.[2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

1. The host controller may continue to execute bus transactions from the current high-speed queue head (if *PM-Count* is not equal to zero), if a PID mismatch is detected (for example expected DATA1 and received DATA0, or visa-versa).
2. This specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

54.4.3.10.4 Write Back qTD

This state is entered from the Execute Transaction state when the *Active* bit is set to zero.

The source data for the write-back is the transfer results area of the queue head overlay area (see Table 54-43).

The host controller uses the *Current qTD Pointer* field as the target address for the qTD.

The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer*, *Cerr*, and *Status*.

The duration of this state depends on when the qTD write-back is committed.

54.4.3.10.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is one on exit from the Execute Transaction state, or
- When the host controller exits the Write Back qTD state, or
- If the Advance Queue state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is one on exit from the Fetch QH state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

54.4.3.10.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*.

This means: if the buffer spans more than one physical page, it must obey the following rules (the figure below illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4 K chunk beyond the first page, each buffer portion matches to a full 4 K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20 K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16 Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction spans a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

The following figure illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page for example 2049 (for example 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4 K page.

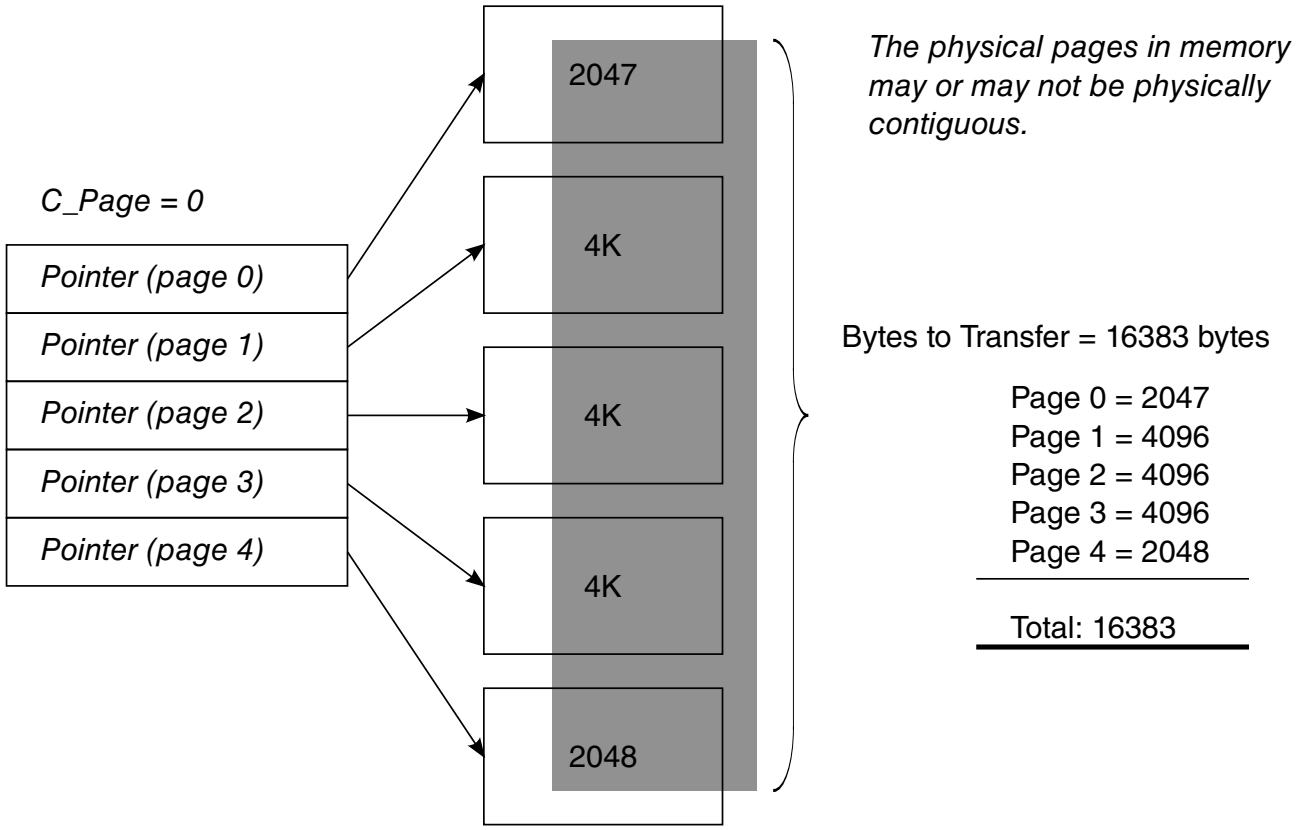


Figure 54-19. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller increments *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is *C_Page*) when necessary. The three conditions for how the host controller handles *C_Page*:

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.

- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (that is the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

NOTE

The only valid adjustment the host controller may make to *C_Page* is to increment by one.

54.4.3.10.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate.

System software sets a bit in a queue head's *S-Mask* to indicate which micro-frame within 1 msec period a transaction should be executed for the queue head. Software must ensure that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in the following table.

Table 54-44. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, and so on *S-Mask* = 01h	A queue head for the *bInterval* of 2 msec (16 micro-frames) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8, and so on *S-Mask* = 02h	Another example of a queue head with a *bInterval* of 2 msec is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

54.4.3.10.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller sets an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to one, or whenever a transfer (qTD) completes with a short packet.

If system software needs multiple qTDs to complete a client request (that is like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

54.4.3.11 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints.

Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it uses in the next transaction to the endpoint (see the table below).

The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

The following table illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 54-45. Ping Control State Transition Table

Event			
Current	Host	Device	Next
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr [1]	Do Ping
Do Ping	PING	Stall	N/C [2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping

Table continues on the next page...

Table 54-45. Ping Control State Transition Table (continued)

Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C [2]

1. Transaction Error (XactErr) is any time the host misses the handshake.
2. No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (for example Active set to zero and Halt set to one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to Do Ping. The Ping State bit has the following encoding:

Table 54-46. Ping State bit Encoding

Value	Meaning
0B	Do OUT The host controller uses an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller uses a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (that is start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

54.4.3.12 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs.

This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol.

Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Split Transaction Isochronous Transfer Descriptor (siTD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Queue Head). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low-/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

54.4.3.12.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an *EPS* field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head.

All full-speed bulk and full-, low-speed control are managed through queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to Do-Start-Split. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.

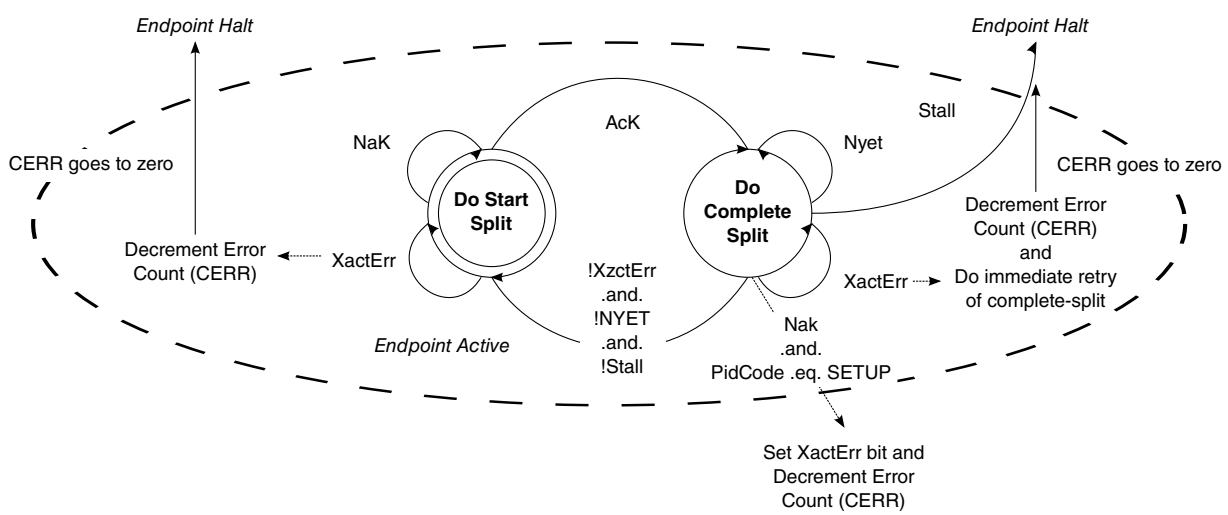


Figure 54-20. Host Controller Asynchronous Schedule Split-Transaction State Machine

54.4.3.12.1.1 Asynchronous - Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the Do Complete Split state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller executes a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or OUT transaction, then the host controller reloads the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller does not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

54.4.3.12.1.2 Asynchronous - Do Complete Split

This state is entered from the Do Start Split state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller executes a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller reloads the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr*.

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, and so on. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to

accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule is the retry for this endpoint.

If *Cerr* went to zero, the host controller must halt the queue.

- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.

If the *PidCode* indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller advances the state of the transfer, for example move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Managing Control/Bulk/Interrupt Transfers through Queue Heads).

54.4.3.12.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed through the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule.

Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller visits a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (that is takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, and so on, occurs as defined in Managing Control/Bulk/Interrupt Transfers through Queue Heads, but only occurs on the completion of a split transaction.

54.4.3.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field.

The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint occurs. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost.

The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and ^{C}X labels indicate micro-frames where software can schedule start-splits and complete splits (respectively).

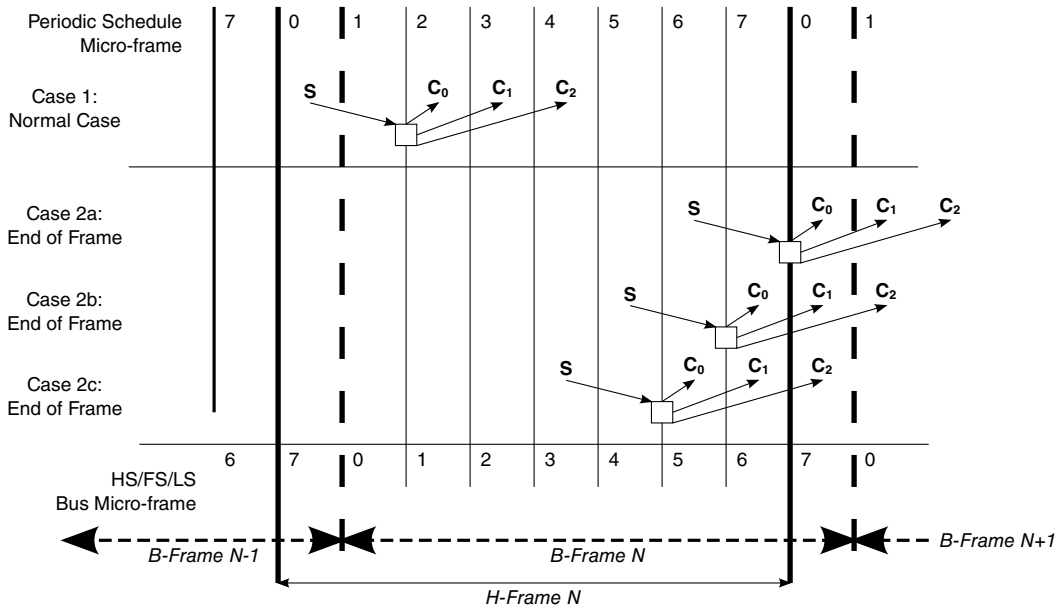


Figure 54-21. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in micro-frame 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs.

The figure below illustrates the general layout of the periodic schedule.

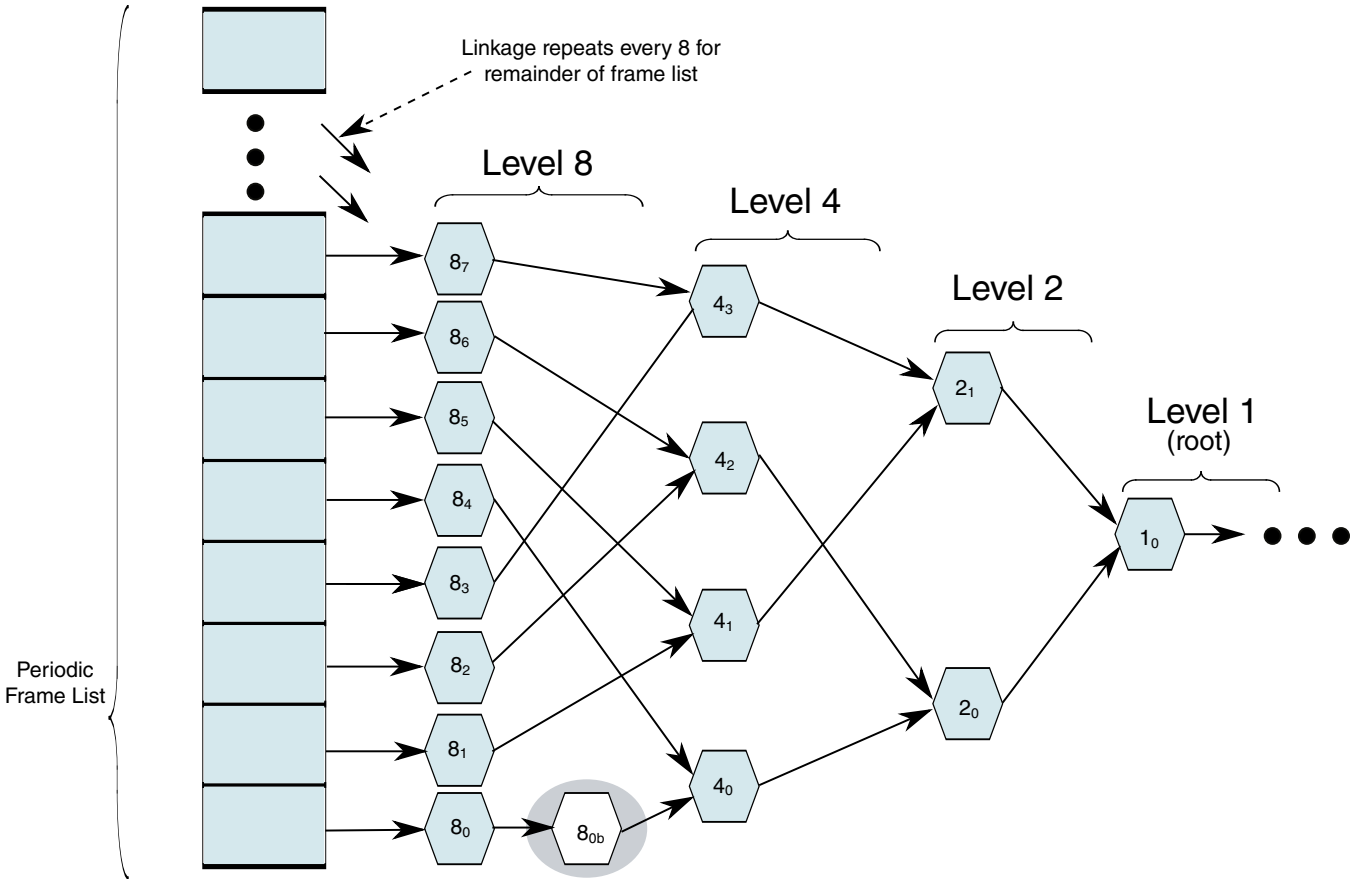


Figure 54-22. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol:

- *SplitXState*. This is single bit residing in the *Status* field of a queue head (see Table 54-23). This bit is used to track the current state of the split transaction.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 54-21, case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Start, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 54-21, case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Complete, and the current micro-frame as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

54.4.3.12.2.2 Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (that is boundary cases 2a through 2c).

An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see siTD Back Link Pointer).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

The four components to the use of FSTNs:

- FSTN data structure.
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to zero.

- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to one.
- Host controller FSTN traversal rules.

When the host controller encounters an FSTN during micro-frames 2 through 7 it simply follows the node's *Normal Path Link Pointer* to access the next schedule data structure.

NOTE

The FSTN's *Normal Path Link Pointer.T-bit* may set to one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in micro-frames 0 or 1, it saves the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller's rules for how it traverses the schedule and limits which data structures is considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the micro-frame (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN's *Back Path Link Pointer*.
- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer*.
- Do not process a QH (Queue Head) whose *EPS* field indicates a high-speed device. Simply follow its *Horizontal Link Pointer*.
- When a QH's *EPS* field indicates a Full/Low-speed device, the host controller considers only it for execution if its *SplitXState* is DoComplete (note: this applies whether the *PID Code* indicates an IN or an OUT). See Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction.
  - The host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Periodic Isochronous - Do Complete Split for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The

host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer*.

- If the host controller determines that there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive micro-frame, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in the following figure.

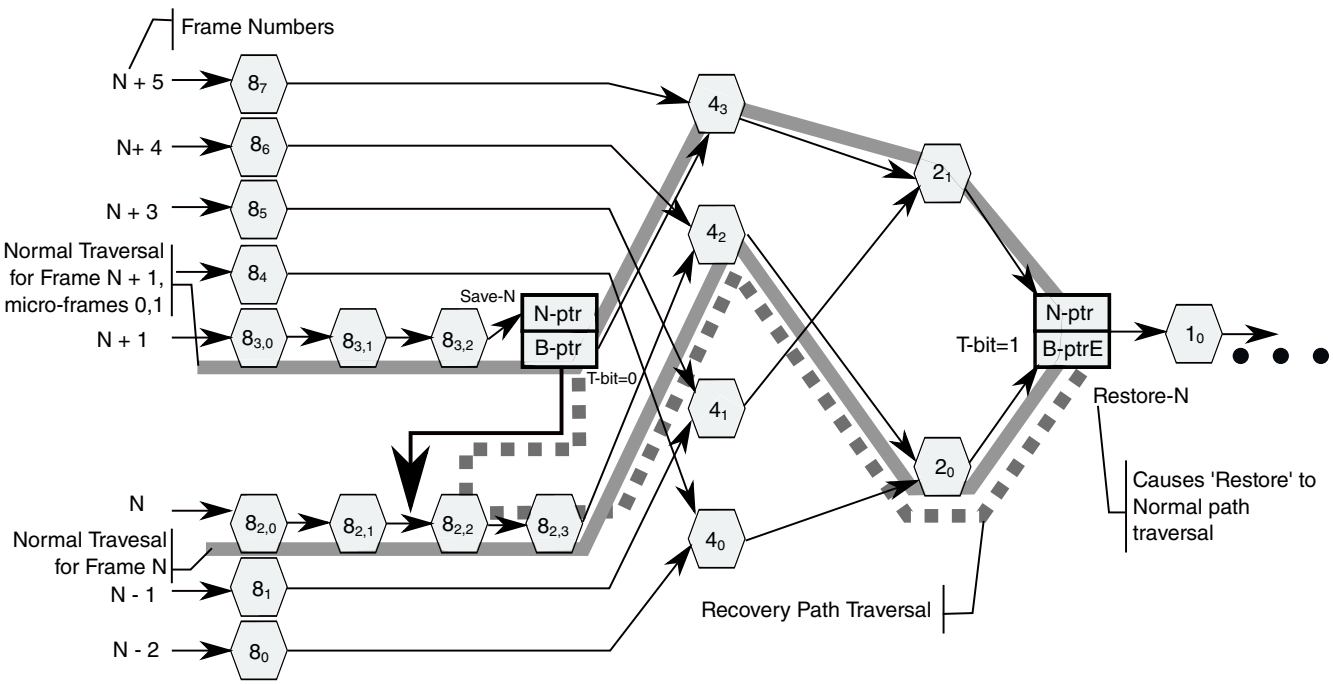


Figure 54-23. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N+1 (micro-frames 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in the figure above with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (for example Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, $8_{2.2}$, $8_{2.3}$, 4_2,

2_0, Restore-N, 4_3, 2_1, Restore-N, 1_0 …}. The nodes on the recovery-path are in bold. In frame N (micro-frames 0-7), for this example, the host controller traverses all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (that is normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 …}.

In frame N+1 (micro-frames 2-7), when the host controller encounters Save-Path FSTN Save-N, it unconditionally follows Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 …}.

54.4.3.12.2.3 Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse.

When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero.
    - *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list in order to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to one, as this is used to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list

location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there is times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth re-balance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

54.4.3.12.2.4 Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost.

For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline.

When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a

complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

54.4.3.12.2.5 Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

>As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence.

Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit*. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the *FRINDEX* register (that is, *cMicroFrameBit* = (1 shifted-left(*FRINDEX*[2:0])). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then *cMicroFrameBit* will equal 00000001b. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current micro-frame.

The following figure illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at Do_Start and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (for example, after the host tries the same transaction three times, and each

encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

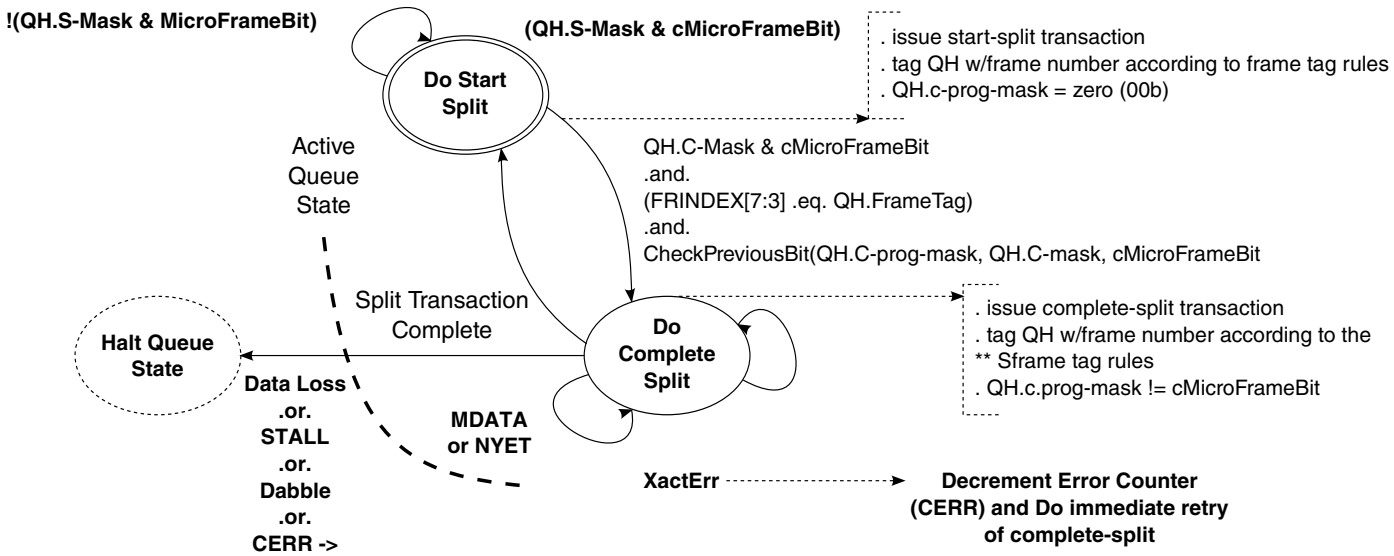


Figure 54-24. Split Transaction State Machine for Interrupt

See Previous Section for the frame tag management rules.

Periodic Interrupt - Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section Periodic Isochronous - Do Complete Split for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field (see Section), set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

Periodic Interrupt - Do Complete Split

This state is entered unconditionally from the Do Start Split state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. *cMicroFrameBit* is bit-wise anded with *QH.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame.
- Test B. *QH.FrameTag* is compared with the current contents of *FRINDEX*[7:3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
Begin
-- Return values:
-- TRUE - no error
-- FALSE - error
--
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                                If not(previousBit bitAND QH.C-prog-mask) then
                      rvalue = FALSE;
          End if
End If
-- If the C-prog-mask already has a one in this bit position,
```

```
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check to see if a start-split should be executed in this micro-frame. Note this is the same test performed in the Do Start Split state (see Section Periodic Isochronous - Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number (see Section). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the Last complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.

- Transaction Error (XactErr). Timeout, data CRC failure, etc. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN, or *CErr* is decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (that is, return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full- and low-speed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.

- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. The table below lists the possible combinations and the appropriate action.

Table 54-47. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	*QH.FrameTag* test failed. This means that exactly one or more *H-Frames* have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one. Note: When executing in the context of a *Recovery Path* mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the

Table 54-47. Interrupt IN/OUT Do Complete Split State Execution Criteria

		normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a *Recovery Path* mode.

Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is 6 *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different *H-Frames* (case 2a, see Figure 54-21).
- Rule 2: If the current value of *FRINDEX*[2:0] is 7, *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c (Figure 54-21).
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is not 6, or currently in Do Complete Split and the current value of (*FRINDEX*[2:0]) is not 7, *FrameTag* is set to *FRINDEX*[7:3]. This accommodates all other cases (Figure 54-21).

54.4.3.12.2.6 Rebalancing the periodic schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation.

This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (that is, new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is DoStart (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is

not required to write the transfer state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed its final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Because the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped via the *I-bit*.

1. Set the *Halted* bit to a one, then
2. Set the *I-bit* to a zero, then
3. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

54.4.3.12.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions.

This data structure uses the scheduling model of isochronous TDs (iTD, Section Isochronous (High-Speed) Transfer Descriptor (iTD)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

54.4.3.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur.

The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{S}X and ^{C}X labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

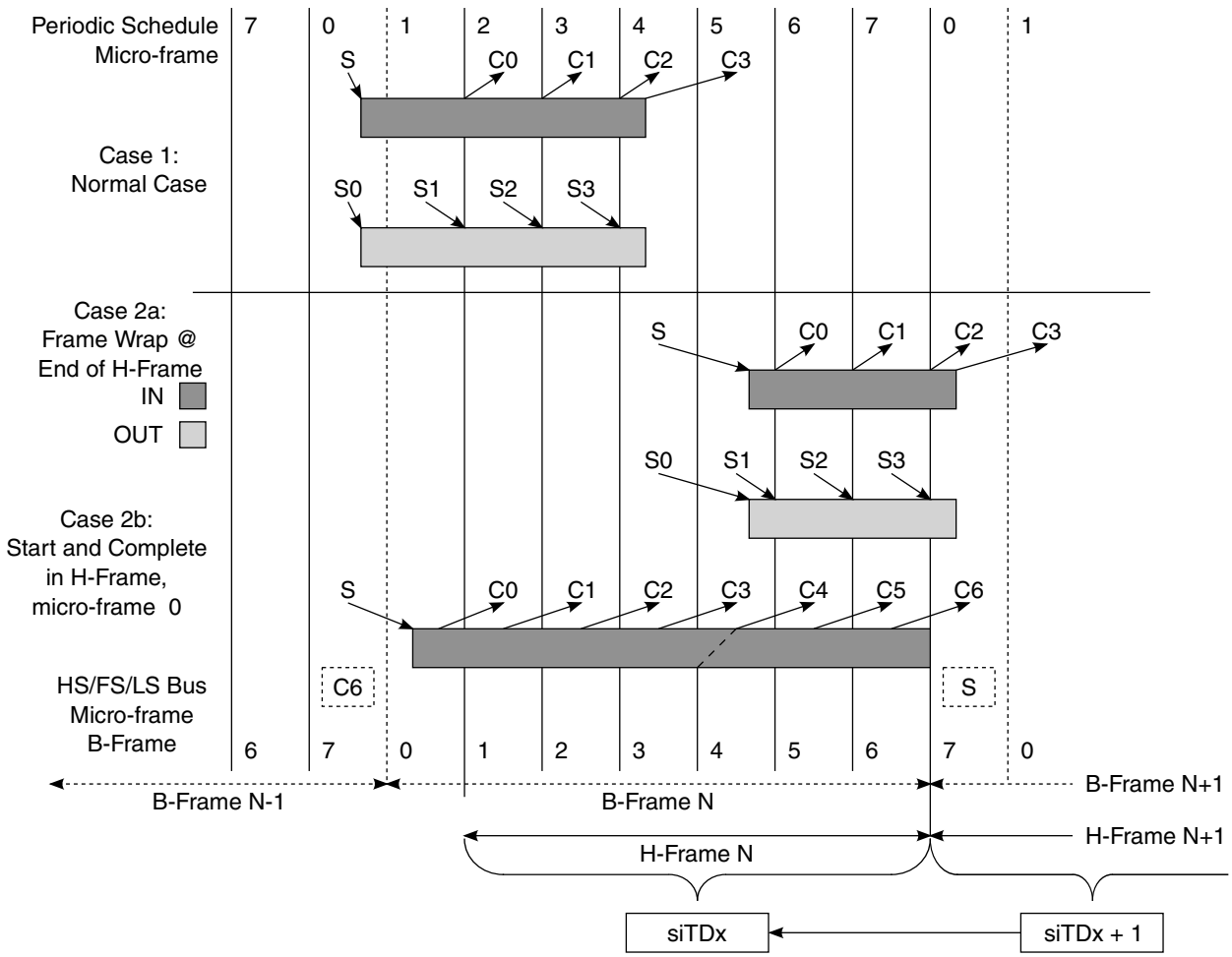


Figure 54-25. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame*. For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.

- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame,* micro-frames 6 or 7 (*H-Frame* micro-frame 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(For example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction.)
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD (see Figure 54-26). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section Split Transaction Execution State Machine for Interrupt .
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 54-25, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do Start Split, and the current micro-frame as indicated by USB_n_FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 54-25, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do

Complete Split, and the current micro-frame as indicated by *USB_n_FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.

- *Back Pointer*. This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. The figure below illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

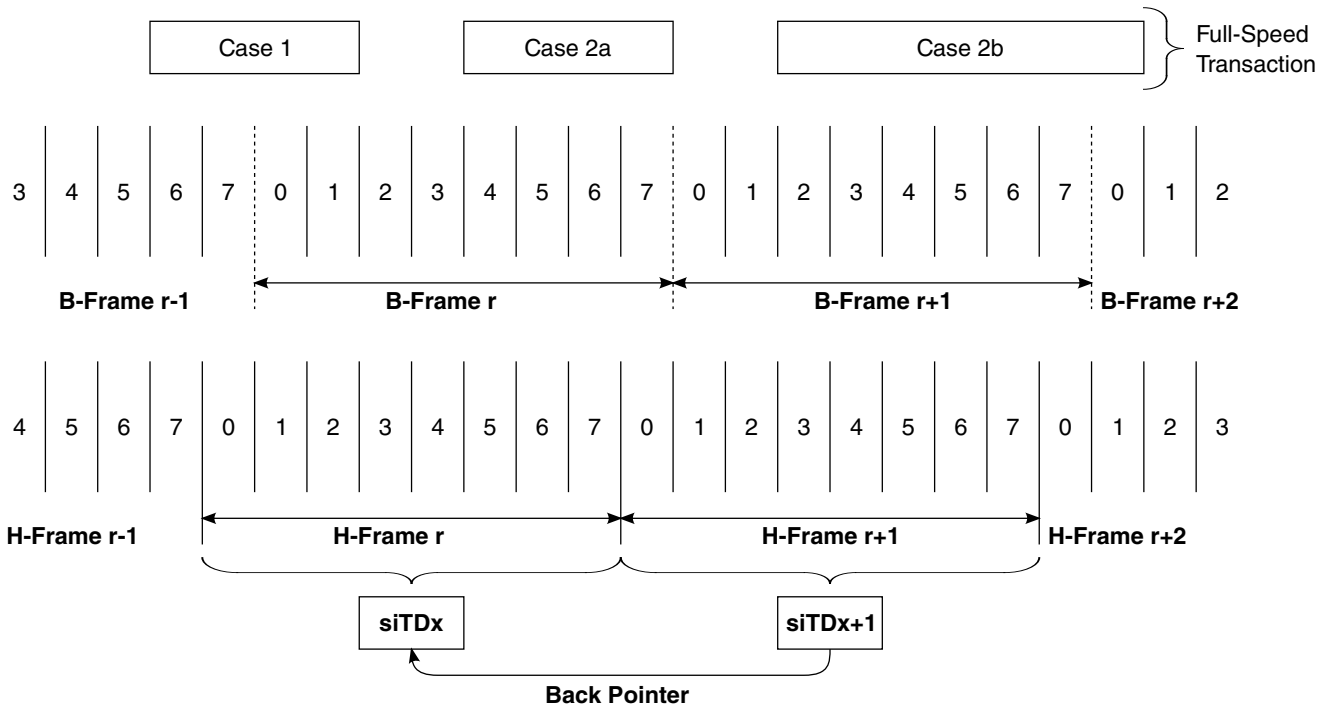


Figure 54-26. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction $siTD_X$ is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during micro-frame 7 of $H\text{-}Frame_{Y+1}$, or micro-frame 0 of $H\text{-}Frame_{Y+2}$. The complete splits are scheduled using $siTD_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from $siTD_{X+1}$. The only way for the host controller to reach $siTD_{X+1}$ from $H\text{-}Frame_{Y+2}$is to use $siTD_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section Periodic Isochronous - Do Complete Split .

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, micro-frame 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of *H-Frame* N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

54.4.3.12.3.2 Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD via the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-count*). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section Periodic Isochronous - Do Start Split for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction

isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (USB_n_FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. Refer to Section Asynchronous - Do Complete Split for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (for example, high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a micro-frame boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the

remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement its *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (for example, a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (that is, state not advanced) and report the appropriate error to the client driver.

54.4.3.12.3.3 Split Transaction Execution State Machine for Isochronous

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Interrupt Transfers , plus the variable *cMicroFrameBit* defined in Section Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. The figure below illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

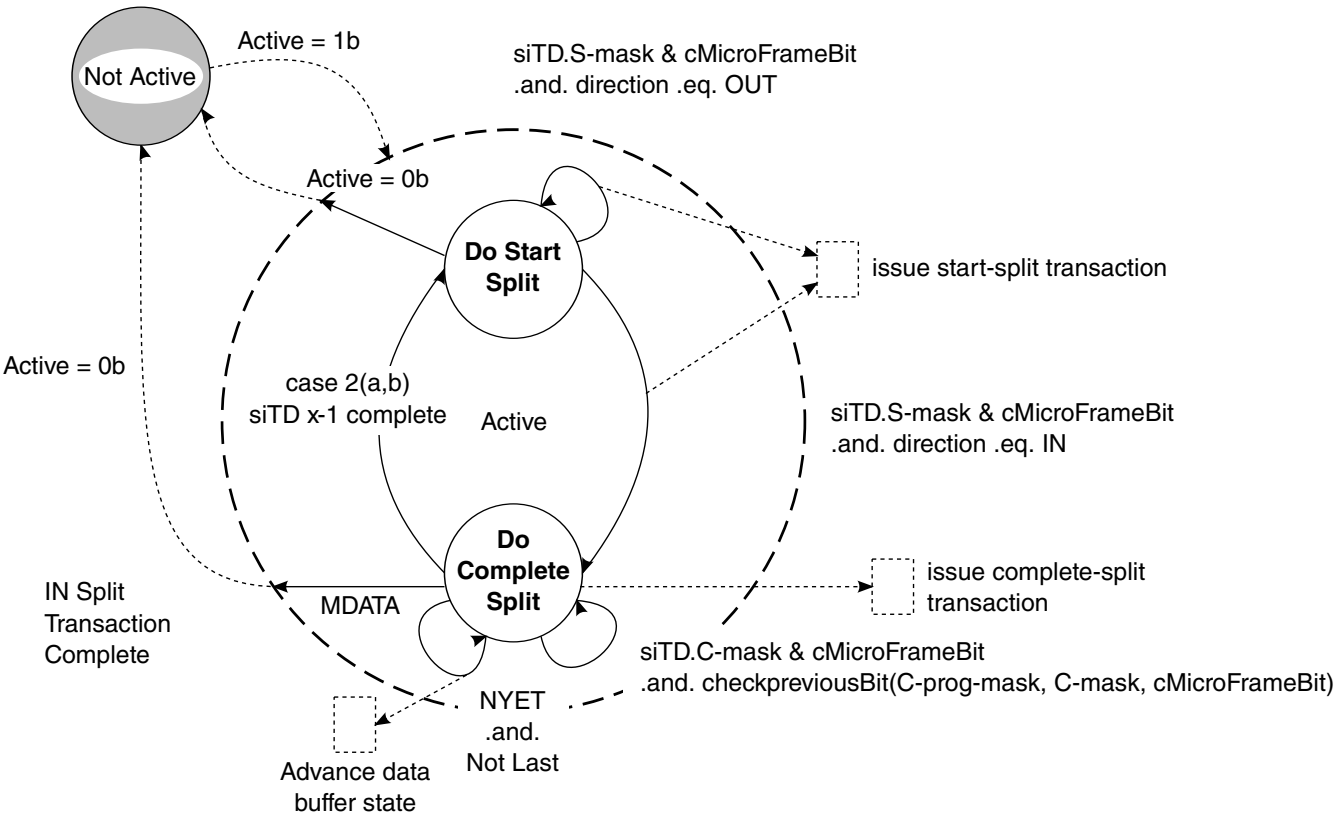


Figure 54-27. Split Transaction State Machine for Isochronous

54.4.3.12.3.4 Periodic Isochronous - Do Start Split

Isochronous split transaction OUTs use only this state.

An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (that is, the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory

buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 54-26) is used to determine the initial value of *TP*. The initial cases are summarized in the following table.

Table 54-48. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated.

The table below illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

.

Table 54-49. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (that is, greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately as defined in Table 54-49.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is *Total Bytes to Transfer* equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (that is, the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Section Split Transaction for Isochronous - Processing Examples .

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Periodic Isochronous - Do Complete Split can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

54.4.3.12.3.5 Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint.

This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which micro-frame the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- Test A. *cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD that is in this state.
- Test B. The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Section Periodic Isochronous - Do Complete Split). The sequence in which this test is applied depends on the current value of USB_n_FRINDEX[2:0]. If USB_n_FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
Begin
          Boolean rvalue = TRUE;
          previousBit = cMicroFrameBit rotate-right(1)
          -- Bit-wise anding previousBit with C-mask indicates whether there was an intent
          -- to send a complete split in the previous micro-frame. So, if the
          -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it
          -- happened.
          if previousBit bitAND siTD.C-mask then
                    if not (previousBit bitAND siTD.C-prog-mask) then
                              rvalue = FALSE
                    End if
          End if
          Return rvalue
End Algorithm
```

If Test A is true and USB_n_FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 54-25). See Section Periodic Isochronous - Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer,*
- Adjust *siTD.Current Offset* by the number of bytes received,

- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.
- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, etc. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the *Active* bit is set to zero.
- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Section Periodic Isochronous - Do Complete Split . If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs,

meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.

- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator will respond with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Micro-Frame* status bit and sets the *Active* bit to a zero.

54.4.3.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 54-25) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. The table below enumerates the transaction state fields.

Table 54-50. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and *T-count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer*

field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is Do Complete Split.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

In order to access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$.*Back Pointer*.

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Complete Split), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 54-50) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of $siTD_X$, and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Start Split), then the host controller must set *Active* bit to a zero and *Missed Micro-Frame* status bit to a one and the resultant status written back to memory.

If $siTD_{X-1}$*'sActive* bit is a zero, (because it was zero when the host controller first visited $siTD_{X-1}$ via $siTD_X$*'s* back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$*'s* complete-split transaction transitioned it to zero), then the host controller returns to the context of $siTD_X$ and transitions its *SplitXState* to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (that is, if *cMicroframeBit* is a 1b and $siTD_X$.*S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of $siTD_X$, then follows $siTD_X$.*Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$.*Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ will have its *Active* bit set to zero when the host controller returns

to the context of $siTD_X$. Also, note that software should not initialize an siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

54.4.3.12.3.7 Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines.

The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced via the Execute Transaction queue head traversal state machine (see Figure 54-18).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, the table below illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 54-51. Example Case 2a - Software Scheduling siTDs for an IN Endpoint

siTDX		Micro-Frames								Initial
#	Masks	0	1	2	3	4	5	6	7	SplitXState
X	S-Mask	-	-	-	-	1	-	-	-	Do Start Split
	C-Mask	1	1	-	-	-	-	1	1	
X+1	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Because this is the case-2a frame-wrap case, *S-masks* of all siTDs for this endpoint have a value of 10h (a one bit in micro-frame 4) and *C-mask* value of C3h (one-bits in micro-frames 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back PointerT-bits* are set to zero).

The initial *SplitXState* of the first siTD is Do Start Split. The host controller will visit the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is Do Start Split. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to Do Complete Split. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has its *SplitXState* initialized to Do Complete Split. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During micro-frames 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to Do Start Split. At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes early (transaction-complete is defined in Section Periodic Isochronous - Do Complete Split), that is, before all the scheduled complete-splits have been executed, the host controller will transition $siTD_X$.*SplitXState* to Do Start Split early and naturally skip the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, micro-frame 1.

During *H-Frame X+2,* micro-frame 0, the host controller detects that $siTD_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of $siTD_{X+2}$, and traverses its next pointer without any state change updates to $siTD_{X+2}$. S

During *H-Frame X+2*, micro-frame 1, the host controller detects $siTD_{X+2}$'s *S-mask[0]* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of $siTD_{X+2}$ and changes its *SplitXState* to Do Start Split. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it

reaches micro-frame 4. <TBD… describe how software detects that there was missing micro-frames (don't think we care about missing out micro-frames. There is enough residual state to identify than not all transactions were executed.).

54.4.3.13 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports.

When the schedules are enabled, the EHCI host controller will access the schedules in main memory each micro-frame. This constant pinging of main memory is known to create ARM platform power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the ARM platform, based on recent history usage. In the more aggressive power saving modes, the ARM platform can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the ARM platform power management software can detect this activity over time and inhibit the transition of the ARM platform into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the ARM platform power management software from placing the ARM platform into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the ARM platform power management to get the ARM platform into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the ARM platform to reach its low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the ARM platform will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing its shared memory based data structures (schedule lists or otherwise).

54.4.3.14 Port Test Modes -Host Operational Model

EHCI host controllers must implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK as described in the USB Specification Revision 2.0.

The system is only allowed to test ports that are owned by the EHCI controller (for example, *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the USB_n_CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate USB_n_PORTSC register to a one.
- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting *HCReset* to a one.

54.4.3.15 Interrupts-Host Operational Model

The EHCI Host Controller hardware provides interrupt capability based on a number of sources.

There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host Controller error events

All transaction-based sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section Interrupt Enable Register (USB_*n*USBINTR)). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller will not generate interrupts any more frequently than once every eight micro-frames.

Section Host System Error details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, ARM platform control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to de-assert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register, see Section Interrupt Enable Register (USB_*n*USBINTR)) to a zero. The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register (Section USB Status Register (USB_*n*USBSTS)) from a one to a zero.

54.4.3.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

54.4.3.15.1.1 Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully.

The table below lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the *XactErr* status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 54-52. Summary of Transaction Errors

Event / Result	Queue Head/qTD/iTD/siTD Side-effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	-1	XactErr set to a one.	1[1]
Timeout	-1	XactErr set to a one.	1[1]
Bad PID[2]	-1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	

1. If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Halting a Queue Head.
2. The host controller received a response from the device, but it could not recognize the PID as a valid PID.

54.4.3.15.1.2 Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*.

When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Halting a Queue Head). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USB_n_USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babble across a micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence.

The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

54.4.3.15.1.3 Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction.

This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

54.4.3.15.1.4 USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USB_n_USBSTS register to be set to a one.

In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USB_n_USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USB_n_USBSTS register is also set to a one.

54.4.3.15.1.5 Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USB_n_USBSTS register is set to a one.

If the *USB Interrupt Enable* bit is set in the USB_n_USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

54.4.3.15.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section Interrupt on Async Advance).

54.4.3.15.2.1 Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one.

If the *Port Change Interrupt Enable* bit in the USB_n_USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

54.4.3.15.2.2 Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs.

If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USB.USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USB.USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

54.4.3.15.2.3 Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USB.USBCMD register.

If it is a one, it sets the *Interrupt on Async Advance* bit in the USB.USBSTS register to a one. If the *Interrupton Async Advance Enable* bit in the USB.USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule .

54.4.3.15.2.4 Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors.

The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion.In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USB.USBCMD register is set to a zero.
- The following bits in the USB.USBSTS register are set:
  - *Host System Error* bit is to a one.
  - *HCHalted* bit is set to a one.
- If the *Host System Error Enable* bit in the USB.USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. The following table summarizes the required actions taken on the various host errors.

Table 54-53. Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a *Host System Error*, Software must reset the host controller through *HCReset* in the USB.USBCMD register before re-initializing and restarting the host controller.

54.4.4 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specification. Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary.

The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator - Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

54.4.4.1 Embedded Transaction Translator Function

The OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

54.4.4.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to USB.HCSPARAMS - Host Control Structural Parameters
- N_PTT added to USB.HCSPARAMS - Host Control Structural Parameters

54.4.4.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- Addition of two-bit Port Speed (PSPD) to the Port Status & Control (USB_*n*PORTSC1) register.

54.4.4.1.3 Discovery-EHCI Deviation

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation.

The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, Chirp completes successfully).

Because this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in USB.PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in the following table.

Table 54-54. Summary of EHCI

Standard EHCI	EHCI with embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (ie. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (ie. Split target hub is the root hub)]

54.4.4.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator.

Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS) - Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - Hub Address = 0
   - Transactions to direct attached device/hub.
     - QH.EPS = Port Speed
   - Transactions to a device downstream from direct attached FS hub.
     - QH.EPS = Downstream Device Speed

NOTE

When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

Maximum Packet Size must be less than or equal 64 or undefined behaviour may result.

2. siTD (for direct attach FS) - Periodic (ISO Endpoint)
   - All FS ISO transactions:
     - Hub Address = 0
     - siTD.EPS = 00 (full speed)
       - Maximum Packet Size must less than or equal to 1023 or undefined behaviour may result.

54.4.4.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specification. Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Because the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

54.4.4.1.5.1 Micro- frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

54.4.4.1.5.2 Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator.

Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. The following table summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 54-55. Summary of the Conditons of Handshakes[1]

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (Ok)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

1. The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits

54.4.4.1.5.3 Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

54.4.4.1.5.3.1 USB 2.0 - 11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

54.4.4.1.5.3.2 USB 2.0 - 11.17.4

- Transaction tracking for 2 data pipes.

54.4.4.1.5.4 Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

54.4.4.1.5.4.1 USB 2.0 - 11.18.6.[1-2]

- Abort of pending start-splits
  - EOF (and not started in micro-frames 6)
  - Idle for more than 4 micro-frames
- Abort of pending complete-splits
  - EOF
  - Idle for more than 4 micro-frames

54.4.4.1.5.4.2 USB 2.0 - 11.18.[7-8]

- Transaction tracking for up to 16 data pipes.
- Complete-split transaction searching.

NOTE

There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

54.4.4.1.5.5 Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the Host Controller Structural Parameters (USB_*n*HCSPARAMS) register.

54.4.4.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

54.4.4.2.1 USB_USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USB Device Mode (USB_*n*USBMODE) register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

54.4.4.2.2 Non-Zero Fields the Register File

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

54.4.4.2.3 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode.

EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USB Status Register (USB_*n*USBSTS) and Interrupt Enable Register (USB_*n*USBINTR) registers.

54.4.4.3 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

54.4.4.3.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. That is, a 60 Mhz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

54.4.4.4 Miscellaneous variations from EHCI

54.4.4.4.1 Programmable Physical Interface Behaviour

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes.

Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase.

The control bits for selecting the Physical Interface operating mode have been added to the Port Status & Control (USB_*n*PORTSC1) register providing a capability that is not defined by EHCI.

54.4.4.4.2 Discovery

54.4.4.4.2.1 Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the Port Status & Control (USB_*n*PORTSC1) register for a duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to reset the device.

- Software shall write a '0' to reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress, the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

54.4.4.4.2.2 Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation, which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices.

Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed - *This information is redundant with the 2-bit Port Speed indicator above.*

54.4.4.4.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI. An alternate host controller driver procedure is not necessary or supported.

54.4.5 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller.

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

NOTE

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writable fields. The device controller must preserve the read-only fields on all data structure writes.

The figure below shows the organization of the EndPoint Queue Head.

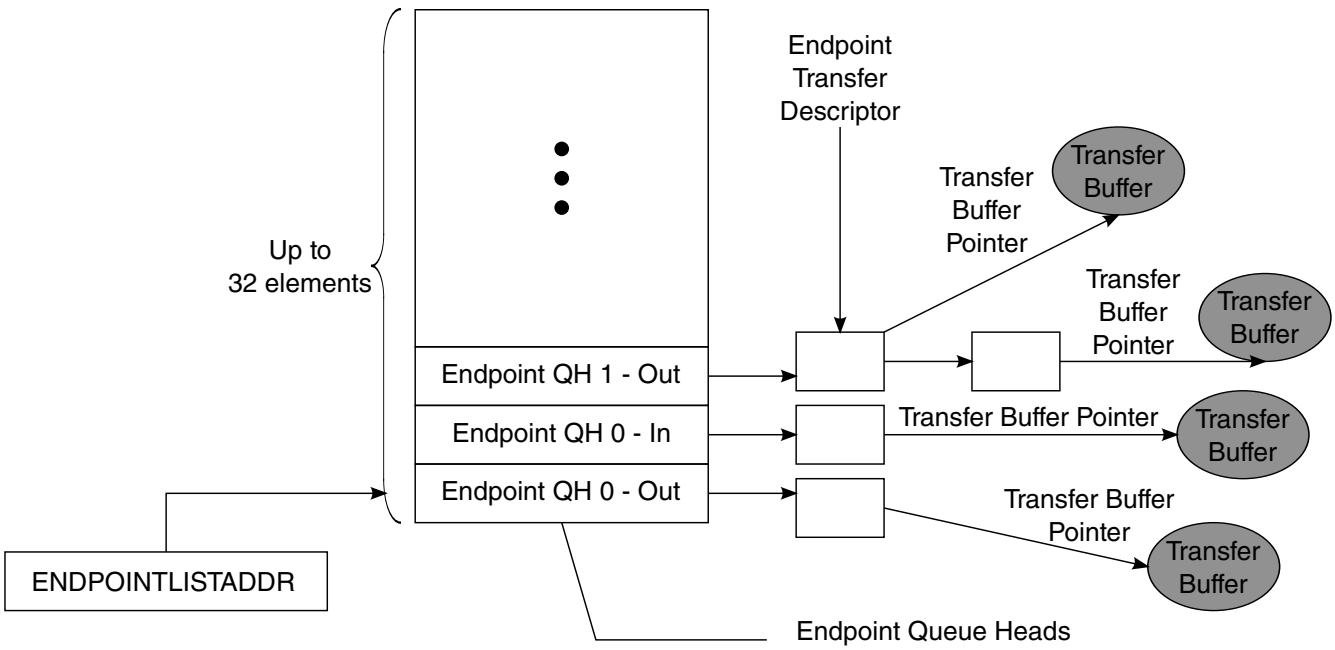


Figure 54-28. EndPoint Queue Head Organization

Endpoint queue heads are arranged in an array in a continuous area of memory pointed to by the USB.ENDPOINTLISTADDR pointer. The even -numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE

The Endpoint Queue Head List must be aligned to a 2k boundary.

54.4.5.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers for a given endpoint are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries.

During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

Table 54-56. Endpoint Queue Head (dQH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult		zlt	0		Maximum Packet Length											ios	0														
Current dTD Pointer																											0				
Next dTD Pointer																											0				T[1]
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				Reserved											
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											
Buffer Pointer (Page 4)[1]																				Reserved											
Reserved																															
Set-up Buffer Bytes 3…0																															
Set-up Buffer Bytes 7…4																															

1. Transfer overlay starts at T and continues through Buffer Pointer (Page 4).

Host Controller Read/Write

Host Controller Read Only

54.4.5.1.1 Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 54-57 describes the endpoint capabilities.

Table 54-57. Endpoint Capabilities/Characteristics

Bit	Description
31-30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 - Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. **NOTE:** Non-ISO endpoints must set Mult="00". ISO endpoints must set Mult="01", "10", or "11" as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0 - Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 - Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28-27	Reserved. These bit reserved for future use and should be set to zero.
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14-0	Reserved. Bits reserved for future use and should be set to zero.

54.4.5.1.2 Transfer Overlay-Endpoint Queue Head

The seven DWords in the overlay area represent a transaction working space for the device controller.

The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

54.4.5.1.3 Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

The following table describes the dTD Pointer.

Table 54-58. Next dTD Pointer

Bit	Description
31-5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4-0	Reserved. Bit reserved for future use and should be set to zero.

54.4.5.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

The following table describes the Multiple Mode Control.

Table 54-59. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31-0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31-0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

54.4.5.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/ received for a given transfer.

The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.

Table below shows the Endpoint Transfer Descriptor (dTD).

Table 54-60. Endpoint Transfer Descriptor (dTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Link Pointer																											0				T
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											

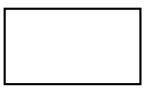
Table continues on the next page...

Table 54-60. Endpoint Transfer Descriptor (dTD) (continued)

Buffer Pointer (Page 1)	0	Frame Number
Buffer Pointer (Page 2)	Reserved	
Buffer Pointer (Page 3)	Reserved	
Buffer Pointer (Page 4)	Reserved	



Host Controller Read/Write



Host Controller Read Only

The following table describes the dTD Pointer.

Table 54-61. Next dTD Pointer

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

The following table describes the dTD Token.

Table 54-62. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30-16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K (5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K (4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14-12	Reserved. Bits reserved for future use and should be set to zero.
11-10	Multiplier Override (MultO). This field can be used for transmit ISO's (ie. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example:

Table continues on the next page...

Table 54-62. dTD Token (continued)

	if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO = "00".
9-8	Reserved. Bits reserved for future use and should be set to zero.
7-0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4, 2, 0 Reserved.

The table below describes the dTD Buffer Page Pointer List.

Table 54-63. dTD Buffer Page Pointer List

Bit	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0,11-0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1,10-0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

54.4.6 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus.

Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

54.4.6.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs.

Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

- Set Controller Mode in the USB.USBMODE register to device mode.

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USB.USBMODE.

- Allocate and Initialize device queue heads in system memory.
  - Minimum: Initialize device queue heads 0 Tx & 0 Rx.

NOTE

All device queue heads for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

  - For information on device queue heads, refer to section Device Data Structures.
- Configure USB.ENDPOINTLISTADDR Pointer.
  - For additional information on USB.ENDPOINTLISTADDR, refer to the register table.
- Enable the microprocessor interrupt associated with the USB core.
  - Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
  - For a list of available interrupts refer to the Interrupt Enable Register (USB_*n*USBINTR) and the USB Status Register (USB_*n*USBSTS) register tables.
- Set Run/Stop bit to Run Mode.
  - After the Run bit is set and the device is connected to a host, a Bus Reset will be issued by host downstream port. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the Port State and Control section below.

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to prime Endpoint 0 initially because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

54.4.6.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'.

After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0.

The following state diagram depicts the state of a USB 2.0 device.

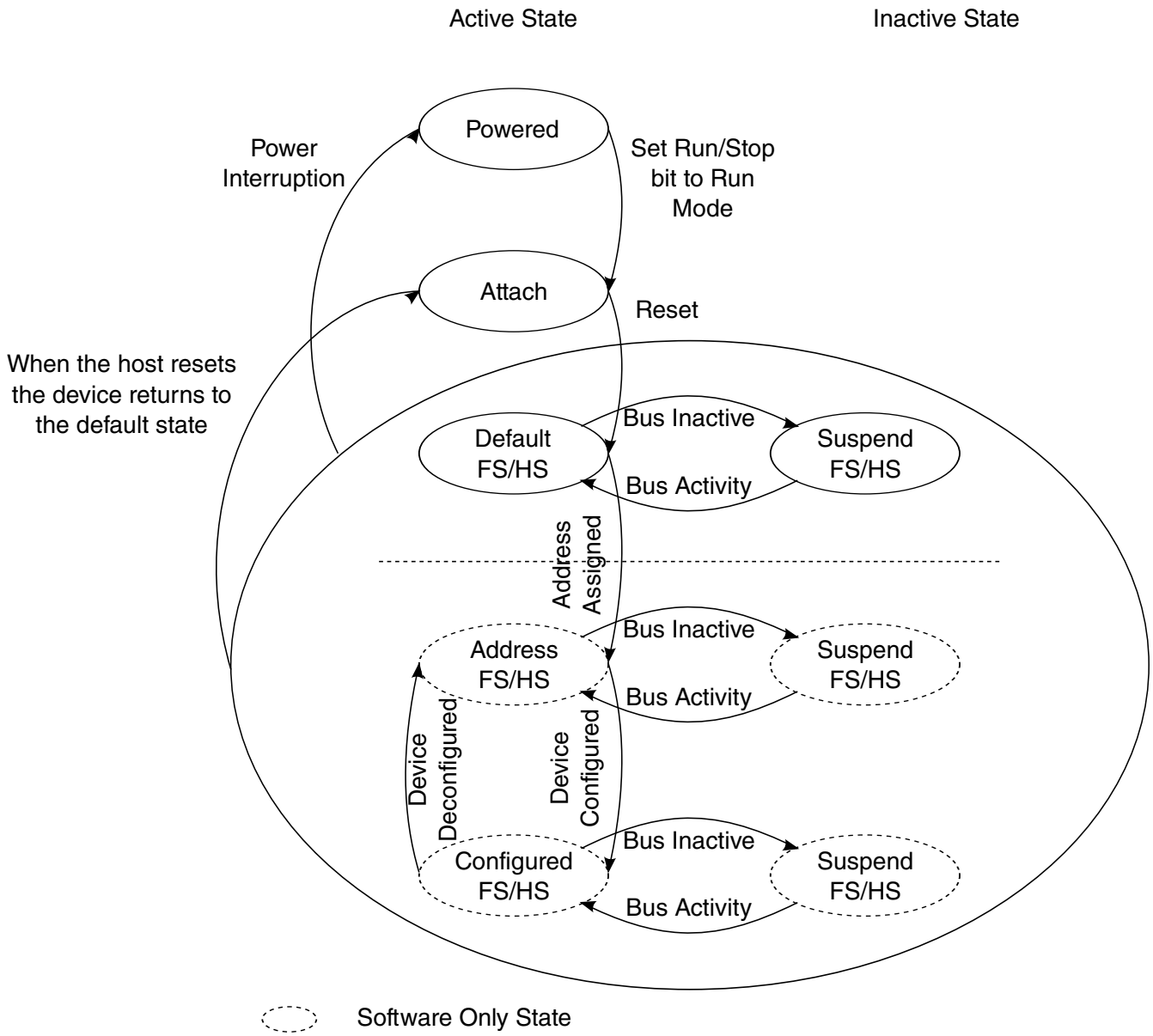


Figure 54-29. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

The following table describes the Device Controller State Information Bits.

Table 54-64. Device Controller State Information Bits

Bit	Register
DCSuspend	USB Status Register (USB_*n*USBSTS)
USB Reset Received	USB Status Register (USB_*n*USBSTS)
Port Change Detect	USB Status Register (USB_*n*USBSTS)
High-Speed Port	Port Status & Control (USB_*n*PORTSC1)

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the USB_UOG_ENDPTCTRLx registers and initializing the associated queue heads.

54.4.6.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices.

When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the Endpoint Status (USB_*n*ENDPTSTAT) register and writing the same value back to the Endpoint Status (USB_*n*ENDPTSTAT) register.

Clear all the endpoint complete status bits by reading the Endpoint Complete (USB_*n*ENDPTCOMPLETE) register and writing the same value back to the Endpoint Complete (USB_*n*ENDPTCOMPLETE) register.

Cancel all primed status by waiting until all bits in the Endpoint Prime (USB_*n*ENDPTPRIME) are 0 and then writing 0xFFFFFFFF to Endpoint Flush (USB_*n*ENDPTFLUSH).

Read the reset bit in the Port Status & Control (USB_*n*PORTSC1) register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the Port Status & Control (USB_*n*PORTSC1) to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

54.4.6.2.2 Suspend/Resume

54.4.6.2.2.1 Suspend

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the Port Status & Control (USB_*n*PORTSC1) is set to a '1', the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

NOTE

Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

54.4.6.2.2.2 Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port.

Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the Port Status & Control (USB_*n*PORTSC1) while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

NOTE

Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

54.4.6.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device.

The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and *isochronous*. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The USB OTG device controller hardware supports up to 8 endpoint numbers.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. To support the 8 endpoint numbers, 16 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

54.4.6.3.1 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the USB_UOG_ENDPTCTRLx register.

Each 32-bit USB_UOG_ENDPTCTRLx is split into an upper and lower half. The lower half of USB_UOG_ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the USB_UOG_ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization. The following table shows the fields and values for the Device Controller Endpoint initialization.

Table 54-65. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 Control 01 Isochronous 10 Bulk

Table continues on the next page...

Table 54-65. Device Controller Endpoint Initialization (continued)

	11 Interrupt
Endpoint Stall	0

54.4.6.3.2 Stalling

There are two occasions where the device controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the USB_UOG_ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the USB_UOG_ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the USB_UOG_ENDPTCTRLx register during operational mode must preserve the endpoint type field (that is, perform a read-modify-write).

The following table shows the response matrix for the Device Controller Stall.

Table 54-66. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'1'	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'0'	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	'1'	None	STALL

Table continues on the next page...

Table 54-66. Device Controller Stall Response Matrix (continued)

IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

54.4.6.3.3 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe.

For more information on data toggle, refer to the USB 2.0 specification.

54.4.6.3.3.1 Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the USB_UOG_ENDPTCTRLx register.

This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

54.4.6.3.3.2 Data Toggle Inhibit

NOTE

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

54.4.6.3.3.3 Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH).

After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the USB_UOG_ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Because only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

54.4.6.3.3.4 Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

54.4.6.4 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification.

At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1

transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "priming" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term "flushing" is used to describe the action of clearing a packet that was queued for execution.

54.4.6.4.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical.

All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 54-67. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	

Table continues on the next page...

Table 54-67. Variable Length Transfer Protocol Example (ZLT = 0) (continued)

512	256	3	256	256	0
512	512	2	512	0	

Table 54-68. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

NOTE

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

54.4.6.4.1.1 Interrupt/Bulk Endpoint Bus Response Matrix

The table below shows the response matrix for Interrput/Bulk Endpoint Bus.

Table 54-69. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

NOTE

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

54.4.6.4.2 Control Endpoint Operation Model

54.4.6.4.2.1 Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

The setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

- Disable Setup Lockout by writing 1 to Setup Lockout Mode (SLOM) in USB Device Mode (USB_*n*USBMODE). (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the Setup Lockout Mode As 0 will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT) to determine that a setup packet was received on a particular pipe:
  a. Write 1 to clear corresponding bit Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT).
  b. Write 1 to Setup Tripwire (SUTW) in USB Command Register (USB_*n*USBCMD) register.
  c. Duplicate contents of dQH.SetupBuffer into local software byte array.
  d. Read Setup TripWire (SUTW) in USB Command Register (USB_*n*USBCMD) register. (if set - continue; if cleared - goto 2)
  e. Write 0 to clear Setup Tripwire (SUTW) in USB Command Register (USB_*n*USBCMD) register.
  f. Process setup packet using local software byte array copy and execute status/ handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

54.4.6.4.2.2 Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the USB.ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the Endpoint Prime (USB_*n*ENDPTPRIME) register is zero and the associated bit in the Endpoint Status

(USB_*n*ENDPTSTAT) register is a one. If a prime fails, ie. The Endpoint Prime (USB_*n*ENDPTPRIME) bit goes to zero and the Endpoint Status (USB_*n*ENDPTSTAT) bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (Endpoint Status (USB_*n*ENDPTSTAT)) to enforce data coherency with the setup packet.

NOTE

- The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

54.4.6.4.2.3 Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase.

The DCD must also perform the same checks of the USB.ENDPTSETUPSTAT as described above in the data phase.

NOTE

- The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

54.4.6.4.2.4 Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

The table below shows the response matrix for the Control Endpoint Bus.

Table 54-70. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A

Table continues on the next page...

Table 54-70. Control Endpoint Bus Response Matrix (continued)

Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

54.4.6.4.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes.

Real time delivery by the device controller will be accomplished by the following:

- Exactly MULT Packets per (micro) Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro) frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - MULT counter reaches zero.
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - MULT counter reaches zero.
  - Non-MDATA Data PID is received**
    - ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
  - Overflow Error:
    - Packet received is > maximum packet length. [*Buffer Error* bit is set]
    - Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT
  - CRC Error [*Transaction Error* bit is set]

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

54.4.6.4.3.1 Isochronous Pipe Synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro) frame number (USB_UOG_FRINDEX register) can be used as a marker.

To cause a packet transfer to occur at a specific (micro) frame number [N], the DCD should interrupt on SOF during frame N-1. When the USB_UOG_FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro) frame N-1 so that the device controller will execute delivery during (micro) frame N.

NOTE

Priming an endpoint towards the end of (micro) frame N-1 will not guarantee delivery in (micro) frame N. The delivery may actually occur in (micro) frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

54.4.6.4.3.2 Isochronous Endpoint Bus Response Matrix

The following table shows the response matrix for the Isochronous Endpoint Bus.

Table 54-71. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. BS Error = Force Bit Stuff Error

   NULL Packet = Zero Length Packet

54.4.6.5 Managing Queue Heads

The following figure shows the End Point Queue Head.

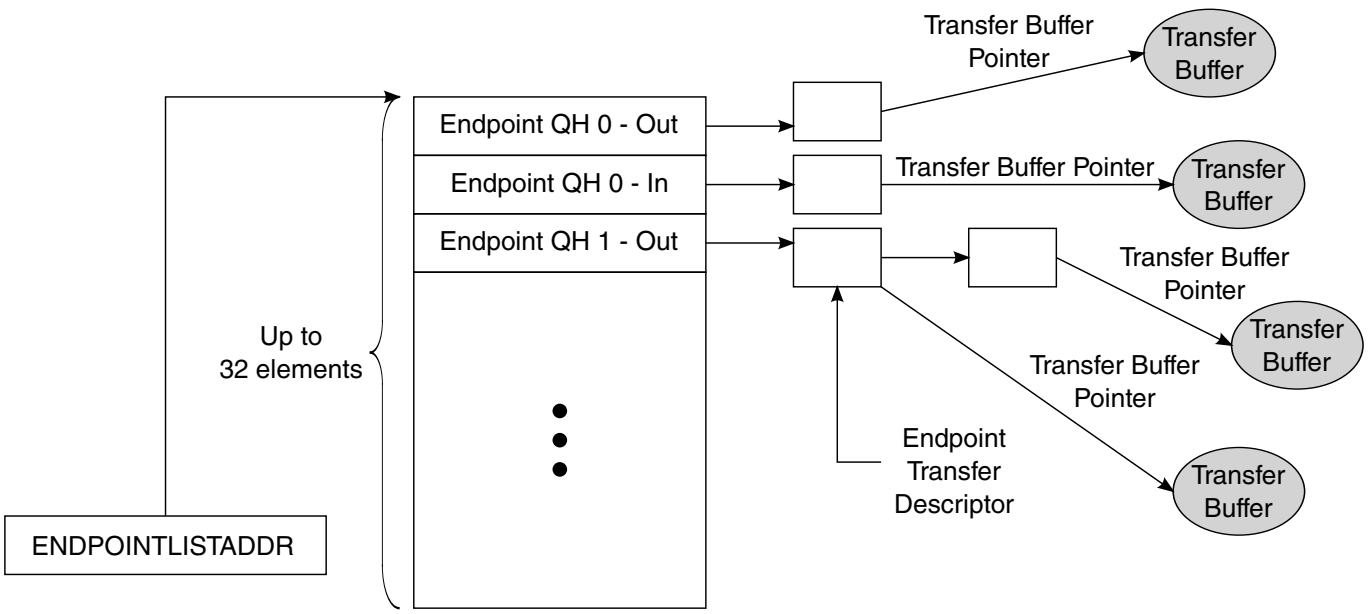


Figure 54-30. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by USB.ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 54-30. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors because pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multipler, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

54.4.6.5.1 Queue Head Initialization

One device queue head must be initialized for each active endpoint.

To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth and in conjunction with the USB Chapter 9 protocol.

NOTE

In FS mode, the multiplier field can only be 1 for ISO endpoints.

- Write the next dTD Terminate bit field to 1.
- Write the Active bit in the status field to 0.
- Write the Halt bit in the status field to 0.

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

54.4.6.5.2 Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

- The acknowledge must occur before continuing to process the setup packet.
- After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

54.4.6.6 Managing Transfers with Transfer Descriptors

54.4.6.6.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head.

This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list. The following figure shows the Software Link Pointers.

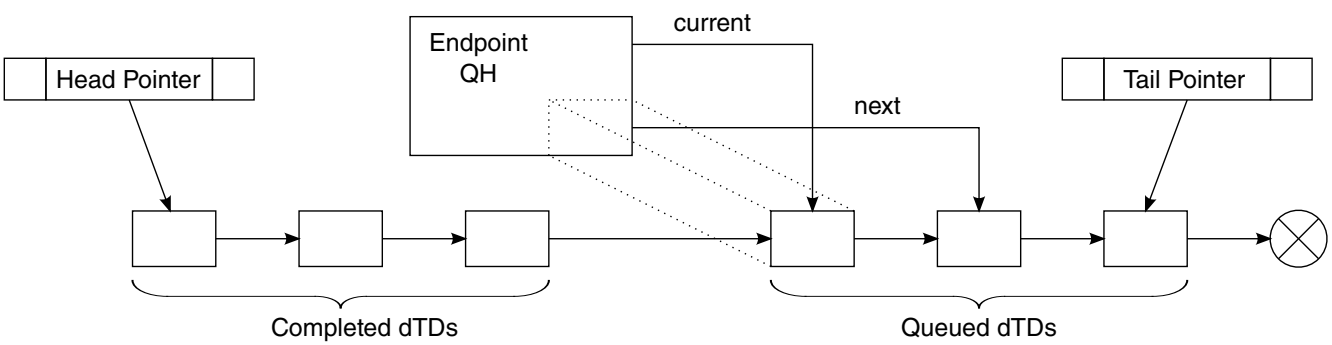


Figure 54-31. Software Link Pointers

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers, but it still remains the responsibility of the DCD to maintain the pointers.

54.4.6.6.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer.

Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to "00000"

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to 1.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to 1 and all remaining status bits set to 0.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

54.4.6.6.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty: Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding).

- Case 1: Link list is empty
  a. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
  b. Clear active & halt bit in dQH (in case set from a previous error).
  c. Prime endpoint by writing 1 to correct bit position in Endpoint Prime (USB_*n*ENDPTPRIME).
- Case 2: Link list is not empty
  a. Add dTD to end of linked list.
  b. Read correct prime bit in Endpoint Prime (USB_*n*ENDPTPRIME)- if 1 DONE.
  c. Set ATDTW bit in USBCMD register to 1.
  d. Read correct status bit in Endpoint Status (USB_*n*ENDPTSTAT). (store in tmp. variable for later)
  e. Read ATDTW bit in USBCMD register.
     - If 0 goto 3.
     - If 1 continue to 6.
  f. Write ATDTW bit in USBCMD register to 0.
  g. If status bit read in (3) is 1 DONE.

h. If status bit read in (3) is 0 then Goto Case 1: Step 1.

54.4.6.6.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

54.4.6.6.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer.

There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in Endpoint Flush (USB_*n*ENDPTFLUSH).
2. Wait until all bits in Endpoint Flush (USB_*n*ENDPTFLUSH) are '0'.

- Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.

3. Read Endpoint Status (USB_*n*ENDPTSTAT) to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
   - Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using Endpoint Flush (USB_*n*ENDPTFLUSH). A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

54.4.6.6.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 54-72. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated. The table below describes the errors.

Table 54-73. Error Descriptions

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length. ** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the "dead" (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

54.4.6.7 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

54.4.6.7.1 High-Frequency Interrupts

High frequency interrupts in particular should be handled in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

The table below describes the High frequency interrupt events.

Table 54-74. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt - USB.ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Figure 54-30 shows the End Point Queue Head). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt[1] - USB.ENDPTCOMPLETE	Handle completion of dTD as indicated in Figure 54-30 shows the End Point Queue Head.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1. It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

54.4.6.7.2 Low-Frequency Interrupts

The low frequency interrupts can be handled in any order because they do not occur often in comparison to the high-frequency interrupts.

The table below shows the Low frequency interrupt events.

Table 54-75. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

54.4.6.7.3 Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

The following table shows the error interrput events.

Table 54-76. Error Interrupt Events

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ USB.ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

54.5 USB Non-Core Memory Map/Register Definition

There are two kinds of registers in the USB module: USB core registers and USB non-core registers.

USB core registers are used to control USB core functions, and more independent of USB features. Each USB controller core has its own core registers.

USB non-core registera are additional to USB core registers, and more dependent on USB features. i.MX series products vary in non-core registers.

This section describes only the USB non-core registers. For detailed descriptions of USB core registers, please refer to Register Interface .

NOTE

- For reserved bits, please preserve the value when writing (read its reset value, then write this value back)
- "USB_UOG1_", "USB_UOG2_" prefix in register name indicates it is a core register for OTG1/OTG2 controller core respectively.
- USBNC_USB_" prefix in register name indicates it is a USB non-core register.

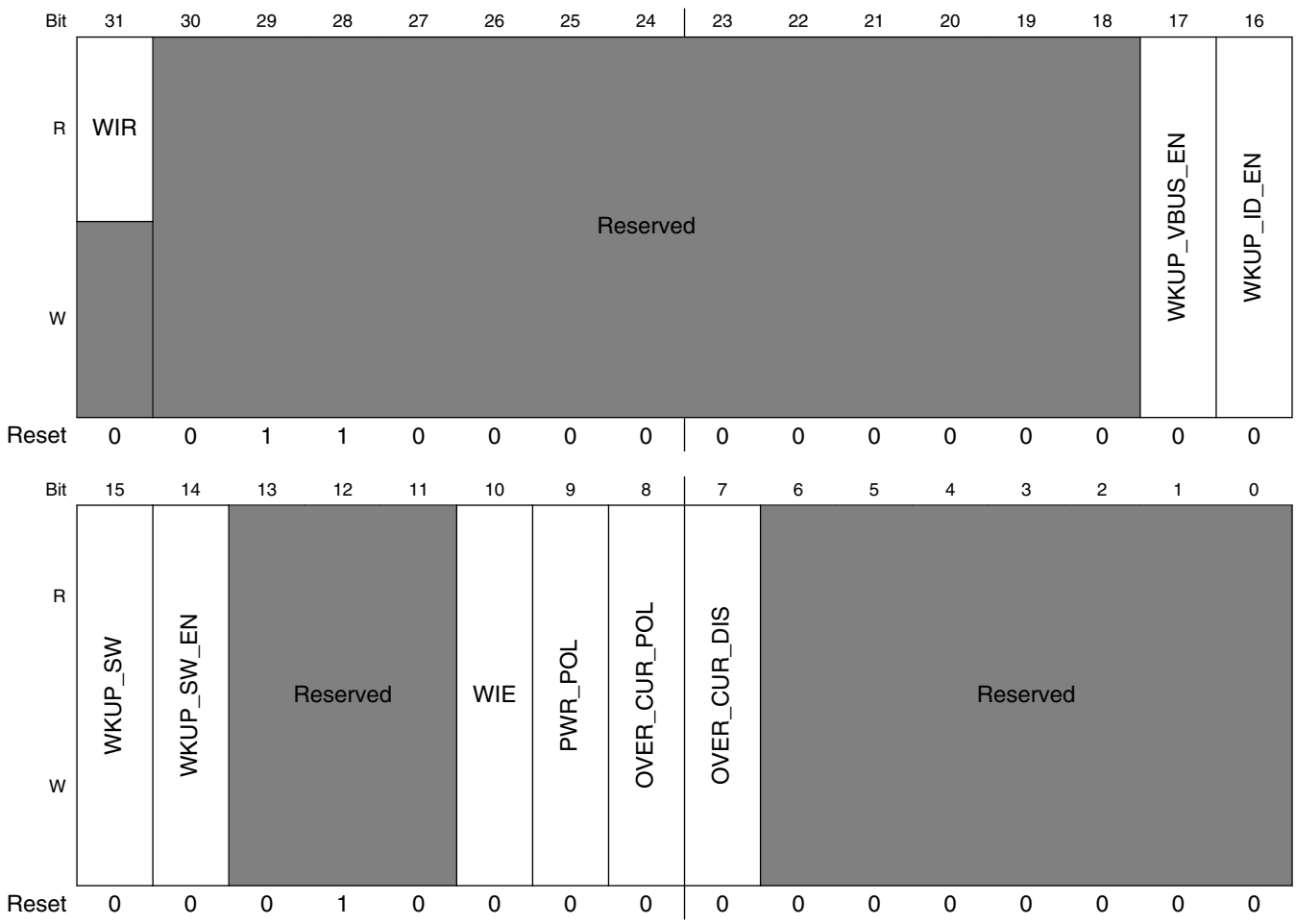
USBNC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4800	USB OTG1 Control Register (USBNC_USB_OTG1_CTRL)	32	R/W	3000_1000h	54.5.1/3373
218_4804	USB OTG2 Control Register (USBNC_USB_OTG2_CTRL)	32	R/W	3000_1000h	54.5.2/3375
218_4818	OTG1 UTMI PHY Control 0 Register (USBNC_USB_OTG1_PHY_CTRL_0)	32	R/W	8000_0000h	54.5.3/3377
218_481C	OTG2 UTMI PHY Control 0 Register (USBNC_USB_OTG2_PHY_CTRL_0)	32	R/W		54.5.4/3378

54.5.1 USB OTG1 Control Register (USBNC_USB_OTG1_CTRL)

The USB OTG1 control register controls the integration specific features of the USB OTG1 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 218_4000h base + 800h offset = 218_4800h



USBNC_USB_OTG1_CTRL field descriptions

Field	Description
31 WIR	OTG1 Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG1 port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received

Table continues on the next page...

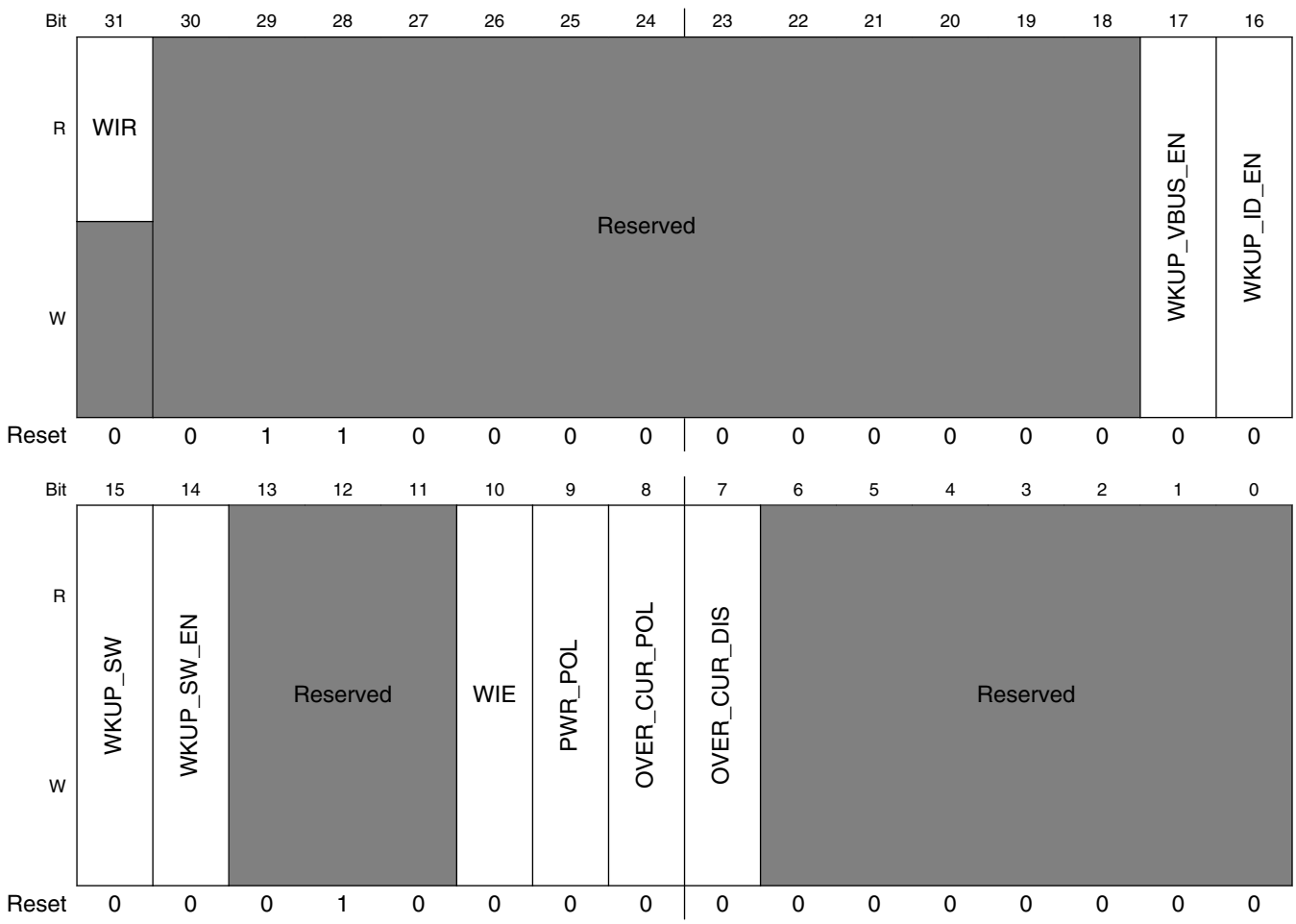
USBNC_USB_OTG1_CTRL field descriptions (continued)

Field	Description
30–18 -	This field is reserved. Reserved
17 WKUP_VBUS_EN	OTG1 wake-up on VBUS change enable 1 Enable 0 Disable
16 WKUP_ID_EN	OTG1 Wake-up on ID change enable 1 Enable 0 Disable
15 WKUP_SW	OTG1 Software Wake-up 1 Force wake-up 0 Inactive
14 WKUP_SW_EN	OTG1 Software Wake-up Enable 1 Enable 0 Disable
13–11 -	This field is reserved. Reserved
10 WIE	OTG1 Wake-up Interrupt Enable This bit enables or disables the OTG1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
9 PWR_POL	OTG1 Power Polarity This bit should be set according to PMIC Power Pin polarity. 1 PMIC Power Pin is High active. 0 PMIC Power Pin is Low active.
8 OVER_CUR_POL	OTG1 Polarity of Overcurrent The polarity of OTG1 port overcurrent event 1 Low active (low on this signal represents an overcurrent condition) 0 High active (high on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable OTG1 Overcurrent Detection 1 Disables overcurrent detection 0 Enables overcurrent detection
-	This field is reserved. Reserved

54.5.2 USB OTG2 Control Register (USBNC_USB_OTG2_CTRL)

The USB OTG2 control register controls the integration specific features of the USB OTG2 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 218_4000h base + 804h offset = 218_4804h



USBNC_USB_OTG2_CTRL field descriptions

Field	Description
31 WIR	OTG2 Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received

Table continues on the next page...

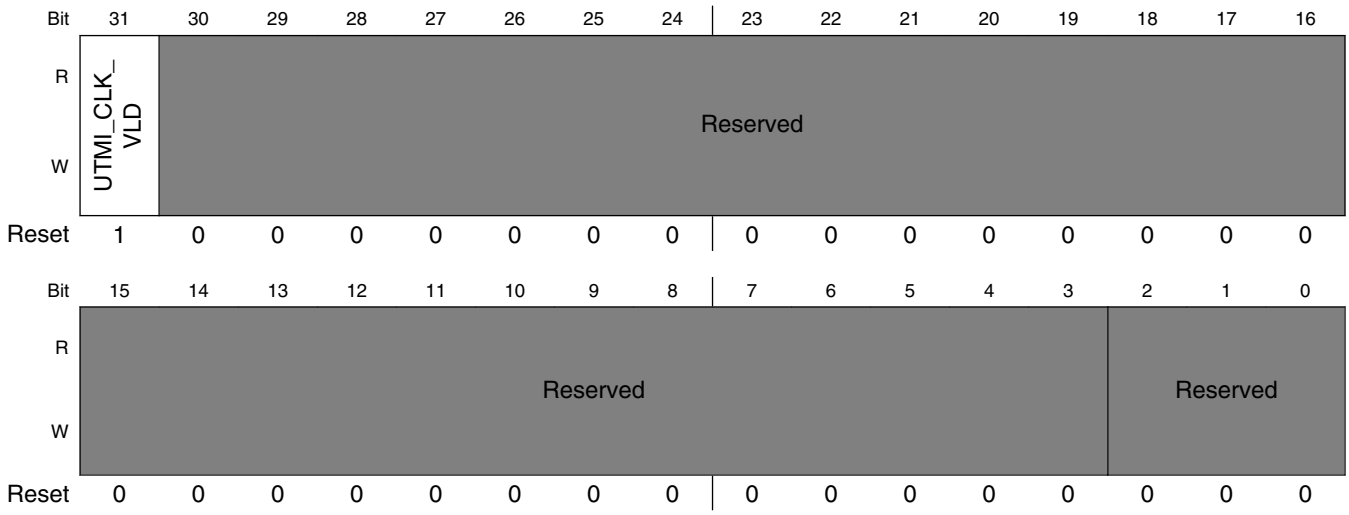
USBNC_USB_OTG2_CTRL field descriptions (continued)

Field	Description
30–18 -	This field is reserved. Reserved
17 WKUP_VBUS_EN	OTG2 wake-up on VBUS change enable 1 Enable 0 Disable
16 WKUP_ID_EN	OTG2 Wake-up on ID change enable 1 Enable 0 Disable
15 WKUP_SW	OTG2 Software Wake-up 1 Force wake-up 0 Inactive
14 WKUP_SW_EN	OTG2 Software Wake-up Enable 1 Enable 0 Disable
13–11 -	This field is reserved. Reserved
10 WIE	OTG2 Wake-up Interrupt Enable This bit enables or disables the OTG2 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
9 PWR_POL	OTG2 Power Polarity This bit should be set according to PMIC Power Pin polarity. 1 PMIC Power Pin is High active. 0 PMIC Power Pin is Low active.
8 OVER_CUR_POL	OTG2 Polarity of Overcurrent The polarity of OTG2 port overcurrent event 1 Low active (low on this signal represents an overcurrent condition) 0 High active (high on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable OTG2 Overcurrent Detection 1 Disables overcurrent detection 0 Enables overcurrent detection
-	This field is reserved. Reserved

54.5.3 OTG1 UTMI PHY Control 0 Register (USBNC_USB_OTG1_PHY_CTRL_0)

USB OTG1 UTMI PHY control register 0 is used to control the on-chip OTG1 UTMI PHY.

Address: 218_4000h base + 818h offset = 218_4818h



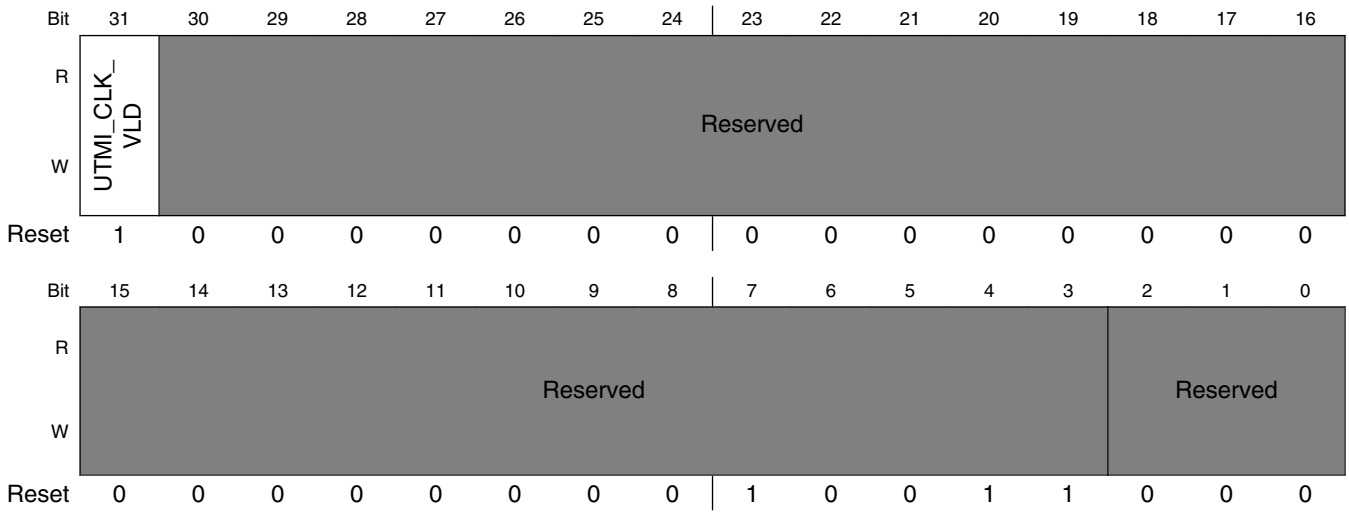
USBNC_USB_OTG1_PHY_CTRL_0 field descriptions

Field	Description
31 UTMI_CLK_VLD	Indicating whether OTG1 UTMI PHY clock is valid 1 Valid 0 Invalid
30–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

54.5.4 OTG2 UTMI PHY Control 0 Register (USBNC_USB_OTG2_PHY_CTRL_0)

USB OTG2 UTMI PHY Control Register 0 are used to control the on-chip OTG2 UTMI PHY.

Address: 218_4000h base + 81Ch offset = 218_481Ch



USBNC_USB_OTG2_PHY_CTRL_0 field descriptions

Field	Description
31 UTMI_CLK_VLD	Indicating whether OTG2 UTMI PHY clock is valid 1 Valid 0 Invalid
30–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

54.6 USB Core Memory Map/Register Definition

USB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4000	Identification register (USB_UOG1_ID)	32	R	E4A1_FA05h	54.6.1/3383

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4004	Hardware General (USB_UOG1_HWGENERAL)	32	R	0000_0035h	54.6.2/3383
218_4008	Host Hardware Parameters (USB_UOG1_HWHOST)	32	R	1002_0001h	54.6.3/3385
218_400C	Device Hardware Parameters (USB_UOG1_HWDEVICE)	32	R	0000_0011h	54.6.4/3386
218_4010	TX Buffer Hardware Parameters (USB_UOG1_HWTXBUF)	32	R	8008_0B08h	54.6.5/3386
218_4014	RX Buffer Hardware Parameters (USB_UOG1_HWRXBUF)	32	R	0000_0808h	54.6.6/3387
218_4080	General Purpose Timer #0 Load (USB_UOG1_GPTIMER0LD)	32	R/W	0000_0000h	54.6.7/3388
218_4084	General Purpose Timer #0 Controller (USB_UOG1_GPTIMER0CTRL)	32	R/W	0000_0000h	54.6.8/3388
218_4088	General Purpose Timer #1 Load (USB_UOG1_GPTIMER1LD)	32	R/W	0000_0000h	54.6.9/3390
218_408C	General Purpose Timer #1 Controller (USB_UOG1_GPTIMER1CTRL)	32	R/W	0000_0000h	54.6.10/ 3390
218_4090	System Bus Config (USB_UOG1_SBUSCFG)	32	R/W	0000_0002h	54.6.11/ 3391
218_4100	Capability Registers Length (USB_UOG1_CAPLENGTH)	8	R	40h	54.6.12/ 3392
218_4102	Host Controller Interface Version (USB_UOG1_HCIVERSION)	16	R	0100h	54.6.13/ 3393
218_4104	Host Controller Structural Parameters (USB_UOG1_HCSPARAMS)	32	R	0001_0011h	54.6.14/ 3393
218_4108	Host Controller Capability Parameters (USB_UOG1_HCCPARAMS)	32	R	0000_0006h	54.6.15/ 3395
218_4120	Device Controller Interface Version (USB_UOG1_DCIVERSION)	16	R	0001h	54.6.16/ 3397
218_4124	Device Controller Capability Parameters (USB_UOG1_DCCPARAMS)	32	R	0000_0188h	54.6.17/ 3398
218_4140	USB Command Register (USB_UOG1_USBCMD)	32	R/W	0008_0000h	54.6.18/ 3399
218_4144	USB Status Register (USB_UOG1_USBSTS)	32	R/W	0000_0000h	54.6.19/ 3403
218_4148	Interrupt Enable Register (USB_UOG1_USBINTR)	32	R/W	0000_0000h	54.6.20/ 3407
218_414C	USB Frame Index (USB_UOG1_FRINDEX)	32	R/W	0000_0000h	54.6.21/ 3409
218_4154	Frame List Base Address (USB_UOG1_PERIODICLISTBASE)	32	R/W	0000_0000h	54.6.22/ 3410
218_4154	Device Address (USB_UOG1_DEVICEADDR)	32	R/W	0000_0000h	54.6.23/ 3410
218_4158	Next Asynch. Address (USB_UOG1_ASYNCLISTADDR)	32	R/W	0000_0000h	54.6.24/ 3411
218_4158	Endpoint List Address (USB_UOG1_ENDPTLISTADDR)	32	R/W	0000_0000h	54.6.25/ 3412

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4160	Programmable Burst Size (USB_UOG1_BURSTSIZE)	32	R/W	0000_0808h	54.6.26/ 3412
218_4164	TX FIFO Fill Tuning (USB_UOG1_TXFILLTUNING)	32	R/W	000A_0000h	54.6.27/ 3413
218_4178	Endpoint NAK (USB_UOG1_ENDPTNAK)	32	R/W	0000_0000h	54.6.28/ 3415
218_417C	Endpoint NAK Enable (USB_UOG1_ENDPTNAKEN)	32	R/W	0000_0000h	54.6.29/ 3415
218_4180	Configure Flag Register (USB_UOG1_CONFIGFLAG)	32	R/W	0000_0001h	54.6.30/ 3416
218_4184	Port Status & Control (USB_UOG1_PORTSC1)	32	R/W	1000_0000h	54.6.31/ 3416
218_41A4	On-The-Go Status & control (USB_UOG1_OTGSC)	32	R/W	0000_1120h	54.6.32/ 3423
218_41A8	USB Device Mode (USB_UOG1_USBMODE)	32	R/W	0000_5000h	54.6.33/ 3427
218_41AC	Endpoint Setup Status (USB_UOG1_ENDPTSETUPSTAT)	32	R/W	0000_0000h	54.6.34/ 3428
218_41B0	Endpoint Prime (USB_UOG1_ENDPTPRIME)	32	R/W	0000_0000h	54.6.35/ 3429
218_41B4	Endpoint Flush (USB_UOG1_ENDPTFLUSH)	32	R/W	0000_0000h	54.6.36/ 3430
218_41B8	Endpoint Status (USB_UOG1_ENDPTSTAT)	32	R	0000_0000h	54.6.37/ 3430
218_41BC	Endpoint Complete (USB_UOG1_ENDPTCOMPLETE)	32	R/W	0000_0000h	54.6.38/ 3431
218_41C0	Endpoint Control0 (USB_UOG1_ENDPTCTRL0)	32	R/W	0080_0080h	54.6.39/ 3432
218_41C4	Endpoint Control 1 (USB_UOG1_ENDPTCTRL1)	32	R/W	0000_0000h	54.6.40/ 3434
218_41C8	Endpoint Control 2 (USB_UOG1_ENDPTCTRL2)	32	R/W	0000_0000h	54.6.41/ 3437
218_41CC	Endpoint Control 3 (USB_UOG1_ENDPTCTRL3)	32	R/W	0000_0000h	54.6.42/ 3439
218_41D0	Endpoint Control 4 (USB_UOG1_ENDPTCTRL4)	32	R/W	0000_0000h	54.6.43/ 3442
218_41D4	Endpoint Control 5 (USB_UOG1_ENDPTCTRL5)	32	R/W	0000_0000h	54.6.44/ 3445
218_41D8	Endpoint Control 6 (USB_UOG1_ENDPTCTRL6)	32	R/W	0000_0000h	54.6.45/ 3448
218_41DC	Endpoint Control 7 (USB_UOG1_ENDPTCTRL7)	32	R/W	0000_0000h	54.6.46/ 3451
218_4200	Identification register (USB_UOG2_ID)	32	R	E4A1_FA05h	54.6.1/3383
218_4204	Hardware General (USB_UOG2_HWGENERAL)	32	R	0000_0035h	54.6.2/3383

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4208	Host Hardware Parameters (USB_UOG2_HWHOST)	32	R	1002_0001h	54.6.3/3385
218_420C	Device Hardware Parameters (USB_UOG2_HWDEVICE)	32	R	0000_0011h	54.6.4/3386
218_4210	TX Buffer Hardware Parameters (USB_UOG2_HWTXBUF)	32	R	8008_0B08h	54.6.5/3386
218_4214	RX Buffer Hardware Parameters (USB_UOG2_HWRXBUF)	32	R	0000_0808h	54.6.6/3387
218_4280	General Purpose Timer #0 Load (USB_UOG2_GPTIMER0LD)	32	R/W	0000_0000h	54.6.7/3388
218_4284	General Purpose Timer #0 Controller (USB_UOG2_GPTIMER0CTRL)	32	R/W	0000_0000h	54.6.8/3388
218_4288	General Purpose Timer #1 Load (USB_UOG2_GPTIMER1LD)	32	R/W	0000_0000h	54.6.9/3390
218_428C	General Purpose Timer #1 Controller (USB_UOG2_GPTIMER1CTRL)	32	R/W	0000_0000h	54.6.10/ 3390
218_4290	System Bus Config (USB_UOG2_SBUSCFG)	32	R/W	0000_0002h	54.6.11/ 3391
218_4300	Capability Registers Length (USB_UOG2_CAPLENGTH)	8	R	40h	54.6.12/ 3392
218_4302	Host Controller Interface Version (USB_UOG2_HCIVERSION)	16	R	0100h	54.6.13/ 3393
218_4304	Host Controller Structural Parameters (USB_UOG2_HCSPARAMS)	32	R	0001_0011h	54.6.14/ 3393
218_4308	Host Controller Capability Parameters (USB_UOG2_HCCPARAMS)	32	R	0000_0006h	54.6.15/ 3395
218_4320	Device Controller Interface Version (USB_UOG2_DCIVERSION)	16	R	0001h	54.6.16/ 3397
218_4324	Device Controller Capability Parameters (USB_UOG2_DCCPARAMS)	32	R	0000_0188h	54.6.17/ 3398
218_4340	USB Command Register (USB_UOG2_USBCMD)	32	R/W	0008_0000h	54.6.18/ 3399
218_4344	USB Status Register (USB_UOG2_USBSTS)	32	R/W	0000_0000h	54.6.19/ 3403
218_4348	Interrupt Enable Register (USB_UOG2_USBINTR)	32	R/W	0000_0000h	54.6.20/ 3407
218_434C	USB Frame Index (USB_UOG2_FRINDEX)	32	R/W	0000_0000h	54.6.21/ 3409
218_4354	Frame List Base Address (USB_UOG2_PERIODICLISTBASE)	32	R/W	0000_0000h	54.6.22/ 3410
218_4354	Device Address (USB_UOG2_DEVICEADDR)	32	R/W	0000_0000h	54.6.23/ 3410
218_4358	Next Asynch. Address (USB_UOG2_ASYNCLISTADDR)	32	R/W	0000_0000h	54.6.24/ 3411
218_4358	Endpoint List Address (USB_UOG2_ENDPTLISTADDR)	32	R/W	0000_0000h	54.6.25/ 3412
218_4360	Programmable Burst Size (USB_UOG2_BURSTSIZE)	32	R/W	0000_0808h	54.6.26/ 3412

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4364	TX FIFO Fill Tuning (USB_UOG2_TXFILLTUNING)	32	R/W	000A_0000h	54.6.27/ 3413
218_4378	Endpoint NAK (USB_UOG2_ENDPTNAK)	32	R/W	0000_0000h	54.6.28/ 3415
218_437C	Endpoint NAK Enable (USB_UOG2_ENDPTNAKEN)	32	R/W	0000_0000h	54.6.29/ 3415
218_4380	Configure Flag Register (USB_UOG2_CONFIGFLAG)	32	R/W	0000_0001h	54.6.30/ 3416
218_4384	Port Status & Control (USB_UOG2_PORTSC1)	32	R/W	1000_0000h	54.6.31/ 3416
218_43A4	On-The-Go Status & control (USB_UOG2_OTGSC)	32	R/W	0000_1120h	54.6.32/ 3423
218_43A8	USB Device Mode (USB_UOG2_USBMODE)	32	R/W	0000_5000h	54.6.33/ 3427
218_43AC	Endpoint Setup Status (USB_UOG2_ENDPTSETUPSTAT)	32	R/W	0000_0000h	54.6.34/ 3428
218_43B0	Endpoint Prime (USB_UOG2_ENDPTPRIME)	32	R/W	0000_0000h	54.6.35/ 3429
218_43B4	Endpoint Flush (USB_UOG2_ENDPTFLUSH)	32	R/W	0000_0000h	54.6.36/ 3430
218_43B8	Endpoint Status (USB_UOG2_ENDPTSTAT)	32	R	0000_0000h	54.6.37/ 3430
218_43BC	Endpoint Complete (USB_UOG2_ENDPTCOMPLETE)	32	R/W	0000_0000h	54.6.38/ 3431
218_43C0	Endpoint Control0 (USB_UOG2_ENDPTCTRL0)	32	R/W	0080_0080h	54.6.39/ 3432
218_43C4	Endpoint Control 1 (USB_UOG2_ENDPTCTRL1)	32	R/W	0000_0000h	54.6.40/ 3434
218_43C8	Endpoint Control 2 (USB_UOG2_ENDPTCTRL2)	32	R/W	0000_0000h	54.6.41/ 3437
218_43CC	Endpoint Control 3 (USB_UOG2_ENDPTCTRL3)	32	R/W	0000_0000h	54.6.42/ 3439
218_43D0	Endpoint Control 4 (USB_UOG2_ENDPTCTRL4)	32	R/W	0000_0000h	54.6.43/ 3442
218_43D4	Endpoint Control 5 (USB_UOG2_ENDPTCTRL5)	32	R/W	0000_0000h	54.6.44/ 3445
218_43D8	Endpoint Control 6 (USB_UOG2_ENDPTCTRL6)	32	R/W	0000_0000h	54.6.45/ 3448
218_43DC	Endpoint Control 7 (USB_UOG2_ENDPTCTRL7)	32	R/W	0000_0000h	54.6.46/ 3451

54.6.1 Identification register (USB_*n*ID)

The ID register identifies the USB 2.0 High-Speed core and its revision.

Address: 218_4000h base + 0h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								REVISION							
W																
Reset	1	1	1	0	0	1	0	0	1	0	1	0	0	0	0	1

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

USB_*n*ID field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 REVISION	Revision number of the controller core.
15–14 -	This field is reserved. Reserved
13–8 NID	Complement version of ID
7–6 -	This field is reserved. Reserved
ID	Configuration number. This number is set to 0x05 and indicates that the peripheral is USB 2.0 High-Speed core.

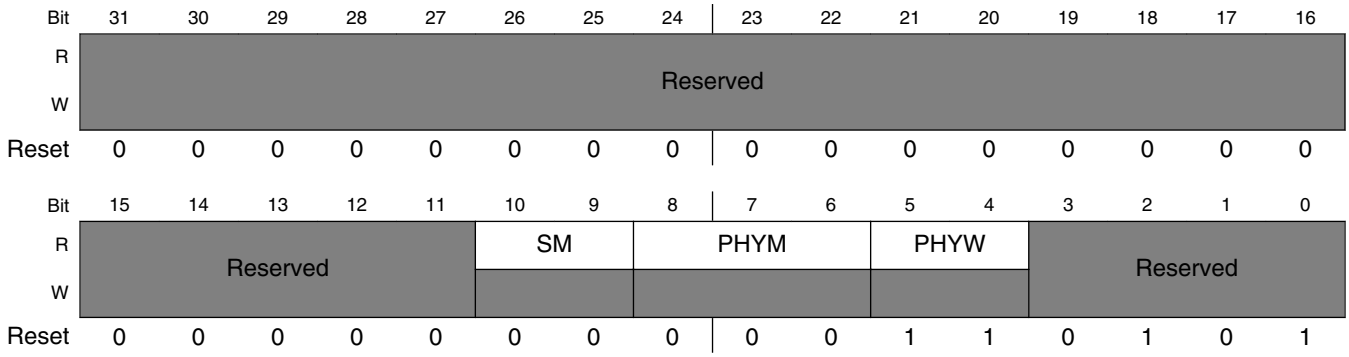
54.6.2 Hardware General (USB_*n*HWGENERAL)

General hardware parameters as defined in System Level Issues and Core Configuration.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: 218_4000h base + 4h offset + (512d × i), where i=0d to 1d



USB_*n*HWGENERAL field descriptions

Field	Description
31–11 -	This field is reserved. Reserved
10–9 SM	Serial interface mode capability 00 No Serial Engine, always use parallel signalling. 01 Serial Engine present, always use serial signalling for FS/LS. 10 Software programmable - Reset to use parallel signalling for FS/LS 11 Software programmable - Reset to use serial signalling for FS/LS
8–6 PHYM	Transciever type 000 UTMI/UMTI+ 001 ULPI DDR 010 ULPI 011 Serial Only 100 Software programmable - reset to UTMI/UTMI+ 101 Software programmable - reset to ULPI DDR 110 Software programmable - reset to ULPI 111 Software programmable - reset to Serial 1000 IC-USB 1001 Software programmable - reset to IC-USB 1010 HSIC 1011 Software programmable - reset to HSIC
5–4 PHYW	Data width of the transciever connected to the controller core. PHYW bit reset value is 00 8 bit wide data bus Software non-programmable 01 16 bit wide data bus Software non-programmable 10 Reset to 8 bit wide data bus Software programmable 11 Reset to 16 bit wide data bus Software programmable

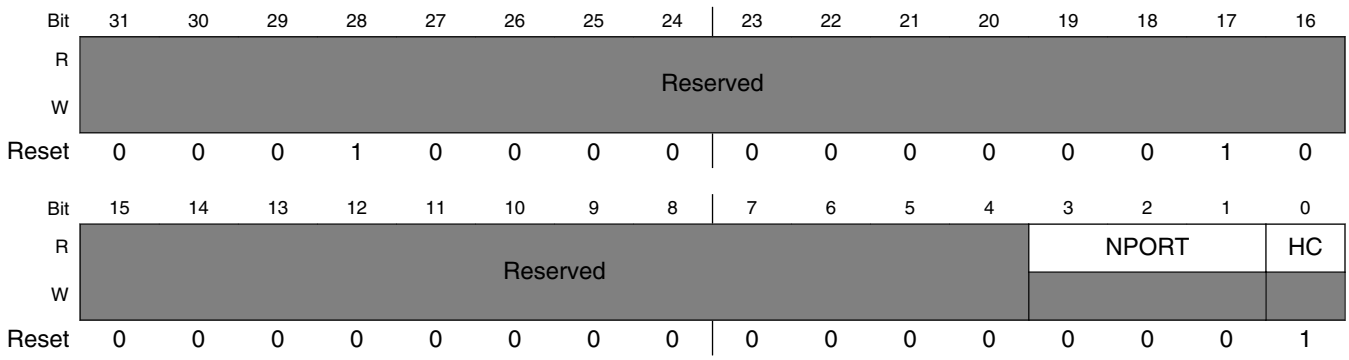
Table continues on the next page...

USB_*n*HWGENERAL field descriptions (continued)

Field	Description
-	This field is reserved. Reserved

54.6.3 Host Hardware Parameters (USB_*n*HWHOST)

Address: 218_4000h base + 8h offset + (512d × i), where i=0d to 1d

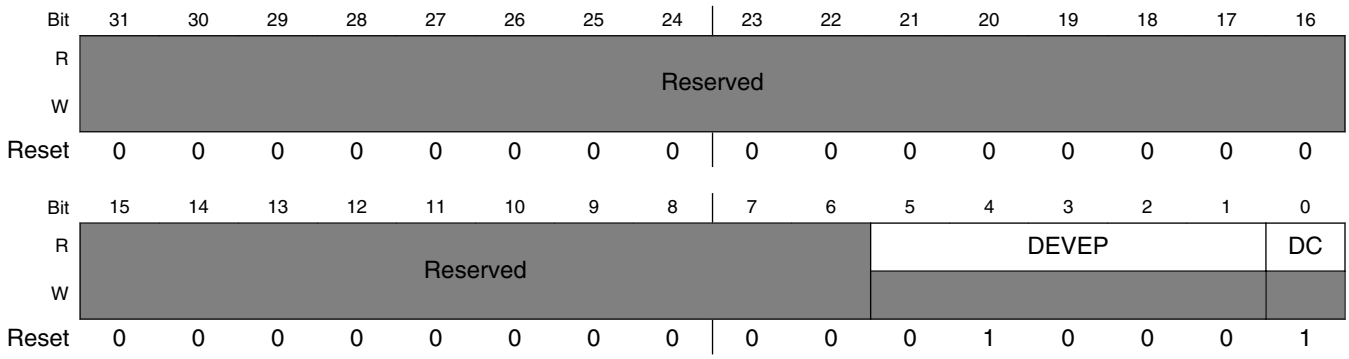


USB_*n*HWHOST field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3–1 NPORT	The Nmber of downstream ports supported by the host controller is (NPORT+1). **NOTE:** When these bits value is '000', it indicates a single-port host controller.
0 HC	Host Capable. Indicating whether host operation mode is supported or not. 1 Supported 0 Not supported

54.6.4 Device Hardware Parameters (USB_*n*HWDEVICE)

Address: 218_4000h base + Ch offset + (512d × i), where i=0d to 1d

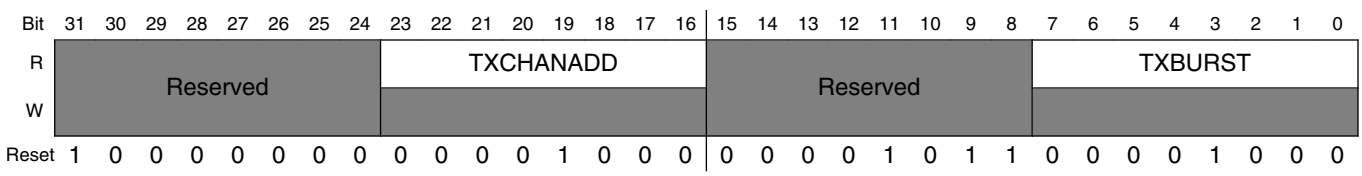


USB_*n*HWDEVICE field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–1 DEVEP	Device Endpoint Number
0 DC	Device Capable. Indicating whether device operation mode is supported or not. 1 Supported 0 Not supported

54.6.5 TX Buffer Hardware Parameters (USB_*n*HWTXBUF)

Address: 218_4000h base + 10h offset + (512d × i), where i=0d to 1d



USB_*n*HWTXBUF field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 TXCHANADD	TX FIFO Buffer size is: (2^TXCHANADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. For the OTG controller operating in device mode, this is the FIFO buffer size per endpoint. As the OTG controller has 8 TX endpoint, there are 8 of these buffers.

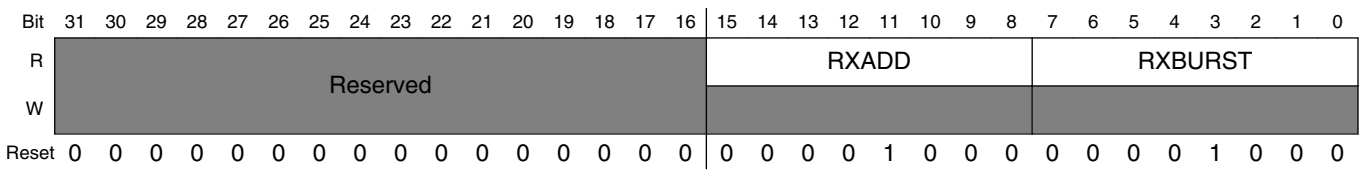
Table continues on the next page...

USB_*n*HWTXBUF field descriptions (continued)

Field	Description
	For the OTG controller operating in host mode, or for Host-only controller, the entire buffer memory is used as a single TX buffer. Therefore, there is only 1 of this buffer
15–8 -	This field is reserved. Reserved
TXBURST	Default burst size for memory to TX buffer transfer. This is reset value of TXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USB_*n*BURSTSIZE) .

54.6.6 RX Buffer Hardware Parameters (USB_*n*HWRXBUF)

Address: 218_4000h base + 14h offset + (512d × i), where i=0d to 1d



USB_*n*HWRXBUF field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 RXADD	Buffer total size for all receive endpoints is (2^RXADD). RX Buffer size is: (2^RXADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. There is a single Receive FIFO buffer in the USB controller. The buffer is shared for all endpoints for the OTG controller in device mode.
RXBURST	Default burst size for memory to RX buffer transfer. This is reset value of RXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USB_*n*BURSTSIZE) .

54.6.7 General Purpose Timer #0 Load (USB_*n*GPTIMER0LD)

This register controls load value of the count timer in register n_GPTIMER0CTRL. Please see General Purpose Timer #0 Controller (USB_*n*GPTIMER0CTRL) .

Address: 218_4000h base + 80h offset + (512d × i), where i=0d to 1d

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

USB_*n*GPTIMER0LD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

54.6.8 General Purpose Timer #0 Controller (USB_*n*GPTIMER0CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI0 bit in n_USBSTS register (See USB Status Register (USB_*n*USBSTS)), interrupt enable bit is TIE0 bit in n_USBINTR register. (See Interrupt Enable Register (USB_*n*USBINTR) .)

Address: 218_4000h base + 84h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

USB_*n*GPTIMER0CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset 0 No action 1 Load counter value from GPTLD bits in n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software; In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

54.6.9 General Purpose Timer #1 Load (USB_*n*GPTIMER1LD)

This register controls load value of the count timer in register n_GPTIMER1CTRL. Please see General Purpose Timer #1 Controller (USB_*n*GPTIMER1CTRL) .

Address: 218_4000h base + 88h offset + (512d × i), where i=0d to 1d

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

USB_*n*GPTIMER1LD field descriptions

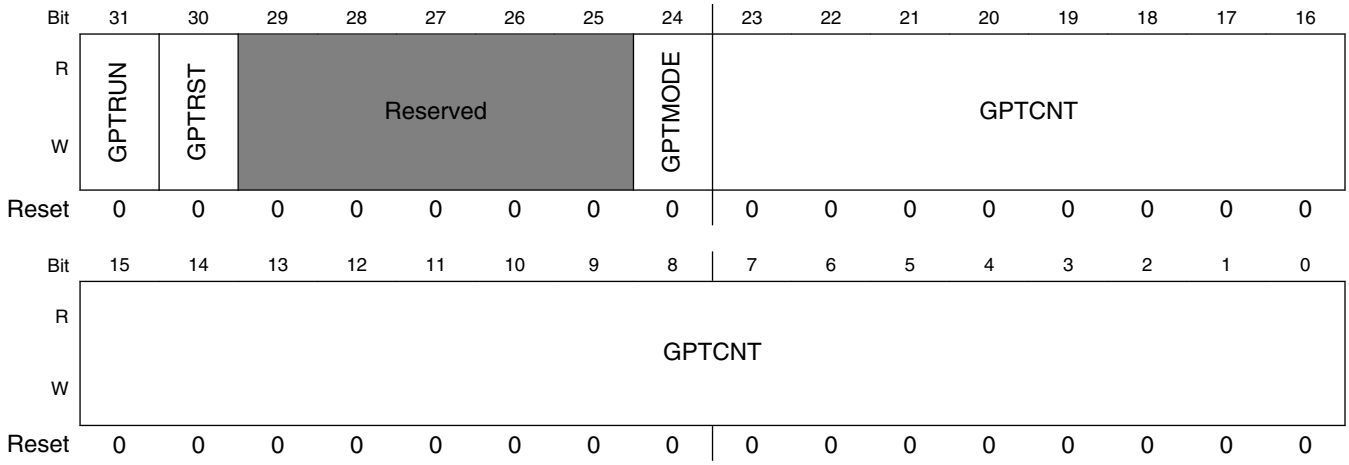
Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

54.6.10 General Purpose Timer #1 Controller (USB_*n*GPTIMER1CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI1 bit in USB_n_USBSTS register (See USB Status Register (USB_*n*USBSTS)), interrupt enable bit is TIE1 bit in n_USBINTR register (See Interrupt Enable Register (USB_*n*USBINTR)).

Address: 218_4000h base + 8Ch offset + (512d × i), where i=0d to 1d

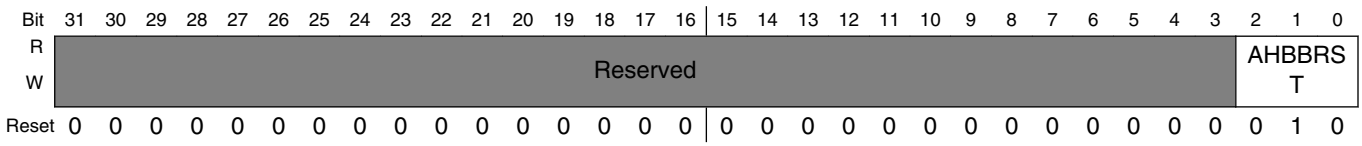


USB_*n*GPTIMER1CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset 0 No action 1 Load counter value from GPTLD bits in USB_n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

54.6.11 System Bus Config (USB_*n*SBUSCFG)

Address: 218_4000h base + 90h offset + (512d × i), where i=0d to 1d



USB_*n*SBUSCFG field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
AHBBRST	AHB master interface Burst configuration These bits control AHB master transfer type sequence (or priority). **NOTE:** This register overrides n_BURSTSIZE register when its value is not zero. 000 Incremental burst of unspecified length only 001 INCR4 burst, then single transfer 010 INCR8 burst, INCR4 burst, then single transfer 011 INCR16 burst, INCR8 burst, INCR4 burst, then single transfer 100 Reserved, don't use 101 INCR4 burst, then incremental burst of unspecified length 110 INCR8 burst, INCR4 burst, then incremental burst of unspecified length 111 INCR16 burst, INCR8 burst, INCR4 burst, then incremental burst of unspecified length

54.6.12 Capability Registers Length (USB_*n*CAPLENGTH)

The Capability Registers Length register contains the address offset to the Operational registers relative to the CAPLENGTH register.

Address: 218_4000h base + 100h offset + (512d × i), where i=0d to 1d

Bit	7	6	5	4	3	2	1	0
Read	CAPLENGTH							
Write								
Reset	0	1	0	0	0	0	0	0

USB_*n*CAPLENGTH field descriptions

Field	Description
CAPLENGTH	These bits are used as an offset to add to register base to find the beginning of the Operational Register. Default value is '40h'.

54.6.13 Host Controller Interface Version (USB_*n*HCIVERSION)

This is a 2-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.

Address: 218_4000h base + 102h offset + (512d × i), where i=0d to 1d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

USB_*n*HCIVERSION field descriptions

Field	Description
HCIVERSION	Host Controller Interface Version Number Default value is '10h', which means EHCI rev1.0.

54.6.14 Host Controller Structural Parameters (USB_*n*HCSPARAMS)

The following figure shows the port steering logic capabilities of Host Control Structural Parameters (n_HCSPARAMS).

Address: 218_4000h base + 104h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				N_TT				N_PTT				Reserved			PI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

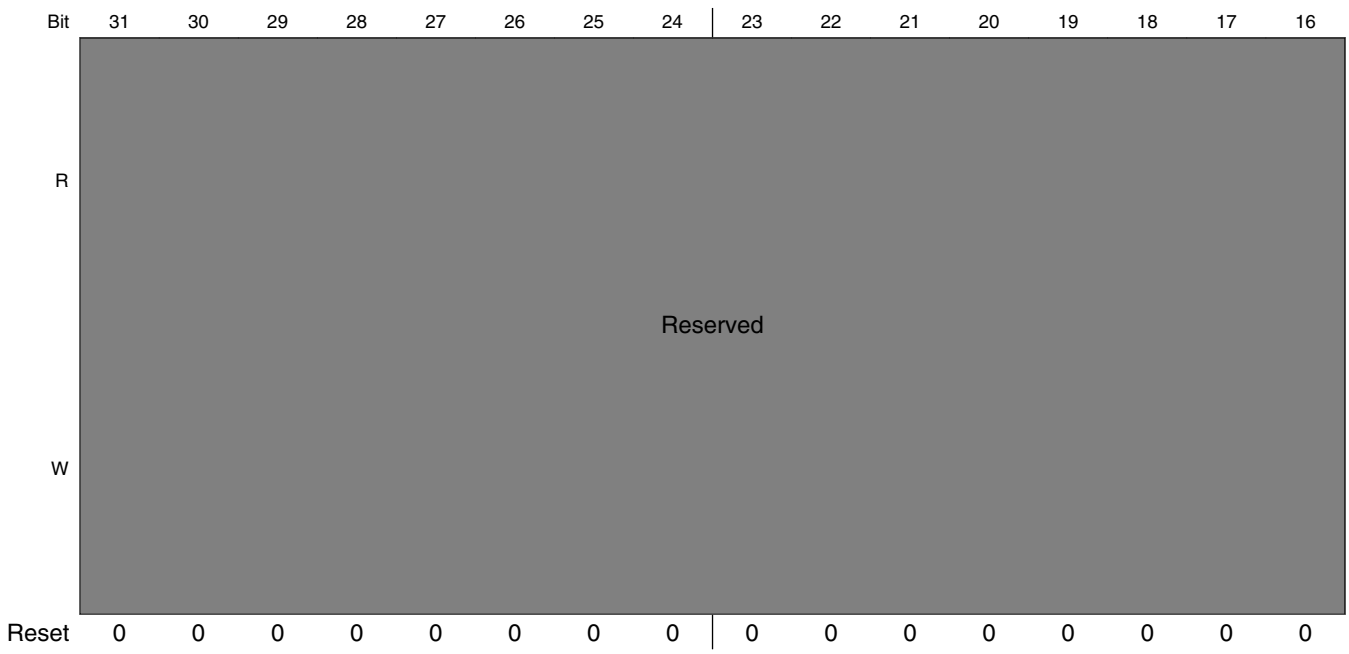
USB_*n*HCSPARAMS field descriptions

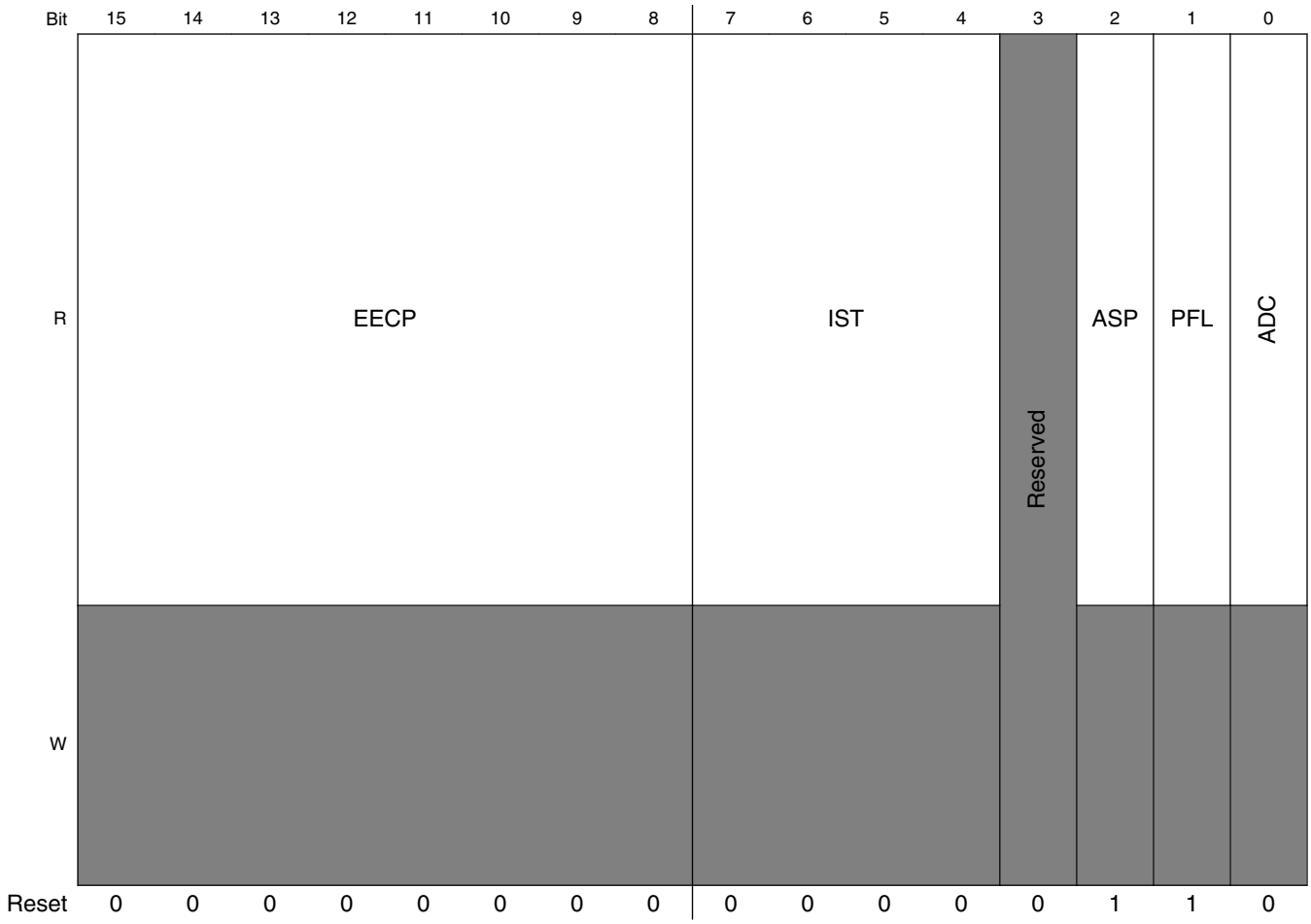
Field	Description
31–28 -	This field is reserved. Reserved
27–24 N_TT	Number of Transaction Translators (N_TT). Default value '0000b' This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. These bits would be set to '0001b' for Multi-Port Host, and '0000b' for Single-Port Host.
23–20 N_PTT	Number of Ports per Transaction Translator (N_PTT). Default value '0000b' This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. These bits would be set to equal N_PORTS for Multi-Port Host, and '0000b' for Single-Port Host.
19–17 -	This field is reserved. Reserved
16 PI	Port Indicators (P INDICATOR) This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator This bit is "1b" in all controller core.
15–12 N_CC	Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. These bits are '0000b' in all controller core. 0 There is no internal Companion Controller and port-ownership hand-off is not supported. 1 There are internal companion controller(s) and port-ownership hand-offs is supported.
11–8 N_PCC	Number of Ports per Companion Controller This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. These bits are '0000b' in all controller core.
7–5 -	This field is reserved. Reserved
4 PPC	Port Power Control This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register
N_PORTS	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. These bits are always set to '0001b' because all controller cores are Single-Port Host.

54.6.15 Host Controller Capability Parameters (USB_*n*HCCPARAMS)

This register identifies multiple mode control (time-base bit functionality), addressing capability.

Address: 218_4000h base + 108h offset + (512d × i), where i=0d to 1d





USB_*n*HCCPARAMS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 EECP	EHCI Extended Capabilities Pointer. This field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device. **NOTE:** These bits are set '00h' in all controller core.
7–4 IST	Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. These bits are set '00h' in all controller core.
3 -	This field is reserved. Reserved

Table continues on the next page...

USB_*n*HCCPARAMS field descriptions (continued)

Field	Description
2 ASP	Asynchronous Schedule Park Capability If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register. **NOTE:** ASP bit reset value: '00b' for OTG controller core, '11b' for Host-only controller core.
1 PFL	Programmable Frame List Flag If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This bit is set '1b' in all controller core.
0 ADC	64-bit Addressing Capability This bit is set '0b' in all controller core, no 64-bit addressing capability is supported.

54.6.16 Device Controller Interface Version (USB_*n*DCIVERSION)

This register indicates the two-byte BCD encoding of the device controller interface version number.

Address: 218_4000h base + 120h offset + (512d × i), where i=0d to 1d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

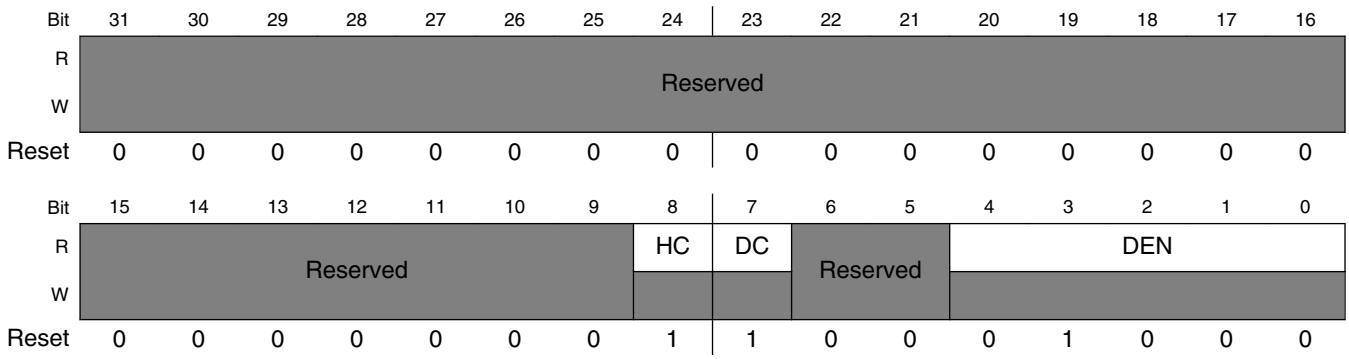
USB_*n*DCIVERSION field descriptions

Field	Description
DCIVERSION	Device Controller Interface Version Number Default value is '01h', which means rev0.1.

54.6.17 Device Controller Capability Parameters (USB_*n*DCCPARAMS)

These fields describe the overall device capability of the controller.

Address: 218_4000h base + 124h offset + (512d × i), where i=0d to 1d



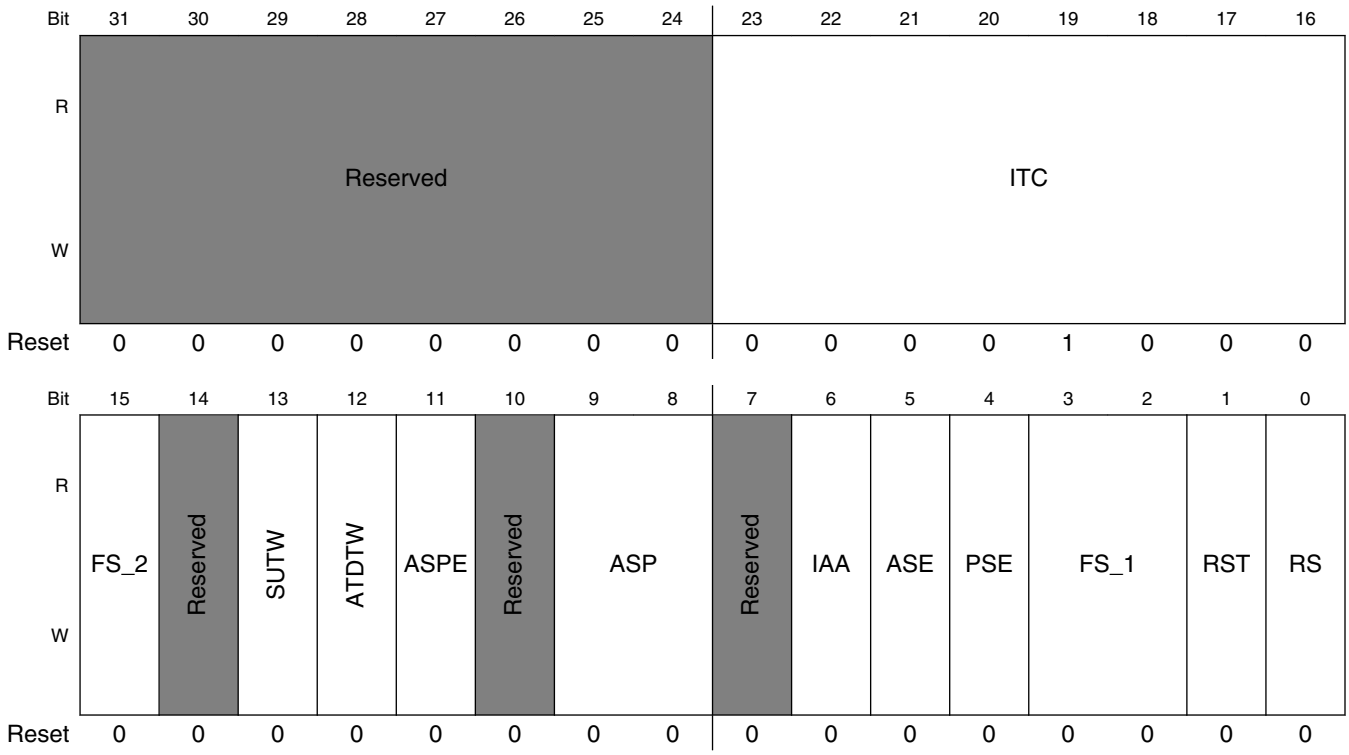
USB_*n*DCCPARAMS field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 HC	Host Capable When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5 -	This field is reserved. Reserved
DEN	Device Endpoint Number This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0 - 15.

54.6.18 USB Command Register (USB_*n*USBCMD)

The Command Register indicates the command to be executed by the serial bus host/device controller. Writing to the register causes a command to be executed.

Address: 218_4000h base + 140h offset + (512d × i), where i=0d to 1d



USB_*n*USBCMD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ITC	Interrupt Threshold Control -Read/Write. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 0x00 Immediate (no threshold) 0x01 1 micro-frame 0x02 2 micro-frames 0x04 4 micro-frames 0x08 8 micro-frames 0x10 16 micro-frames

Table continues on the next page...

USB_*n*USBCMD field descriptions (continued)

Field	Description
	0x20 32 micro-frames 0x40 64 micro-frames
15 FS_2	See also bits 3-2 Frame List Size - (Read/Write or Read Only). [host mode only] This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. **NOTE:** This field is made up from USBCMD bits 15, 3 and 2. Value Meaning 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes)
14 -	This field is reserved. Reserved
13 SUTW	Setup TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (SLOM bit in USB core register n_USBMODE, see USB Device Mode (USB_*n*USBMODE)) then there is a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software. This bit would also be cleared by hardware when a hazard detected.
12 ATDTW	Add dTD TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit would also be cleared by hardware when state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.
11 ASPE	Asynchronous Schedule Park Mode Enable - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. **NOTE:** ASPE bit reset value: '0b' for OTG controller .
10 -	This field is reserved. Reserved
9–8 ASP	Asynchronous Schedule Park Mode Count - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is Read-Only. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the

Table continues on the next page...

USB_*n*USBCMD field descriptions (continued)

Field	Description
	Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when *Park Mode Enable* is a one as this will result in undefined behavior. This field is set to 3h in all controller core.
7 -	This field is reserved. Reserved
6 IAA	Interrupt on Async Advance Doorbell - Read/Write. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable - Read/Write. Default 0b. This bit controls whether the host controller skips processing the Asynchronous Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
4 PSE	Periodic Schedule Enable- Read/Write. Default 0b. This bit controls whether the host controller skips processing the Periodic Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the Periodic Schedule.
3–2 FS_1	See description at bit 15
1 RST	Controller Reset (RESET) - Read/Write. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host operation mode: When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device operation mode:

Table continues on the next page...

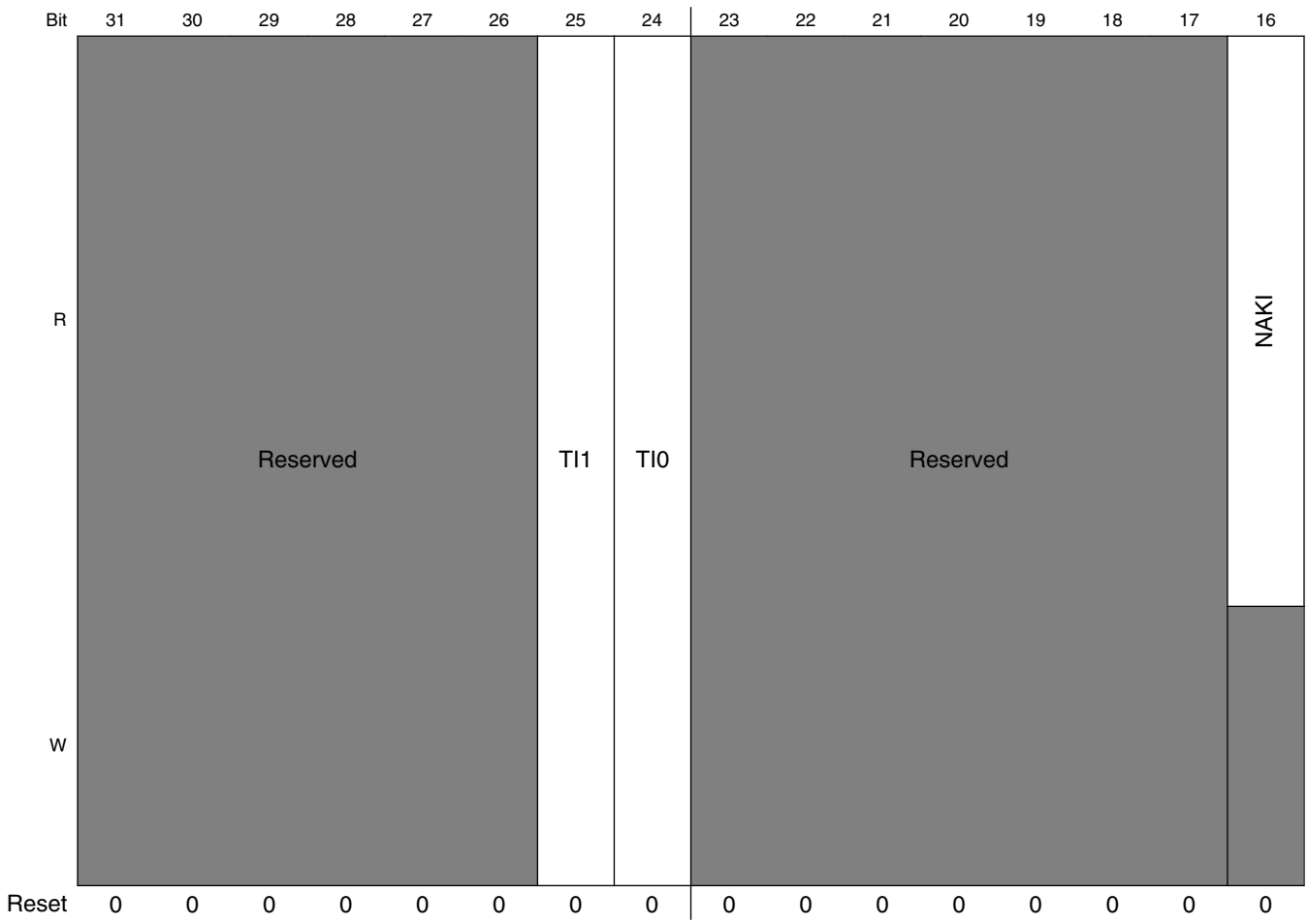
USB_*n*USBCMD field descriptions (continued)

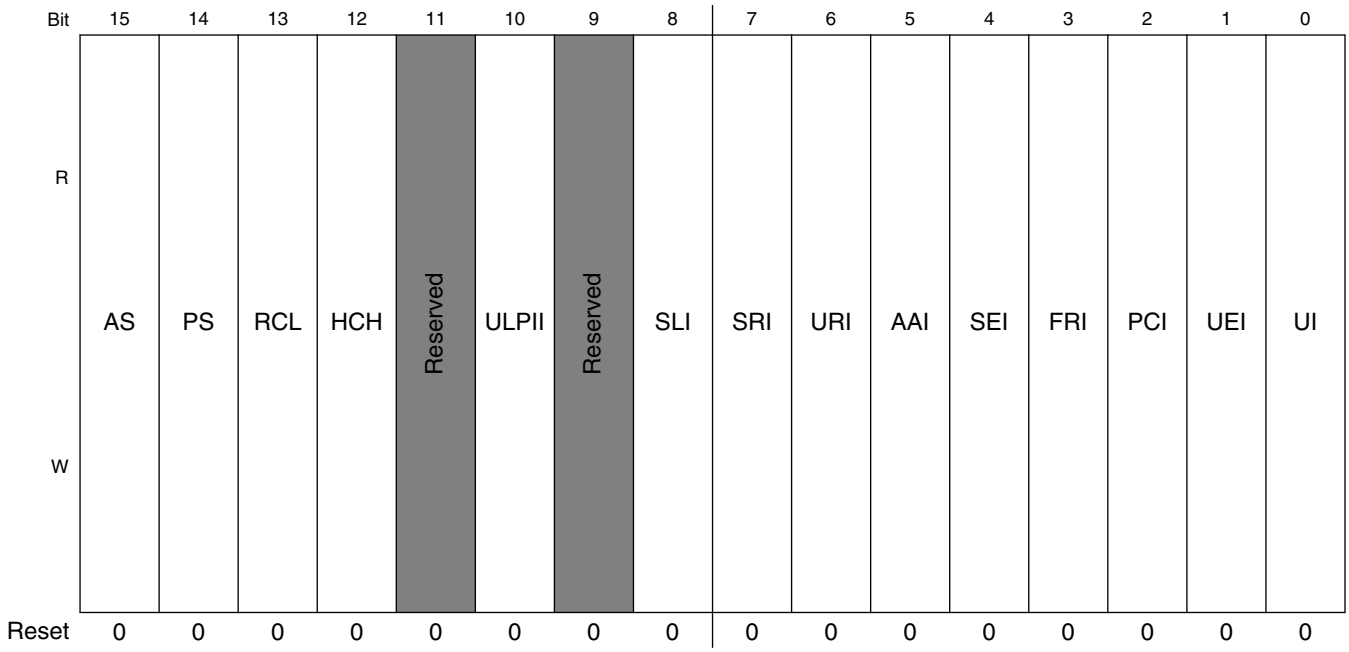
Field	Description
	When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, because the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS) - Read/Write. Default 0b. 1=Run. 0=Stop. Host operation mode: When set to '1b', the Controller proceeds with the execution of the schedule. The Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device operation mode: Writing a one to this bit will cause the controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Writing a 0 to this will cause a detach event.

54.6.19 USB Status Register (USB_*n*USBSTS)

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus.

Address: 218_4000h base + 144h offset + (512d × i), where i=0d to 1d





USB_*n*USBSTS field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TI1	General Purpose Timer Interrupt 1(GPTINT1)--R/WC. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero, writing a one to this bit will clear it.
24 TI0	General Purpose Timer Interrupt 0(GPTINT0)--R/WC. This bit is set when the counter in the GPTIMER0CTRL register transitions to zero, writing a one to this bit clears it.
23–17 -	This field is reserved. Reserved
16 NAKI	NAK Interrupt Bit--RO. This bit is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when all Enabled TX/RX Endpoint NAK bits are cleared.
15 AS	Asynchronous Schedule Status - Read Only. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
14 PS	Periodic Schedule Status - Read Only.

Table continues on the next page...

USB_*n*USBSTS field descriptions (continued)

Field	Description
	This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
13 RCL	Reclamation - Read Only. This is a read-only status bit used to detect an empty asynchronous schedule. Only used in the host operation mode.
12 HCH	HCHalted - Read Only. This bit is a zero whenever the Run/Stop bit is a one. The Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Controller hardware (for example, an internal error). Only used in the host operation mode. Default value is '0b' for OTG core. This is because OTG core is not operating as host in default. Please see CM bit in USB_n_USBMODE register. **NOTE:** HCH bit reset value: '0b' for OTG controller core.
11 -	This field is reserved. Reserved
10 ULPII	ULPI Interrupt - R/WC. This bit will be set '1b' by hardware when there is an event completion in ULPI viewport. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLI	DCSuspend - R/WC. When a controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used in device operation mode.
7 SRI	SOF Received - R/WC. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Because the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it.
6 URI	USB Reset Received - R/WC. When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device operation mode.

Table continues on the next page...

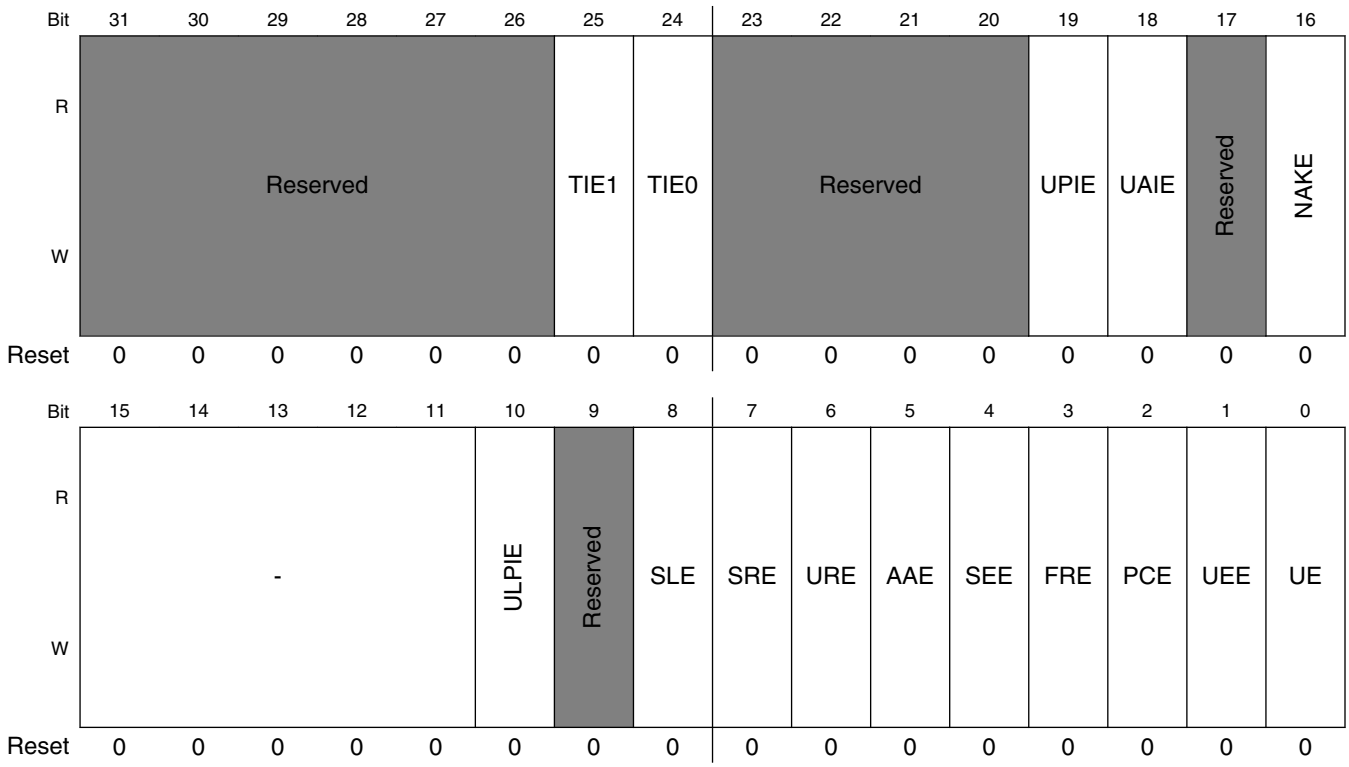
USB_*n*USBSTS field descriptions (continued)

Field	Description
5 AAI	Interrupt on Async Advance - R/WC. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the n_USBCMD register. This status bit indicates the assertion of that interrupt source. Only used in host operation mode.
4 SEI	System Error- R/WC. This bit is will be set to '1b' when an Error response is seen to a read on the system interface.
3 FRI	Frame List Rollover - R/WC. The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USB_n_USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used in host operation mode.
2 PCI	Port Change Detect - R/WC. The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/ Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively.
1 UEI	USB Error Interrupt (USBERRINT) - R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT) - R/WC. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

54.6.20 Interrupt Enable Register (USB_*n*USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt source is active. The USB Status register (n_USBSTS) still shows interrupt sources even if they are disabled by the n_USBINTR register, allowing polling of interrupt events by the software.

Address: 218_4000h base + 148h offset + (512d × i), where i=0d to 1d



USB_*n*USBINTR field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TIE1	General Purpose Timer #1 Interrupt Enable When this bit is one and the TI1 bit in n_USBSTS register is a one the controller will issue an interrupt.
24 TIE0	General Purpose Timer #0 Interrupt Enable When this bit is one and the TI0 bit in n_USBSTS register is a one the controller will issue an interrupt.
23–20 -	This field is reserved. Reserved
19 UPIE	USB Host Periodic Interrupt Enable

Table continues on the next page...

USB_*n*USBINTR field descriptions (continued)

Field	Description
	When this bit is one, and the UPI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
18 UAIE	USB Host Asynchronous Interrupt Enable When this bit is one, and the UAI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
17 -	This field is reserved. Reserved
16 NAKE	NAK Interrupt Enable When this bit is one and the NAKI bit in n_USBSTS register is a one the controller will issue an interrupt.
15–11 -	These bits are reserved and should be set to zero.
10 ULPIE	ULPI Interrupt Enable When this bit is one and the UPLII bit in n_USBSTS register is a one the controller will issue an interrupt. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLE	Sleep Interrupt Enable When this bit is one and the SLI bit in n_n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
7 SRE	SOF Received Interrupt Enable When this bit is one and the SRI bit in n_USBSTS register is a one the controller will issue an interrupt.
6 URE	USB Reset Interrupt Enable When this bit is one and the URI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
5 AAE	Async Advance Interrupt Enable When this bit is one and the AAI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
4 SEE	System Error Interrupt Enable When this bit is one and the SEI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
3 FRE	Frame List Rollover Interrupt Enable When this bit is one and the FRI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
2 PCE	Port Change Detect Interrupt Enable When this bit is one and the PCI bit in n_USBSTS register is a one the controller will issue an interrupt.
1 UEE	USB Error Interrupt Enable When this bit is one and the UEI bit in n_USBSTS register is a one the controller will issue an interrupt.
0 UE	USB Interrupt Enable When this bit is one and the UI bit in n_USBSTS register is a one the controller will issue an interrupt.

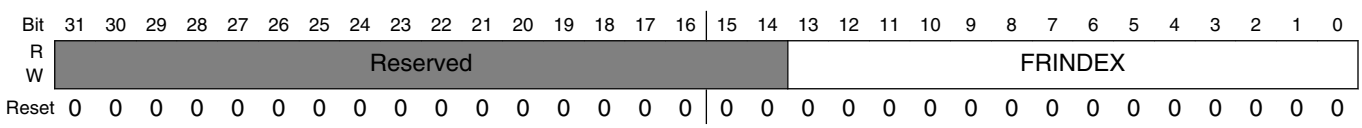
54.6.21 USB Frame Index (USB_*n*FRINDEX)

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the n_USBCMD register.

This register must be written as a DWord. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be increment (that is, SOF for 125 us micro-frame.).

Address: 218_4000h base + 14Ch offset + (512d × i), where i=0d to 1d



USB_*n*FRINDEX field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
FRINDEX	Frame Index. The value, in this register, increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe. 000 (1024) 12 001 (512) 11

Table continues on the next page...

USB_*n*FRINDEX field descriptions (continued)

Field	Description
	010 (256) 10 011 (128) 9 100 (64) 8 101 (32) 7 110 (16) 6 111 (8) 5

54.6.22 Frame List Base Address (USB_*n*PERIODICLISTBASE)

Host Controller only

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (USB_n_FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Address: 218_4000h base + 154h offset + (512d × i), where i=0d to 1d

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

USB_*n*PERIODICLISTBASE field descriptions

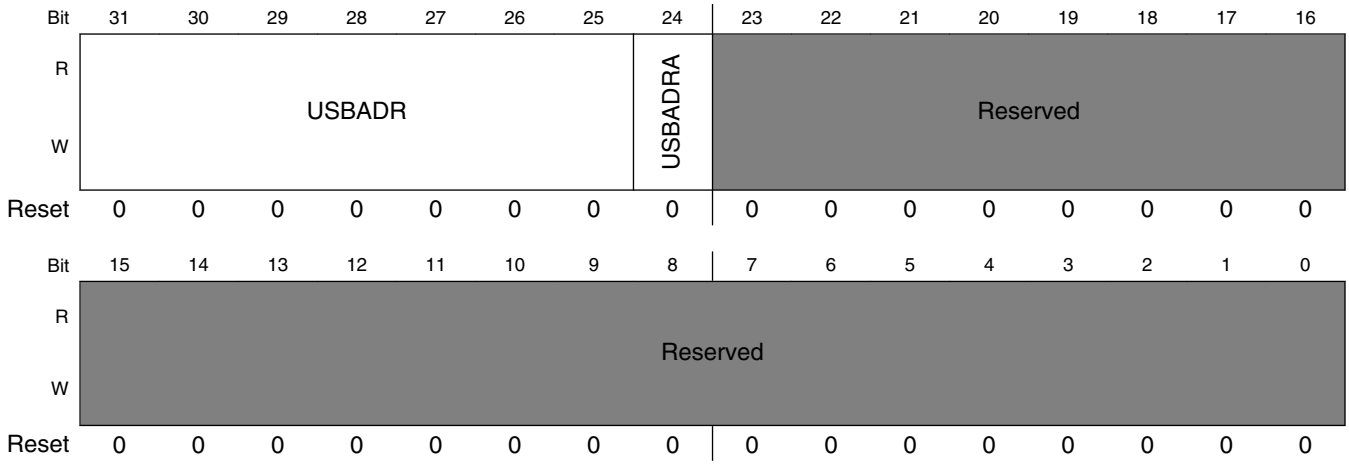
Field	Description
31–12 BASEADR	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
-	This field is reserved. Reserved

54.6.23 Device Address (USB_*n*DEVICEADDR)

Device Controller only

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

Address: 218_4000h base + 154h offset + (512d × i), where i=0d to 1d



USB_*n*DEVICEADDR field descriptions

Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24 USBADRA	Device Address Advance. Default=0. When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a '1' at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). **NOTE:** After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
-	This field is reserved. Reserved

54.6.24 Next Asynch. Address (USB_*n*ASYNCLISTADDR)

Host Controller only

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Address: 218_4000h base + 158h offset + (512d × i), where i=0d to 1d

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

USB_*n*ASYNCLISTADDR field descriptions

Field	Description
31–5 ASYBASE	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). Only used by the host controller.
-	This field is reserved. Reserved

54.6.25 Endpoint List Address (USB_*n*ENDPTLISTADDR)

Device Controller only

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

Address: 218_4000h base + 158h offset + (512d × i), where i=0d to 1d

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

USB_*n*ENDPTLISTADDR field descriptions

Field	Description
31–11 EPBASE	Endpoint List Pointer(Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Head (QH) (that is, one queue head per endpoint & direction).
-	This field is reserved. Reserved

54.6.26 Programmable Burst Size (USB_*n*BURSTSIZE)

This register is used to control the burst size used during data movement on the AHB master interface. This register is ignored if AHBBRST bits in SBUSCFG register is non-zero value.

Address: 218_4000h base + 160h offset + (512d × i), where i=0d to 1d

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

USB_*n*BURSTSIZE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16–8 TXPBURST	Programmable TX Burst Size. Default value is determined by TXBURST bits in n_HWTXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
RXPBURST	Programmable RX Burst Size. Default value is determined by TXBURST bits in n_HWRXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

54.6.27 TX FIFO Fill Tuning (USB_*n*TXFILLTUNING)

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

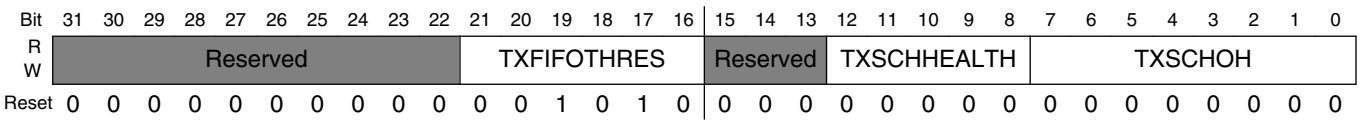
Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a

mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the n_TSCHHEALTH (T_{ff}) described below.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: 218_4000h base + 164h offset + (512d × i), where i=0d to 1d



USB_*n*TXFILLTUNING field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 TXFIFOTHRES	FIFO Burst Threshold. (Read/Write) This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USB_n_USBMODE register is set. Default value is '0Ah for OTG controller core.
15–13 -	This field is reserved. Reserved
12–8 TXSCHHEALTH	Scheduler Health Counter. (Read/Write To Clear) This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31. Default value is '00h' for OTG controller core.
TXSCHOH	Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode. Default value is '08h' for OTG controller core.

54.6.28 Endpoint NAK (USB_*n*ENDPTNAK)

Address: 218_4000h base + 178h offset + (512d × i), where i=0d to 1d

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

USB_*n*ENDPTNAK field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 EPTN	TX Endpoint NAK - R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
EPRN	RX Endpoint NAK - R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7

54.6.29 Endpoint NAK Enable (USB_*n*ENDPTNAKEN)

Address: 218_4000h base + 17Ch offset + (512d × i), where i=0d to 1d

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

USB_*n*ENDPTNAKEN field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 EPTNE	TX Endpoint NAK Enable - R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
EPRNE	RX Endpoint NAK Enable - R/W.

Table continues on the next page...

USB_*n*ENDPTNAKEN field descriptions (continued)

Field	Description
	Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7

54.6.30 Configure Flag Register (USB_*n*CONFIGFLAG)

Address: 218_4000h base + 180h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

USB_*n*CONFIGFLAG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 CF	Configure Flag Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. 0 Port routing control logic default-routes each port to an implementation dependent classic host controller. 1 Port routing control logic default-routes all ports to this host controller.

54.6.31 Port Status & Control (USB_*n*PORTSC1)

Host Controller

A host controller could implement one to eight port status and control registers. The number is determined by N_PORTs bits in HWSPARAMs register (please see Host Controller Structural Parameters (USB_*n*HCSPARAMS)). Software could read this parameter register to determine how many ports need service.

All controller cores on this product are Single-Port, so there is only one port status and control register for each controller core.

This register is only reset by power on reset or controller core reset. The initial conditions of a port are:

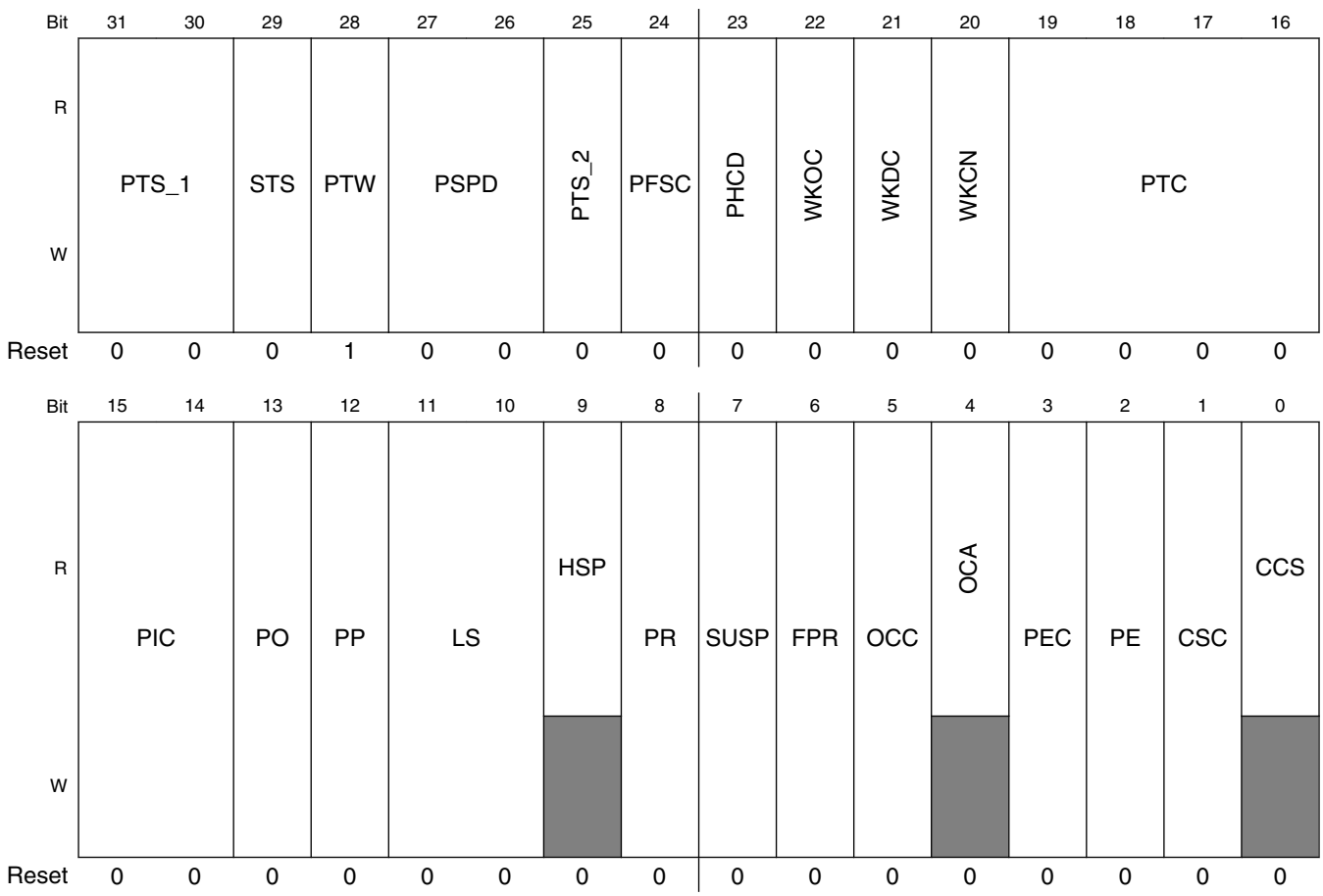
- No device connected
- Port disabled

If the port supports power control, this state remains until port power is supplied (by software).

Device Controller

A controller operating in device mode has only port register one (PORTSC1) and it does not support power control in that mode. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Address: 218_4000h base + 184h offset + (512d × i), where i=0d to 1d



USB_*n*PORTSC1 field descriptions

Field	Description
31–30 PTS_1	**NOTE:** All USB port interface modes are listed in this field description, but not all are supported. For detail feature of each controller core, please see Features . The behaviour is unknown when unsupported interface mode is selected.
29 STS	Serial Transceiver Select 1 Serial Interface Engine is selected 0 Parallel Interface signals is selected Serial Interface Engine can be used in combination with UTMI+/ULPI physical interface to provide FS/LS signaling instead of the parallel interface signals. When this bit is set '1b', serial interface engine will be used instead of parallel interface signals. This bit has no effect unless PTS bits is set to select UTMI+/ULPI interface. The Serial/USB1.1 PHY/IC-USB will use the serial interface engine for FS/LS signaling regardless of this bit value.
28 PTW	Parallel Transceiver Width This bit has no effect if serial interface engine is used. For OTG1/OTG2 core, it is Read-Only. Reset value is '1b'. 0 Select the 8-bit UTMI interface [60MHz] 1 Select the 16-bit UTMI interface [30MHz]
27–26 PSPD	Port Speed - Read Only. This register field indicates the speed at which the port is operating. 00 Full Speed 01 Low Speed 10 High Speed 11 Undefined
25 PTS_2	See description at bits 31-30
24 PFSC	Port Force Full Speed Connect - Read/Write. Default = 0b. When this bit is set to '1b', the port will be forced to only connect at Full Speed, It disables the chirp sequence that allows the port to identify itself as High Speed. 1 Forced to full speed 0 Normal operation
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD) - Read/Write. Default = 0b. When this bit is set to '1b', the PHY clock is disabled. Reading this bit will indicate the status of the PHY clock. **NOTE:** The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend when the device is not running (USBCMD Run/Stop=0b) or the host has signalled suspend (PORTSC1 SUSPEND=1b). PHY Low power suspend will be cleared automatically when the host initials resume. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software.

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	1 Disable PHY clock 0 Enable PHY clock
22 WKOC	Wake on Over-current Enable (WKOC_E) - Read/Write. Default = 0b. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero.
21 WKDC	Wake on Disconnect Enable (WKDSCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero or in device mode.
20 WKCN	Wake on Connect Enable (WKCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero or in device mode.
19–16 PTC	Port Test Control - Read/Write. Default = 0000b. Refer to Port Test Mode for the operational model for using these test modes and the USB Specification Revision 2.0, Chapter 7 for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **NOTE:** *Low speed operations are not supported as a peripheral device.* Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000 TEST_MODE_DISABLE 0001 J_STATE 0010 K_STATE 0011 SE0 (host) / NAK (device) 0100 Packet 0101 FORCE_ENABLE_HS 0110 FORCE_ENABLE_FS 0111 FORCE_ENABLE_LS 1000-1111 Reserved
15–14 PIC	Port Indicator Control - Read/Write. Default = Ob. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if *Port Power* is zero. Bit Value Meaning 00 Port indicators are off 01 Amber 10 Green 11 Undefined
13 PO	Port Owner-Read/Write. Default = 0.

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not supported in all controller cores, therefore this bit will always be 0.
12 PP	Port Power (PP)-Read/Write or Read Only. The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0 1b *Read Only - Host controller does not have port power control switches. Each port is hard-wired to power.* 1 1b/0b - *Read/Write. OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, etc.* When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitional by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in all controller cores (PPC = 1).
11–10 LS	Line Status-Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary. The encoding of the bits are: Bits [11:10] Meaning 00 SE0 10 J-state 01 K-state 11 Undefined
9 HSP	High-Speed Port - Read Only. Default = 0b. When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. **NOTE:** HSP is redundant with PSPD(bit 27, 26) but remained for compatibility.
8 PR	Port Reset - Read/Write or Read Only. Default = 0b. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. *This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.*

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero.
7 SUSP	Suspend - Read/Write or Read Only. Default = 0b. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the *Force Port Resume* bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, *Port enabled* bit is a zero) the results are undefined. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: Read Only. In device mode this bit is a read only status bit.
6 FPR	Force Port Resume -Read/Write. 1= Resume detected/driven on port. 0=No resume (K-state) detected/driven on port. Default = 0. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to one. *This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver.* Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation, clear the bit the port control state switches to HS or FS idle. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. This bit is not-EHCI compatible. In Device mode:

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit wil be cleared because a K-to-J transition detected, the *Port Change Detect* bit in the USBSTS register is also set to one.
5 OCC	Over-current Change-R/WC. Default=0. This bit is set '1b' by hardware when there is a change to Over-current Active. Software can clear this bit by writing a one to this bit position.
4 OCA	Over-current Active-Read Only. Default 0. This bit will automatically transition from one to zero when the over current condition is removed. 1 This port currently has an over-current condition 0 This port does not have an over-current condition.
3 PEC	Port Enable/Disable Change-R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. In Host Mode: For the root hub, this bit is set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero. In Device mode: The device port is always enabled, so this bit is always '0b'.
2 PE	Port Enabled/Disabled-Read/Write. 1=Enable. 0=Disable. Default 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: The device port is always enabled, so this bit is always '1b'.
1 CSC	Connect Status Change-R/WC. 1 =Change in Current Connect Status. 0=No change. Default 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: This bit is undefined in device controller mode.

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
0 CCS	Current Connect Status-Read Only. In Host Mode: 1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the *Connect Status Change* bit (Bit 1) to be set. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached. Default=0. A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

54.6.32 On-The-Go Status & control (USB_*n*OTGSC)

This register is availabe only in OTG controller core. It has four sections:

- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

The status inputs are debounced using a 1 ms time constant. Values on the status inputs that do not persist for more than 1 ms does not cause an update of the status input register, or cause an OTG interrupt.

See also USB Device Mode (USB_*n*USBMODE) register.

Address: 218_4000h base + 1A4h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	DPIE	EN_1MS	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	Reserved	DPIS	STATUS_1MS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	DPS	TOG_1MS	BSE	BSV	ASV	AVV	ID	Reserved		IDPU	DP	OT	Reserved	VC	VD
W																
Reset	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0

USB_*n*OTGSC field descriptions

Field	Description
31 -	This field is reserved. Reserved
30 DPIE	Data Pulse Interrupt Enable
29 EN_1MS	1 millisecond timer Interrupt Enable - Read/Write
28 BSEIE	B Session End Interrupt Enable - Read/Write. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable - Read/Write. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable - Read/Write. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable - Read/Write. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable - Read/Write. Setting this bit enables the USB ID interrupt.
23 -	This field is reserved. Reserved
22 DPIS	Data Pulse Interrupt Status - Read/Write to Clear. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC1(0)[PP] = 0. Software must write a one to clear this bit.
21 STATUS_1MS	1 millisecond timer Interrupt Status - Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status - Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit.
19 BSVIS	B Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold. Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold. Software must write a one to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold on an A device. Software must write a one to clear this bit.
16 IDIS	USB ID Interrupt Status - Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.

Table continues on the next page...

USB_*n*OTGSC field descriptions (continued)

Field	Description
15 -	This field is reserved. Reserved
14 DPS	Data Bus Pulsing Status - Read Only. A '1' indicates data bus pulsing is being detected on the port.
13 TOG_1MS	1 millisecond timer toggle - Read Only. This bit toggles once per millisecond.
12 BSE	B Session End - Read Only. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid - Read Only. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid - Read Only. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid - Read Only. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID - Read Only. 0 = A device, 1 = B device
7–6 -	This field is reserved. Reserved
5 IDPU	ID Pullup - Read/Write This bit provide control over the ID pull-up resistor; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing - Read/Write. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination - Read/Write. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 -	This field is reserved. Reserved
1 VC	VBUS Charge - Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge - Read/Write. Setting this bit causes VBus to discharge through a resistor.

54.6.33 USB Device Mode (USB_*n*USBMODE)

Address: 218_4000h base + 1A8h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

USB_*n*USBMODE field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14–5 -	This field is reserved. Reserved
4 SDIS	Stream Disable Mode. (0 - Inactive [default]; 1 - Active) Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received are responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. **NOTE:** Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TX FIFO Fill Tuning (USB_*n*TXFILLTUNING) and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature.

Table continues on the next page...

USB_*n*USBMODE field descriptions (continued)

Field	Description
	NOTE: The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model . 0 Setup Lockouts On (default); 1 Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USB Command Register (USB_*n*USBCMD) .
2 ES	Endian Select - Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. Bit Meaning 0 Little Endian [Default] 1 Big Endian
CM	Controller Mode - R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the *RESET* bit in the USBCMD register before reprogramming this register. For OTG controller core, reset value is '00b'. 00 Idle [Default for combination host/device] 01 Reserved 10 Device Controller [Default for device only controller] 11 Host Controller [Default for host only controller]

54.6.34 Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT)

Address: 218_4000h base + 1ACh offset + (512d × i), where i=0d to 1d

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

USB_*n*ENDPTSETUPSTAT field descriptions

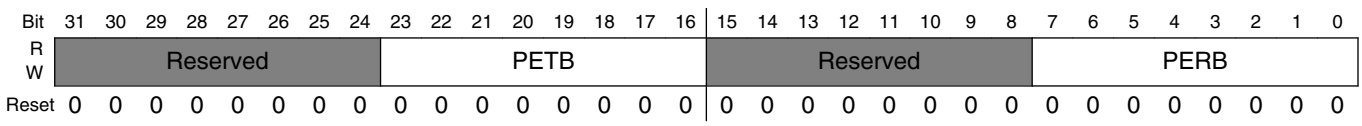
Field	Description
31–16 -	This field is reserved. Reserved
ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See Managing Endpoints in the Device Operational Model. This register is only used in device mode.

54.6.35 Endpoint Prime (USB_*n*ENDPTPRIME)

This register is only used in device mode.

When software sets the prime bit for a given endpoint, the device controller loads the transfer descriptor, pointed to by the queue head, such that the endpoint is ready to transmit or receive when the host sends a request (IN/OUT token). The endpoint will NAK all requests from the host until the endpoint is primed. The controller will automatically re-prime the endpoint with a new transfer descriptor when one is found via the next_dtd pointer of the current transfer descriptor. Hence, the prime bit must only be set by software when a descriptor is added to the queue head.

Address: 218_4000h base + 1B0h offset + (512d × i), where i=0d to 1d



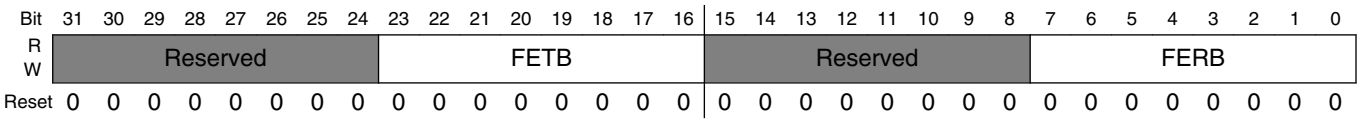
USB_*n*ENDPTPRIME field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 PETB	Prime Endpoint Transmit Buffer - R/WS. For each endpoint a corresponding bit is used to request that a buffer is prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
PERB	Prime Endpoint Receive Buffer - R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PERB[N] - Endpoint #N, N is in 0..7

54.6.36 Endpoint Flush (USB_*n*ENDPTFLUSH)

This register is only used in device mode.

Address: 218_4000h base + 1B4h offset + (512d × i), where i=0d to 1d



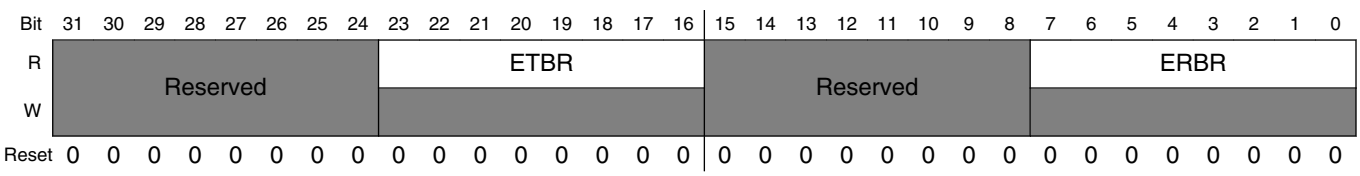
USB_*n*ENDPTFLUSH field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 FETB	Flush Endpoint Transmit Buffer - R/WS. Writing one to a bit(s) in this register causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
FERB	Flush Endpoint Receive Buffer - R/WS. Writing one to a bit(s) causes the assocUOGiated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FERB[N] - Endpoint #N, N is in 0..7\

54.6.37 Endpoint Status (USB_*n*ENDPTSTAT)

This register is only used in device mode.

Address: 218_4000h base + 1B8h offset + (512d × i), where i=0d to 1d



USB_*n*ENDPTSTAT field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ETBR	Endpoint Transmit Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There is always a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
ERBR	Endpoint Receive Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPRIME register. There is always a delay between setting a bit in the ENDPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[N] - Endpoint #N, N is in 0..7

54.6.38 Endpoint Complete (USB_*n*ENDPTCOMPLETE)

This register is only used in device mode.

Address: 218_4000h base + 1BCh offset + (512d × i), where i=0d to 1d

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

USB_*n*ENDPTCOMPLETE field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ETCE	Endpoint Transmit Complete Event - R/WC. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ETCE[N] - Endpoint #N, N is in 0..7

Table continues on the next page...

USB_*n*ENDPTCOMPLETE field descriptions (continued)

Field	Description
15–8 -	This field is reserved. Reserved
ERCE	Endpoint Receive Complete Event - RW/C. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ERCE[N] - Endpoint #N, N is in 0..7

54.6.39 Endpoint Control0 (USB_*n*ENDPTCTRL0)

Every Device implements Endpoint 0 as a control endpoint.

Address: 218_4000h base + 1C0h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	Reserved			TXT		Reserved	TXS
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

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

USB_*n*ENDPTCTRL0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 1 Enabled

Table continues on the next page...

USB_*n*ENDPTCTRL0 field descriptions (continued)

Field	Description
	Endpoint0 is always enabled.
22–20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 - Control Endpoint0 is fixed as a Control End Point.
17 -	This field is reserved. Reserved
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. **NOTE:** There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a newsetup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
6–4 -	This field is reserved. Reserved
3–2 RXT	RX Endpoint Type - Read/Write 00 Control Endpoint0 is fixed as a Control End Point.
1 -	This field is reserved. Reserved
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.

Table continues on the next page...

USB_*n*ENDPTCTRL0 field descriptions (continued)

Field	Description
	NOTE: There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the dcd software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a newsetup has been received by checking the associated endptsetupstat bit.

54.6.40 Endpoint Control 1 (USB_*n*ENDPTCTRL1)

This is endpoint control register for endpoint 1 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1C4h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTCTRL1 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved

Table continues on the next page...

USB_*n*ENDPTCTRL1 field descriptions (continued)

Field	Description
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

54.6.41 Endpoint Control 2 (USB_*n*ENDPTCTRL2)

This is endpoint control register for endpoint 2 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1C8h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTCTRL2 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.

Table continues on the next page...

USB_*n*ENDPTCTRL2 field descriptions (continued)

Field	Description
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence

Table continues on the next page...

USB_*n*ENDPTCTRL2 field descriptions (continued)

Field	Description
	Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

54.6.42 Endpoint Control 3 (USB_*n*ENDPTCTRL3)

This is endpoint control register for endpoint 3 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control

causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1CCh offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTCTRL3 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved

Table continues on the next page...

USB_*n*ENDPTCTRL3 field descriptions (continued)

Field	Description
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control

Table continues on the next page...

USB_*n*ENDPTCTRL3 field descriptions (continued)

Field	Description
	01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

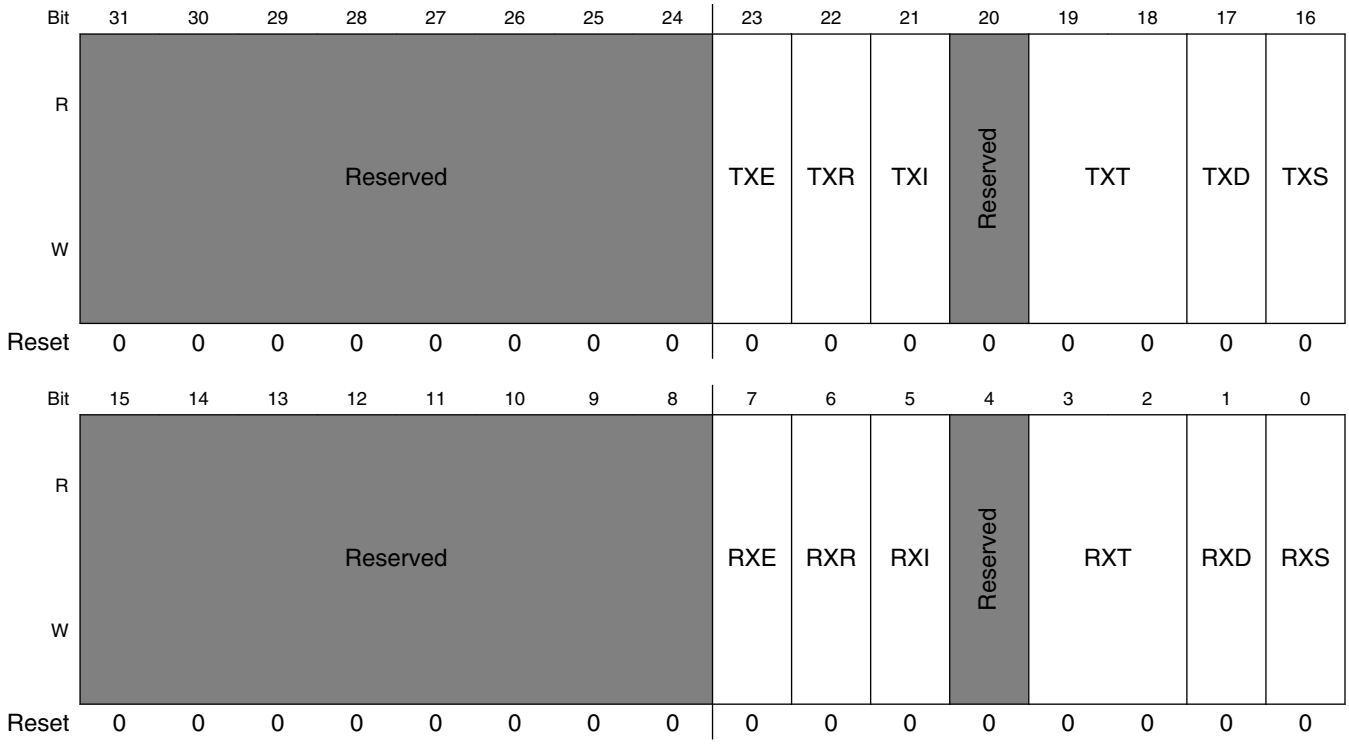
54.6.43 Endpoint Control 4 (USB_*n*ENDPTCTRL4)

This is endpoint control register for endpoint 4 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1D0h offset + (512d × i), where i=0d to 1d



USB_*n*ENDPTCTRL4 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous

Table continues on the next page...

USB_*n*ENDPTCTRL4 field descriptions (continued)

Field	Description
	10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt

Table continues on the next page...

USB_*n*ENDPTCTRL4 field descriptions (continued)

Field	Description
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

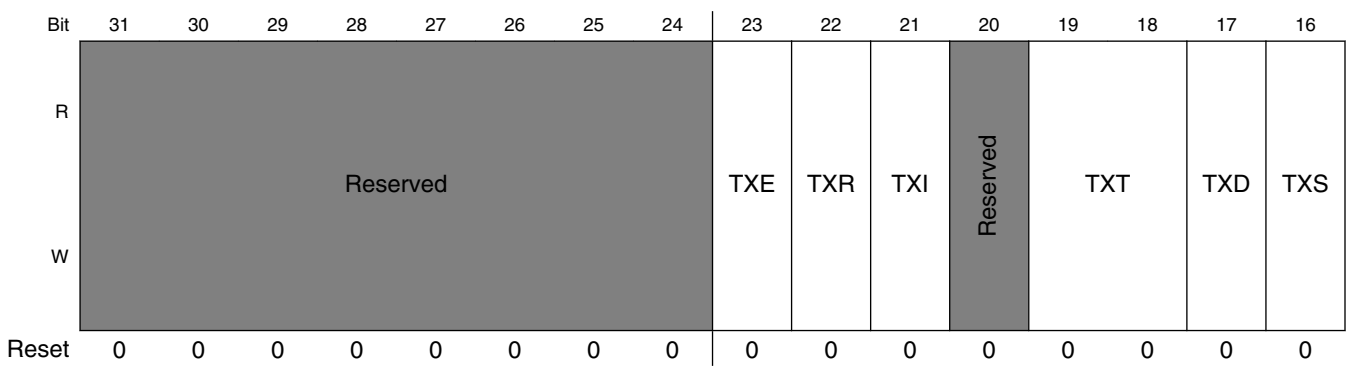
54.6.44 Endpoint Control 5 (USB_*n*ENDPTCTRL5)

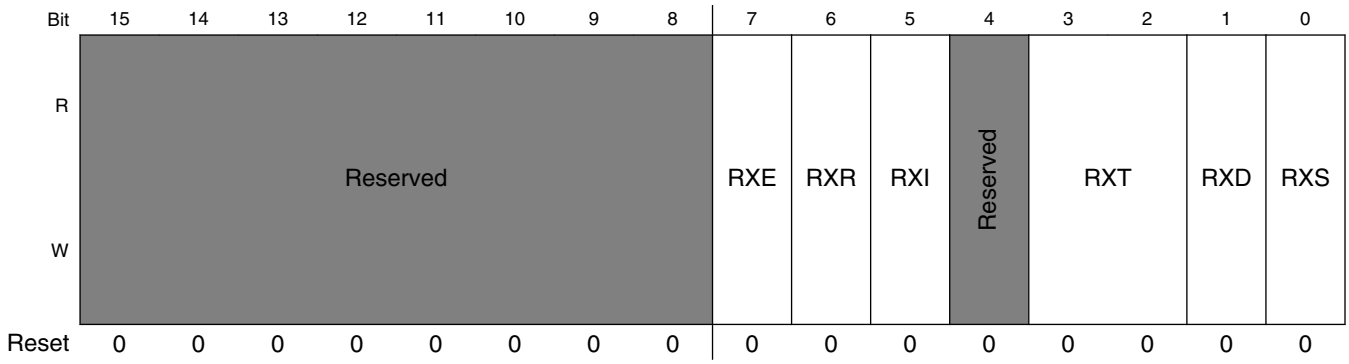
This is endpoint control register for endpoint 5 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1D4h offset + (512d × i), where i=0d to 1d





USB_*n*ENDPTCTRL5 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL5 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL5 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

54.6.45 Endpoint Control 6 (USB_*n*ENDPTCTRL6)

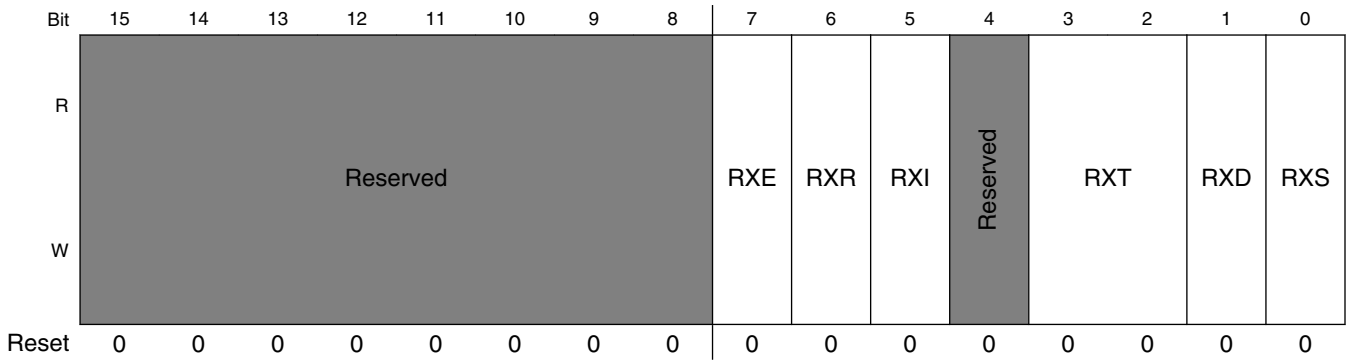
This is endpoint control register for endpoint 6 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1D8h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USB_*n*ENDPTCTRL6 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL6 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL6 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

54.6.46 Endpoint Control 7 (USB_*n*ENDPTCTRL7)

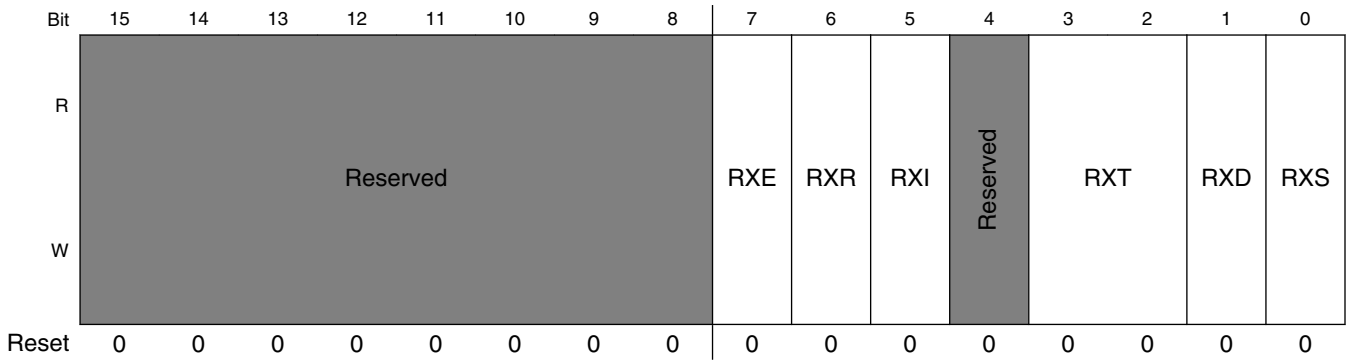
This is endpoint control register for endpoint 7 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1DCh offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USB_*n*ENDPTCTRL7 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL7 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL7 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

Chapter 55
Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

55.1 USB PHY Overview

The chip contains 2 integrated USB 2.0 PHY macrocells capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s.

The integrated PHY provides a standard UTM interface. The USB_*n*_DN and USB_*n*_DP pins connect directly to a USB connector.

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 PHY.

55.2 Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 55-1.

55.2.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection.

The PLL supplies a 120 MHz signal to all of the digital logic. The UTMI block does a final divide-by-four to develop the 30 MHz clock used in the interface.

55.2.2 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake.

In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed or 1.5 Mbit for low-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the low-speed (LS), full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

55.2.3 Digital Receiver

The digital receiver receives the raw serial bitstream from the low speed (LS) differential transceiver, full speed (FS) differential transceiver, and a 9X, 480 MHz sampled data from the high speed (HS) differential transceiver.

As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480 Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

55.2.4 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module

, as shown in the figure below and described further in this section.

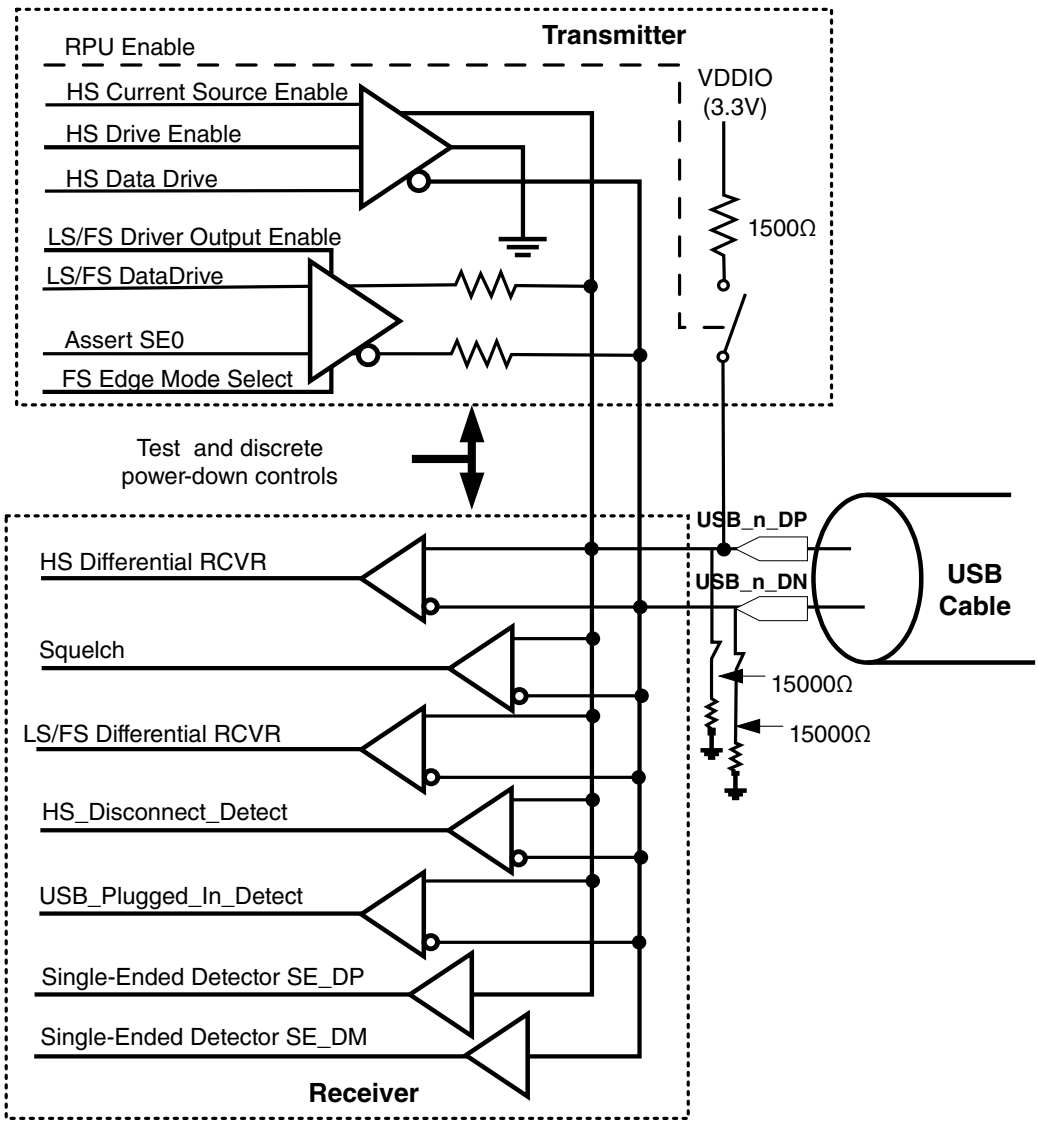


Figure 55-1. USB 2.0 PHY Analog Transceiver Block Diagram

55.2.4.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is a one if the differential signal is greater than a 0-V threshold.

Otherwise, its output is 0. Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual 45Ω terminations found on each pin of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

55.2.4.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator.

Its output is 1, if the differential magnitude is less than a nominal 100 mV threshold. Otherwise, its output is 0.

Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

55.2.4.3 LS/FS Differential Receiver

The low-speed/full-speed differential receiver is both a differential analog receiver and threshold comparator.

The crossover voltage falls between 1.3 V and 2.0 V. Its output is 1, when the USB_*n*_DP line is above the crossover point and the USB_*n*_DN line is below the crossover point. The digital receiver section decodes the receiver data into J or K state according to the speed.

55.2.4.4 HS Disconnect Detector

It is a differential analog receiver and threshold comparator. It outputs high when differential magnitude is greater than a nominal 575-mV threshold. Otherwise, it outputs low.

55.2.4.5 USB Plugged-In Detector

The USB plugged-in detector looks for both USB_*n*_DP and USB_*n*_DN to be high. There is a pair of large on-chip pullup resistors (200 KΩ) that hold both USB_*n*_DP and USB_*n*_DN high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case.

When operating in device mode, the upstream port in host/hub interface contains a 15 KΩ pulldown resistor which could easily override the 200 KΩ pullup resistor. When plugged in, at least one signal in the pair will be low, which will force the plugged-in detector's output high.

55.2.4.6 Single-Ended USB_DP Receiver

The single-ended USB_*n*_DP receiver output is high whenever the USB_*n*_DP input is above its nominal 1.8 V threshold.

55.2.4.7 Single-Ended USB_DN Receiver

The single-ended USB_*n*_DN receiver output is high whenever the USB_*n*_DN input is above its nominal 1.8 V threshold.

55.2.4.8 9X Oversample Module

The 9X oversample module uses nine identically spaced phases of the 480 MHz clock to sample a high speed bit data. The squelch signal is sampled only 1X.

55.2.5 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5 KΩ pullup resistor.

See Figure 55-1.

55.2.5.1 Switchable High-Speed 45Ω Termination Resistors

High-speed current mode differential signaling requires good 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable.

Because each signal is parallel terminated with 45 Ω at each end, each driver sees a 22.5 Ω load. This load impedance is much too low for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure

55-2. The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45Ω terminator on the USB_*n*_DP signal.

55.2.5.2 Low-Speed/Full-Speed Differential Driver

The low-speed/full-speed differential drivers are essentially low-impedance pulldown devices that are switched in a differential mode for low-speed or full-speed signaling, that is, either one or the other device is turned on to signal the "J" state or the "K" state.

55.2.5.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78 mA current from the constant current source (Iref) and essentially steers it down either the USB_DP signal or the USB_DN signal or alternatively to ground.

This current will produce approximately a 400 mV drop across the 22.5 Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage-band-gap (Vbg) circuit. The Iref, Ibias, and V to I circuits are shared with the integrated battery charger.

55.2.5.4 Switchable 1.5KΩ USB_DP Pullup Resistor

This product contains a switchable 1.5 KΩ pullup resistor on the USB_*n*_DP signal.

This resistor is switched on to indicate to the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until the processor software enables the announcement of a full-speed device.

55.2.5.5 Switchable 15KΩ USB_DP Pulldown Resistor

This product contains a switchable 15 KΩ pulldown resistor on both USB_*n*_DP and USB_*n*_DN signals. This is used in host mode to indicate to the device controller that a host is present.

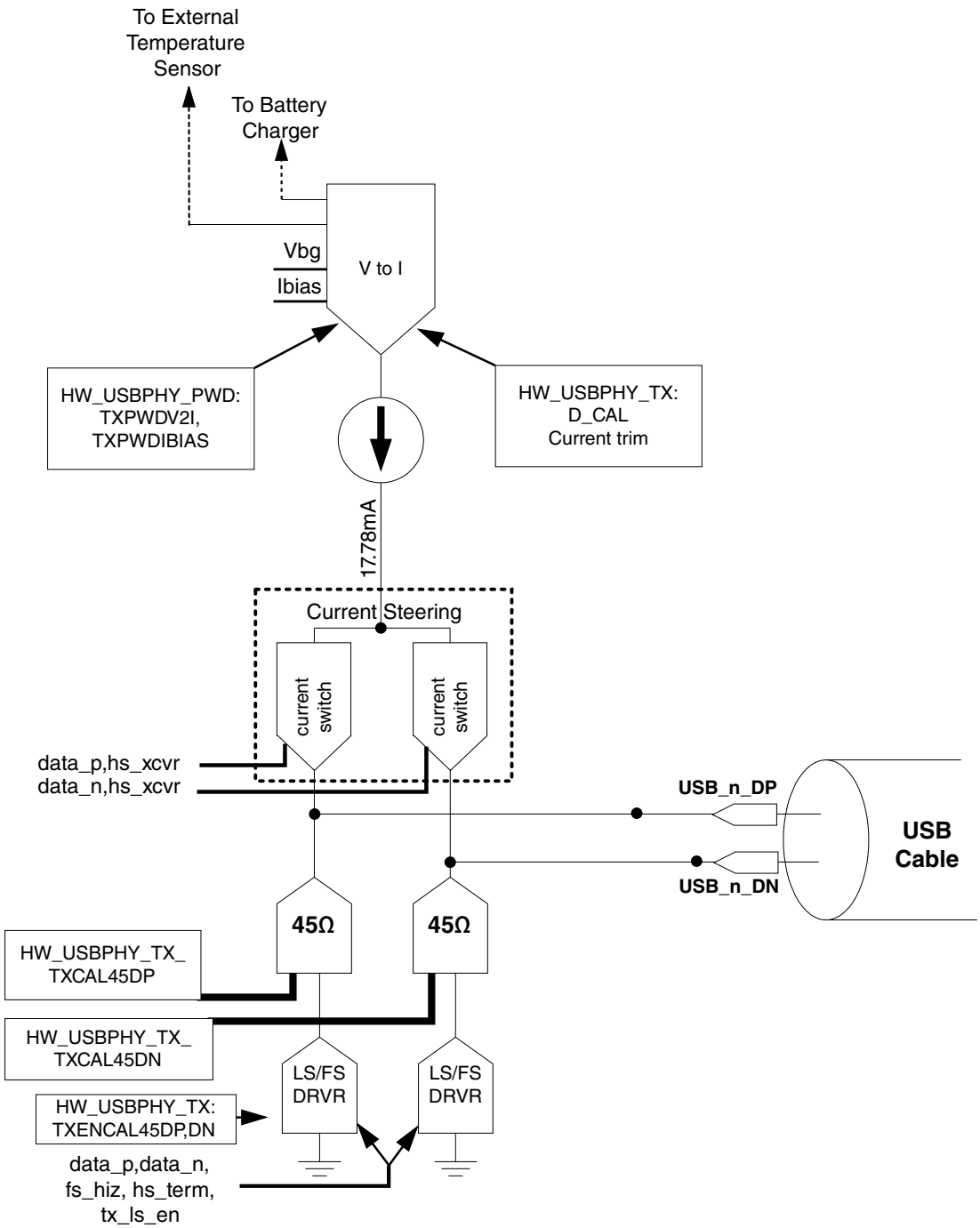


Figure 55-2. USB 2.0 PHY Transmitter Block Diagram

55.2.6 Recommended Register Configuration for USB Certification

The register settings in this section are recommended for passing USB certification.

The following settings lower the J/K levels to certifiable limits:

HW_USBPHY_TX_TXCAL45DP = 0x0
HW_USBPHY_TX_TXCAL45DN = 0x0
HW_USBPHY_TX_D_CAL = 0x7

55.2.7 Charger detection

The USB charger detector is a block that detects whether the upstream-facing device is connected to a down-stream facing charger, either a dedicated USB charger or a host charger.

The USB charger detector is comprised of two sub-blocks, namely the USB data-pin contact detector and the charger detector.

This section details those two sub-blocks and gives the software flow of USB charger detection. Finally, this chapter discusses the detection of a USB charger in case of a dead battery.

55.2.7.1 Charger detect control table

Before we dive into the details of the detectors, we show the logic table of the control signals to give the user an overall picture of the charger detector.

Table 55-1. Charger detection control table

EN_B	CHK_CHRG_B	CHK_CONTACT	Data pin contact detector	Charger detector
0	1	1	Enabled	Disabled
0	0	x(don't care)	Disabled	Enabled
1	x	x(don't care)	Disabled	Disabled

55.2.7.2 Data pin contact detector

According to Battery Charging Specification (rev 1.2), USB plugs and receptacles are designed such that when the plug is inserted into the receptacle, the power pins make contact before the data pins make contact. Therefore, there is inevitably a time interval during which USB_*n*_VBUS has been observed by the device while the USB_*n*_DP and USB_*n*_DN pins are not still pending for contact. The USB data pin contact detector is designed to give the software an indication of the contact of the data pins.

To enable the USB data pin contact detector, the user should set the CHK_CONTACT bit of the USB1_CHRG_DETECT register to 1 and monitor the PLUG_CONTACT bit status of the USB1_CHRG_DETECT_STAT register. If PLUG_CONTACT is 1, then it indicates that the data pins have make good contacts, otherwise the user should continue to wait until this bit is set.

According to Table 1, it should be noted that the data pin contact detector only works when EN_B=0 and CHK_CHRG_B=1, both bit being of the USB1_CHRG_DETECT register.

55.2.7.3 Charger detector

Once the data pins make contact, the user should enable the charger detector by clearing the CHK_CHRG_B bit that is low-active. Then the user should wait for 40ms and then check the status bit of CHRG_DETECTED in register hw_anadig_usb1_chrg_det_stat. CHRG_DETECTED=1 means that the device is connected to a charger, either a dedicated charger or a charging downstream port (or equivalently called a host charger, or charging host). To further differentiate between a host charger and a dedicated charger, the user is suggested to pull up USB_*n*_DP signaling a connect event to the host. Then the user should monitor the USB_*n*_DN line status. If USB_*n*_DN=1, then the charger is a dedicated charger; if USB_*n*_DN=0, then it is a host charger.

55.2.7.4 Charger detection software flow

Upon seeing VBUS, the software should follow the software flow for the charger detection process. The flow chart mentions the "enable the vdd3p0 current limiter". Please refer to the power chapter for details.

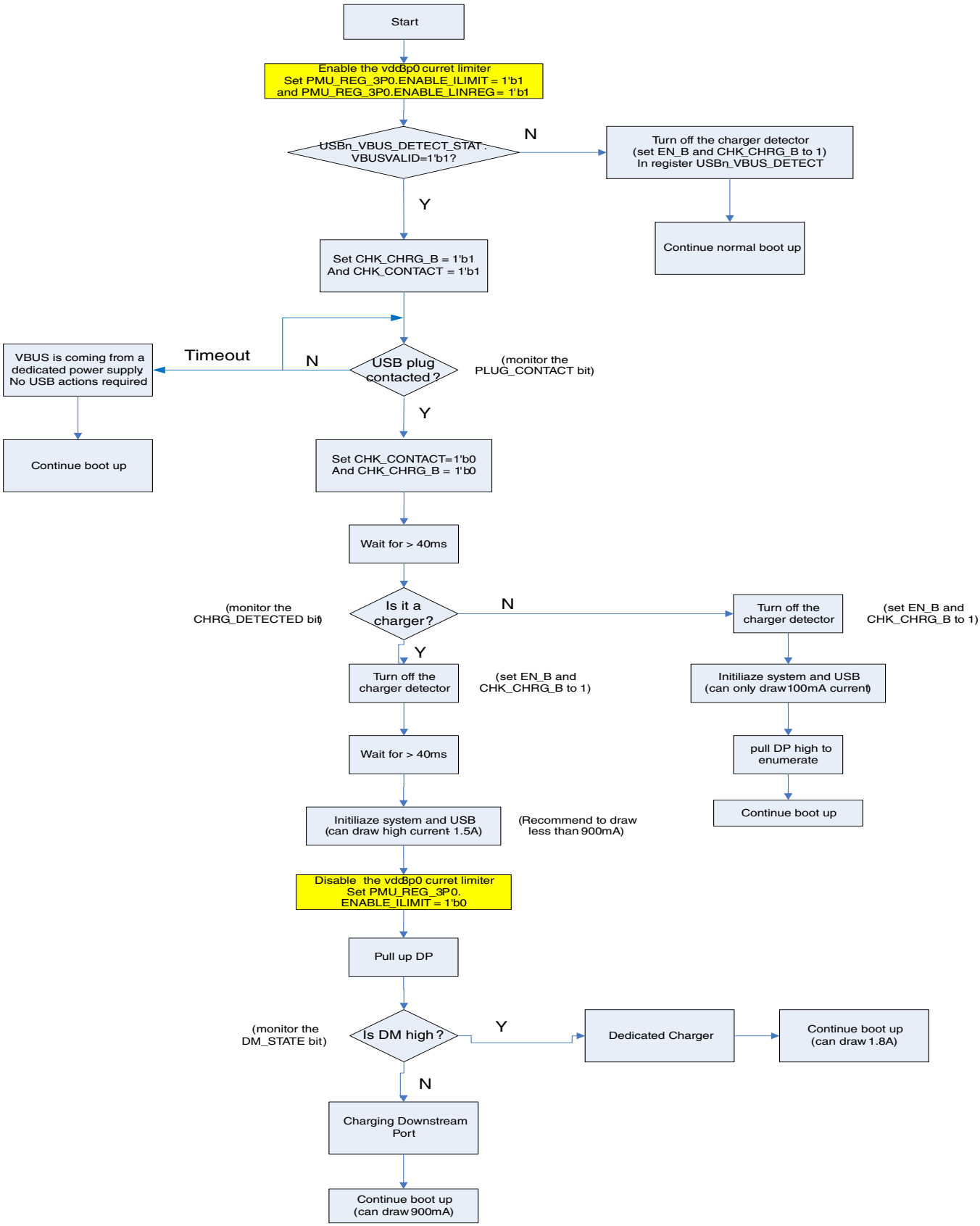


Figure 55-3. USBPHY Charger Detection Software Flow

55.2.7.5 Dead Battery Protect

All the descriptions above are based on the assumption that all the power supplies have been on when the device is plugged into a remote host (or charger). However, there are cases when the local battery of the portable device has been so depleted that the system could not be turned on. In such scenarios the user may prefer a method of signaling the external power management unit (PMIC) the existence of the USB charger to draw a current larger than 100mA from the remote host to speed up system boot up or battery charging. The charger detector indeed supports this function.

When we have a fully depleted battery, all the power supplies might be off. Upon insertion of the 5V, the supplies are brought up by the external PMIC and the internal regulators. Due to the 100mA inrush current limit of the USB spec, we cannot draw larger than 100mA current which might be a limit for system boot-up. Since by default, EN_B=0, CHK_CHRG_B=0 and CHK_CONTACT=1, the usb charger detector is automatically enabled without any software operation needed and it can signal the external PMIC the existence of a USB charger through the open-drain output pin USB_OTG_CHD_B. This pin should be pulled up to an external voltage that is acceptable to the PMIC. If this signal is low, then the PMIC can get that the device is connected to a charger. In this case, the PMIC can draw more than 100mA current from the USB.

It should be noted that this function requires cooperation between the chip and the external PMIC. It is suggested that the user consult Freescale for such use cases.

55.3 USB PHY Memory Map/Register Definition

USBPHY Hardware Register Format Summary

USBPHY memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_9000	USB PHY Power-Down Register (USBPHY1_PWD)	32	R/W	001E_1C00h	55.3.1/3468
20C_9004	USB PHY Power-Down Register (USBPHY1_PWD_SET)	32	R/W	001E_1C00h	55.3.1/3468
20C_9008	USB PHY Power-Down Register (USBPHY1_PWD_CLR)	32	R/W	001E_1C00h	55.3.1/3468
20C_900C	USB PHY Power-Down Register (USBPHY1_PWD_TOG)	32	R/W	001E_1C00h	55.3.1/3468
20C_9010	USB PHY Transmitter Control Register (USBPHY1_TX)	32	R/W	1006_0607h	55.3.2/3470

Table continues on the next page...

USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_9014	USB PHY Transmitter Control Register (USBPHY1_TX_SET)	32	R/W	1006_0607h	55.3.2/3470
20C_9018	USB PHY Transmitter Control Register (USBPHY1_TX_CLR)	32	R/W	1006_0607h	55.3.2/3470
20C_901C	USB PHY Transmitter Control Register (USBPHY1_TX_TOG)	32	R/W	1006_0607h	55.3.2/3470
20C_9020	USB PHY Receiver Control Register (USBPHY1_RX)	32	R/W	0000_0000h	55.3.3/3471
20C_9024	USB PHY Receiver Control Register (USBPHY1_RX_SET)	32	R/W	0000_0000h	55.3.3/3471
20C_9028	USB PHY Receiver Control Register (USBPHY1_RX_CLR)	32	R/W	0000_0000h	55.3.3/3471
20C_902C	USB PHY Receiver Control Register (USBPHY1_RX_TOG)	32	R/W	0000_0000h	55.3.3/3471
20C_9030	USB PHY General Control Register (USBPHY1_CTRL)	32	R/W	C020_0000h	55.3.4/3473
20C_9034	USB PHY General Control Register (USBPHY1_CTRL_SET)	32	R/W	C020_0000h	55.3.4/3473
20C_9038	USB PHY General Control Register (USBPHY1_CTRL_CLR)	32	R/W	C020_0000h	55.3.4/3473
20C_903C	USB PHY General Control Register (USBPHY1_CTRL_TOG)	32	R/W	C020_0000h	55.3.4/3473
20C_9040	USB PHY Status Register (USBPHY1_STATUS)	32	R/W	0000_0000h	55.3.5/3476
20C_9050	USB PHY Debug Register (USBPHY1_DEBUG)	32	R/W	7F18_0000h	55.3.6/3478
20C_9054	USB PHY Debug Register (USBPHY1_DEBUG_SET)	32	R/W	7F18_0000h	55.3.6/3478
20C_9058	USB PHY Debug Register (USBPHY1_DEBUG_CLR)	32	R/W	7F18_0000h	55.3.6/3478
20C_905C	USB PHY Debug Register (USBPHY1_DEBUG_TOG)	32	R/W	7F18_0000h	55.3.6/3478
20C_9060	UTMI Debug Status Register 0 (USBPHY1_DEBUG0_STATUS)	32	R	0000_0000h	55.3.7/3480
20C_9070	UTMI Debug Status Register 1 (USBPHY1_DEBUG1)	32	R/W	0000_1000h	55.3.8/3481
20C_9074	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_SET)	32	R/W	0000_1000h	55.3.8/3481
20C_9078	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_CLR)	32	R/W	0000_1000h	55.3.8/3481
20C_907C	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_TOG)	32	R/W	0000_1000h	55.3.8/3481
20C_9080	UTMI RTL Version (USBPHY1_VERSION)	32	R	0402_0000h	55.3.9/3482
20C_A000	USB PHY Power-Down Register (USBPHY2_PWD)	32	R/W	001E_1C00h	55.3.1/3468
20C_A004	USB PHY Power-Down Register (USBPHY2_PWD_SET)	32	R/W	001E_1C00h	55.3.1/3468
20C_A008	USB PHY Power-Down Register (USBPHY2_PWD_CLR)	32	R/W	001E_1C00h	55.3.1/3468
20C_A00C	USB PHY Power-Down Register (USBPHY2_PWD_TOG)	32	R/W	001E_1C00h	55.3.1/3468
20C_A010	USB PHY Transmitter Control Register (USBPHY2_TX)	32	R/W	1006_0607h	55.3.2/3470
20C_A014	USB PHY Transmitter Control Register (USBPHY2_TX_SET)	32	R/W	1006_0607h	55.3.2/3470
20C_A018	USB PHY Transmitter Control Register (USBPHY2_TX_CLR)	32	R/W	1006_0607h	55.3.2/3470
20C_A01C	USB PHY Transmitter Control Register (USBPHY2_TX_TOG)	32	R/W	1006_0607h	55.3.2/3470
20C_A020	USB PHY Receiver Control Register (USBPHY2_RX)	32	R/W	0000_0000h	55.3.3/3471

Table continues on the next page...

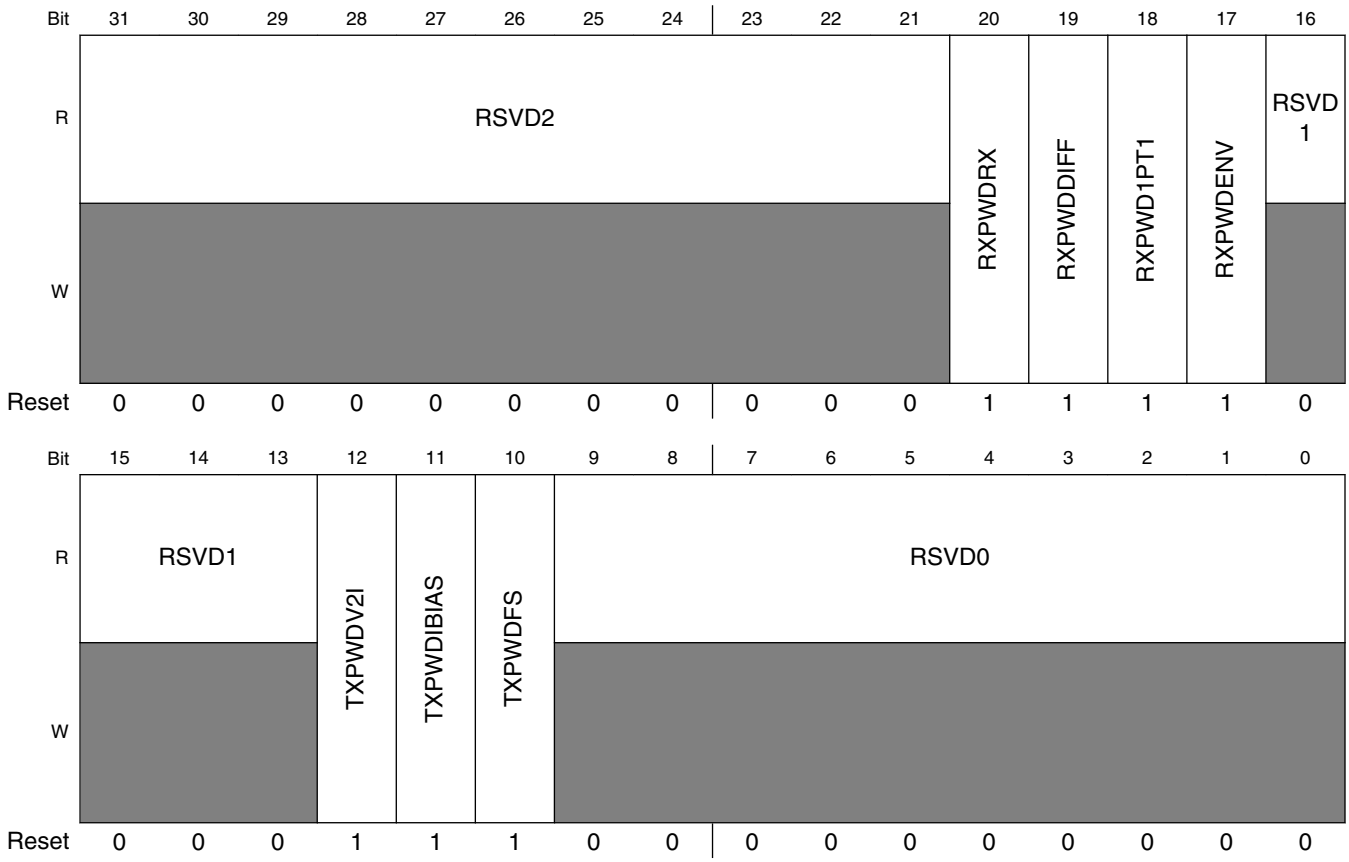
USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_A024	USB PHY Receiver Control Register (USBPHY2_RX_SET)	32	R/W	0000_0000h	55.3.3/3471
20C_A028	USB PHY Receiver Control Register (USBPHY2_RX_CLR)	32	R/W	0000_0000h	55.3.3/3471
20C_A02C	USB PHY Receiver Control Register (USBPHY2_RX_TOG)	32	R/W	0000_0000h	55.3.3/3471
20C_A030	USB PHY General Control Register (USBPHY2_CTRL)	32	R/W	C020_0000h	55.3.4/3473
20C_A034	USB PHY General Control Register (USBPHY2_CTRL_SET)	32	R/W	C020_0000h	55.3.4/3473
20C_A038	USB PHY General Control Register (USBPHY2_CTRL_CLR)	32	R/W	C020_0000h	55.3.4/3473
20C_A03C	USB PHY General Control Register (USBPHY2_CTRL_TOG)	32	R/W	C020_0000h	55.3.4/3473
20C_A040	USB PHY Status Register (USBPHY2_STATUS)	32	R/W	0000_0000h	55.3.5/3476
20C_A050	USB PHY Debug Register (USBPHY2_DEBUG)	32	R/W	7F18_0000h	55.3.6/3478
20C_A054	USB PHY Debug Register (USBPHY2_DEBUG_SET)	32	R/W	7F18_0000h	55.3.6/3478
20C_A058	USB PHY Debug Register (USBPHY2_DEBUG_CLR)	32	R/W	7F18_0000h	55.3.6/3478
20C_A05C	USB PHY Debug Register (USBPHY2_DEBUG_TOG)	32	R/W	7F18_0000h	55.3.6/3478
20C_A060	UTMI Debug Status Register 0 (USBPHY2_DEBUG0_STATUS)	32	R	0000_0000h	55.3.7/3480
20C_A070	UTMI Debug Status Register 1 (USBPHY2_DEBUG1)	32	R/W	0000_1000h	55.3.8/3481
20C_A074	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_SET)	32	R/W	0000_1000h	55.3.8/3481
20C_A078	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_CLR)	32	R/W	0000_1000h	55.3.8/3481
20C_A07C	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_TOG)	32	R/W	0000_1000h	55.3.8/3481
20C_A080	UTMI RTL Version (USBPHY2_VERSION)	32	R	0402_0000h	55.3.9/3482

55.3.1 USB PHY Power-Down Register (USBPHYx_PWD*n*)

The USB PHY Power-Down Register provides overall control of the PHY power state. Before programming this register, the PHY clocks must be enabled in registers USBPHYx_CTRLn and CCM_ANALOG_USBPHYx_PLL_480_CTRLn.

Address: Base address + 0h offset + (4d × i), where i=0d to 3d



USBPHYx_PWD*n* field descriptions

Field	Description
31–21 RSVD2	Reserved.
20 RXPWDRX	0 = Normal operation. 1 = Power-down the entire USB PHY receiver block except for the full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
19 RXPWDDIFF	0 = Normal operation. 1 = Power-down the USB high-speed differential receiver.

Table continues on the next page...

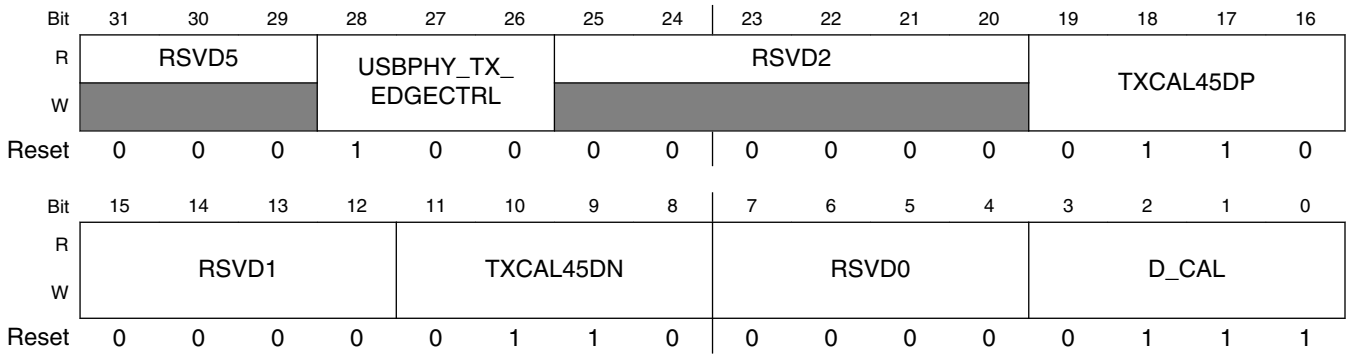
USBPHYx_PWD*n* field descriptions (continued)

Field	Description
	Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
18 RXPWD1PT1	0 = Normal operation. 1 = Power-down the USB full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
17 RXPWDENV	0 = Normal operation. 1 = Power-down the USB high-speed receiver envelope detector (squelch signal). Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
16–13 RSVD1	Reserved.
12 TXPWDV2I	0 = Normal operation. 1 = Power-down the USB PHY transmit V-to-I converter and the current mirror. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
11 TXPWDIBIAS	0 = Normal operation. 1 = Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
10 TXPWDFS	0 = Normal operation. 1 = Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
RSVD0	Reserved.

55.3.2 USB PHY Transmitter Control Register (USBPHY*x*_TX*n*)

The USB PHY Transmitter Control Register handles the transmit controls.

Address: Base address + 10h offset + (4d × i), where i=0d to 3d



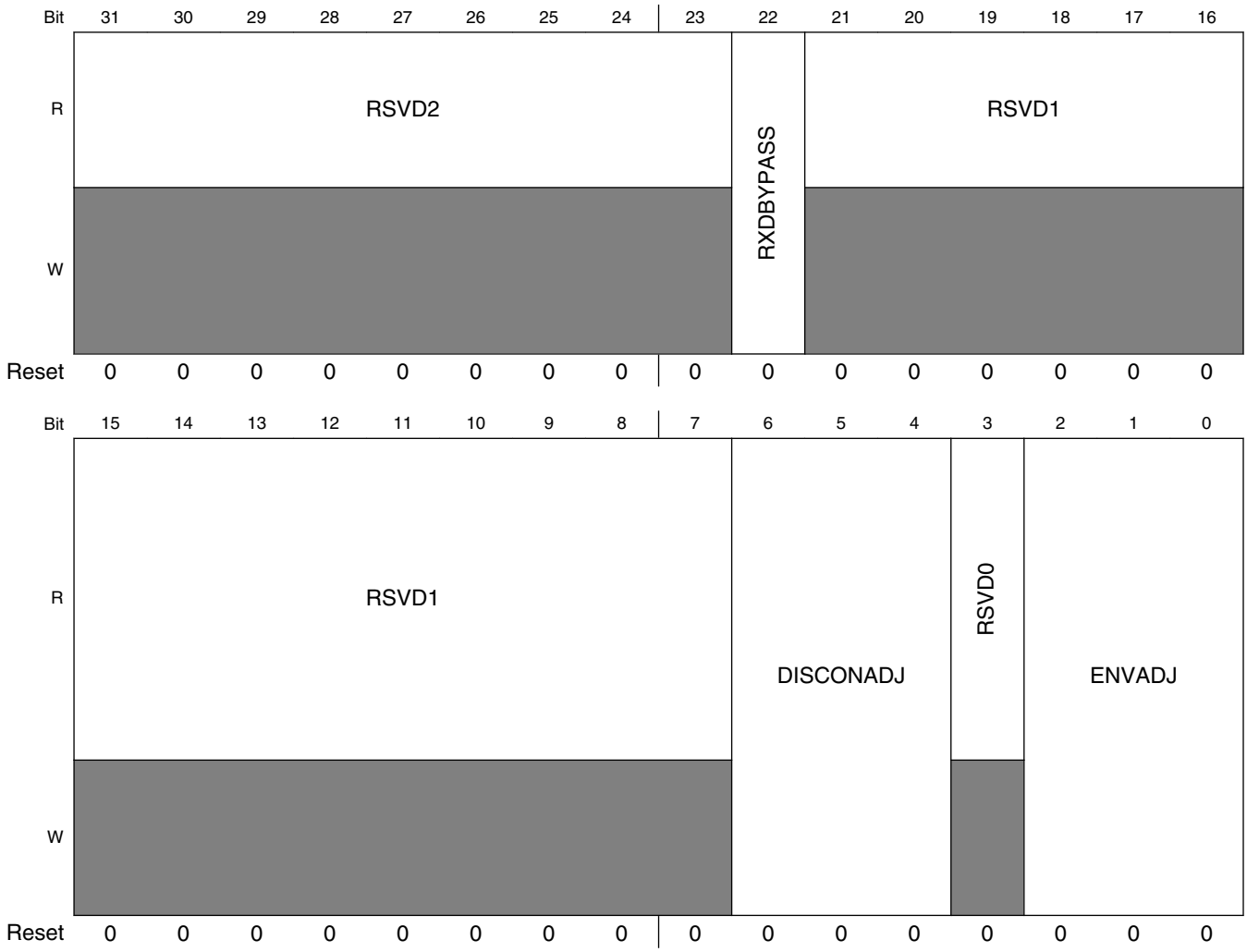
USBPHY*x*_TX*n* field descriptions

Field	Description
31–29 RSVD5	Reserved.
28–26 USBPHY_TX_ EDGECTRL	Controls the edge-rate of the current sensing transistors used in HS transmit. NOT FOR CUSTOMER USE.
25–20 RSVD2	Reserved.
19–16 TXCAL45DP	Decode to select a 45-Ohm resistance to the USB_DP output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
15–12 RSVD1	Reserved. **Note:** This bit should remain clear.
11–8 TXCAL45DN	Decode to select a 45-Ohm resistance to the USB_DN output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
7–4 RSVD0	Reserved. **Note:** This bit should remain clear.
D_CAL	Resistor Trimming Code: 0000 = 0.16% 0111 = Nominal 1111 = +25%

55.3.3 USB PHY Receiver Control Register (USBPHY*x*_RX*n*)

The USB PHY Receiver Control Register handles receive path controls.

Address: Base address + 20h offset + (4d × i), where i=0d to 3d



USBPHY*x*_RX*n* field descriptions

Field	Description
31–23 RSVD2	Reserved.
22 RXDBYPASS	0 = Normal operation. 1 = Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver. This test mode is intended for lab use only.
21–7 RSVD1	Reserved.

Table continues on the next page...

USBPHYx_RXn field descriptions (continued)

Field	Description
6–4 DISCONADJ	The DISCONADJ field adjusts the trip point for the disconnect detector: 000 = Trip-Level Voltage is 0.57500 V 001 = Trip-Level Voltage is 0.56875 V 010 = Trip-Level Voltage is 0.58125 V 011 = Trip-Level Voltage is 0.58750 V 1XX = Reserved
3 RSVD0	Reserved.
ENVADJ	The ENVADJ field adjusts the trip point for the envelope detector. 000 = Trip-Level Voltage is 0.12500 V 001 = Trip-Level Voltage is 0.10000 V 010 = Trip-Level Voltage is 0.13750 V 011 = Trip-Level Voltage is 0.15000 V 1XX = Reserved

55.3.4 USB PHY General Control Register (USBPHYx_CTRLn)

The USB PHY General Control Register handles OTG and Host controls. This register also includes interrupt enables and connectivity detect enables and results.

Address: Base address + 30h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	UTMI_SUSPENDM	HOST_FORCE_LS_SE0	OTG_ID_VALUE	RSVD1		FSDLL_RST_EN	ENVBUSCHG_WKUP	ENIDCHG_WKUP	ENDPDMCHG_WKUP	ENAUTOCLR_PHY_PWD	ENAUTOCLR_CLKGATE	RSVD0	WAKEUP_IRQ	ENIRQWAKEUP
W																
Reset	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUTMILEVEL3	ENUTMILEVEL2	DATA_ON_LRADC	DEVPLUGIN_IRQ	ENIRQDEVPLUGIN	RESUME_IRQ	ENIRQRESUMEDETECT	RESUMEIRQSTICKY	ENOTGIDDETECT	OTG_ID_CHG_IRQ	DEVPLUGIN_POLARITY	ENDEVPLUGINDETECT	HOSTDISCONDETECT_IRQ	ENIRQHOSTDISCON	ENHOSTDISCONDETECT	ENOTG_ID_CHG_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBPHYx_CTRL*n* field descriptions

Field	Description
31 SFTRST	Writing a 1 to this bit will soft-reset the USBPHYx_PWD, USBPHYx_TX, USBPHYx_RX, and USBPHYx_CTRL registers. Set to 0 to release the PHY from reset.
30 CLKGATE	Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wakeup event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHYx_CTRL is enabled.
29 UTMI_SUSPENDM	Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHYx_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.
28 HOST_FORCE_LS_SE0	Forces the next FS packet that is transmitted to have a EOP with LS timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHYx_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHYx_DEBUG_HOST_RESUME_DEBUG.
27 OTG_ID_VALUE	Almost same as OTGID_STATUS in USBPHYx_STATUS Register. The only difference is that OTG_ID_VALUE has debounce logic to filter the glitches on ID Pad.
26–25 RSVD1	Reserved.
24 FSDLL_RST_EN	Enables the feature to reset the FSDLL lock detection logic at the end of each TX packet.
23 ENVBUSCHG_WKUP	Enables the feature to wakeup USB if VBUS is toggled when USB is suspended.
22 ENIDCHG_WKUP	Enables the feature to wakeup USB if ID is toggled when USB is suspended.
21 ENDPDMCHG_WKUP	Enables the feature to wakeup USB if DP/DM is toggled when USB is suspended. This bit is enabled by default.
20 ENAUTOCLR_PHY_PWD	Enables the feature to auto-clear the PWD register bits in USBPHYx_PWD if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
19 ENAUTOCLR_CLKGATE	Enables the feature to auto-clear the CLKGATE bit if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
18 RSVD0	Reserved.
17 WAKEUP_IRQ	Indicates that there is a wakeup event. Reset this bit by writing a 1 to the clear address space and not by a general write.
16 ENIRQWAKEUP	Enables interrupt for the wakeup events.
15 ENUTMILEVEL3	Enables UTMI+ Level3. This should be enabled if needs to support external FS Hub with LS device connected
14 ENUTMILEVEL2	Enables UTMI+ Level2. This should be enabled if needs to support LS device
13 DATA_ON_LRADC	Enables the LRADC to monitor USB_DP and USB_DM. This is for use in non-USB modes only.
12 DEVPLUGIN_IRQ	Indicates that the device is connected. Reset this bit by writing a 1 to the clear address space and not by a general write.
11 ENIRQDEVPLUGIN	Enables interrupt for the detection of connectivity to the USB line.

Table continues on the next page...

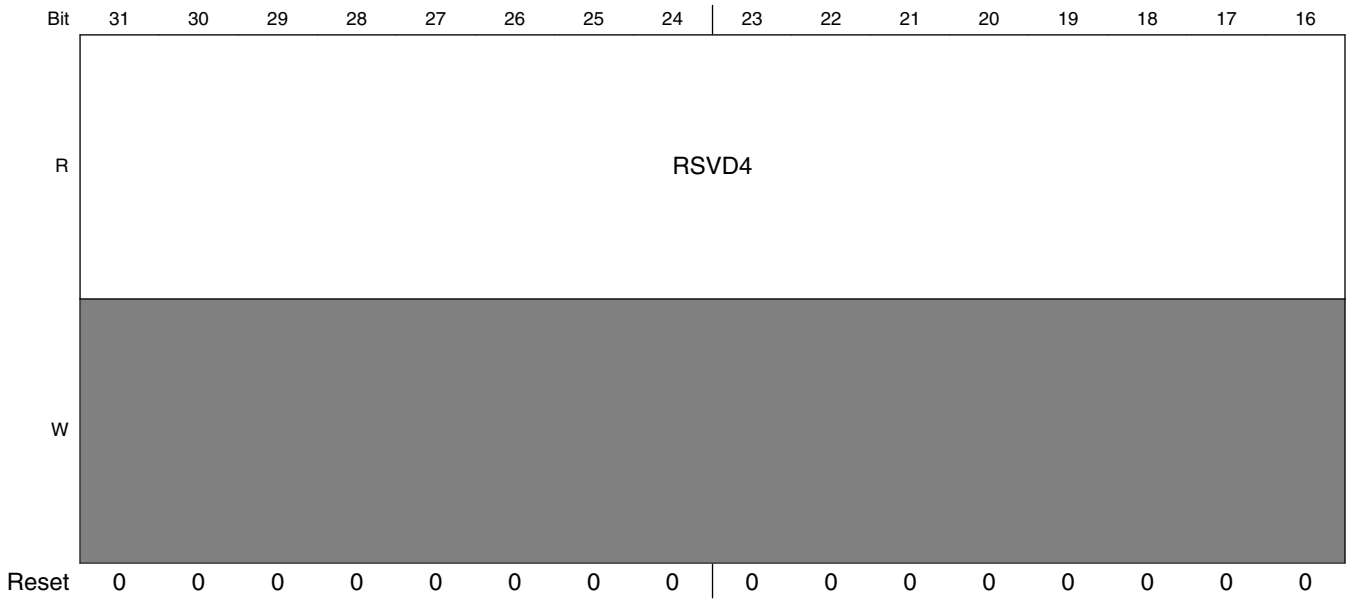
USBPHY*x*_CTRL*n* field descriptions (continued)

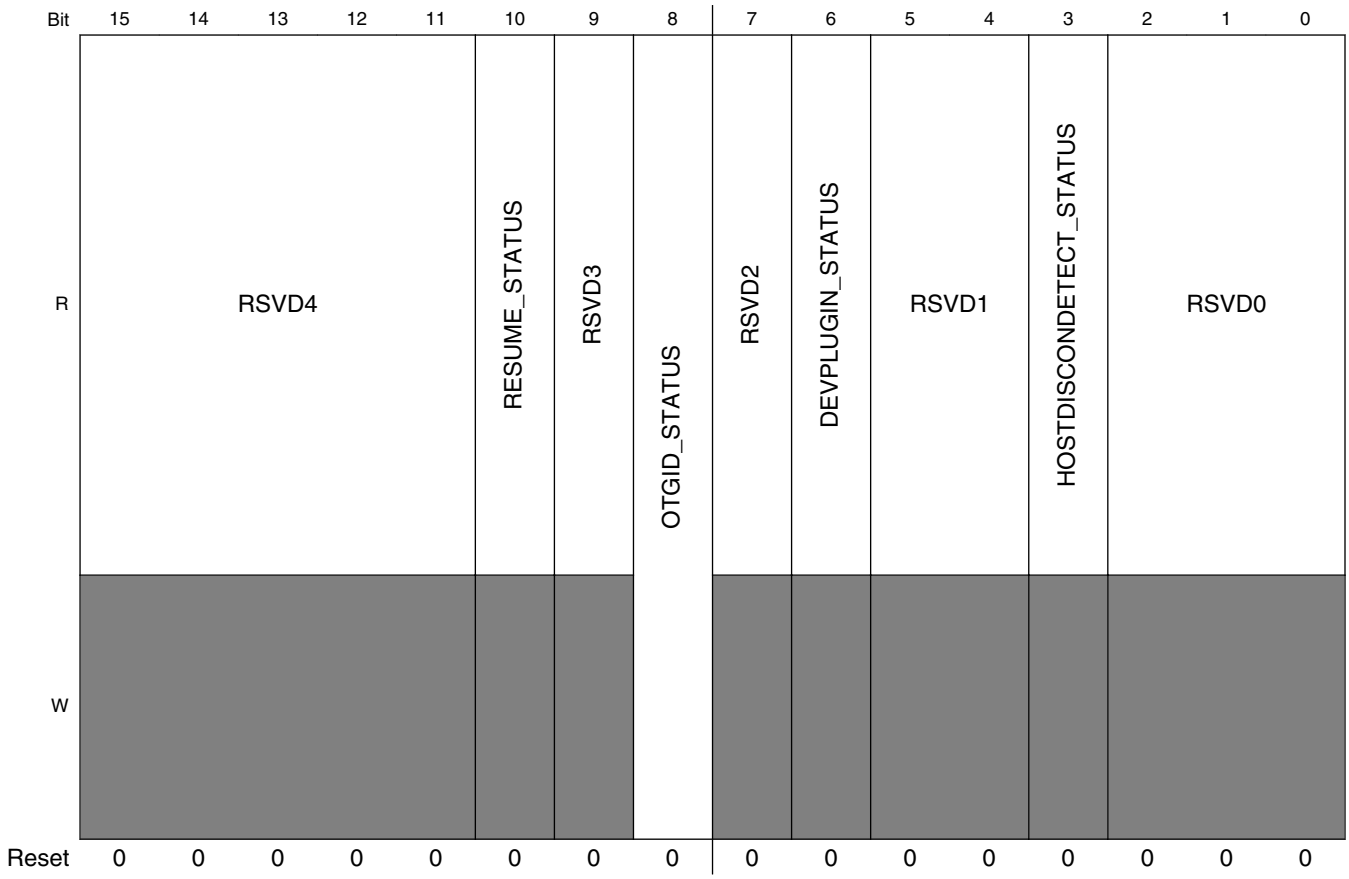
Field	Description
10 RESUME_IRQ	Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the clear address space and not by a general write.
9 ENIRQRESUMEDETECT	Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.
8 RESUMEIRQSTICKY	Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.
7 ENOTGIDDETECT	Enables circuit to detect resistance of MiniAB ID pin.
6 OTG_ID_CHG_IRQ	OTG ID change interrupt. Indicates the value of ID pin changed.
5 DEVPLUGIN_POLARITY	For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.
4 ENDEVPLUGINDETECT	For device mode, enables 200-KOhm pullups for detecting connectivity to the host.
3 HOSTDISCONDETECT_ IRQ	Indicates that the device has disconnected in high-speed mode. Reset this bit by writing a 1 to the clear address space and not by a general write.
2 ENIRQHOSTDISCON	Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.
1 ENHOSTDISCONDETECT	For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. SW shall set this bit when it found the high-speed device is connected, suggested during bus reset, after found high-speed device in USB_PORTSC1.PSPD). SW shall make sure this bit is not set at the end of resume, otherwise a wrong disconnect status may be detected. Suggest clear it after set USB_PORTSC1.SUSP, set it again after resume is ended(USB_PORTSC1.FPR==0).
0 ENOTG_ID_CHG_IRQ	Enable OTG_ID_CHG_IRQ.

55.3.5 USB PHY Status Register (USBPHYx_STATUS)

The USB PHY Status Register holds results of IRQ and other detects.

Address: Base address + 40h offset





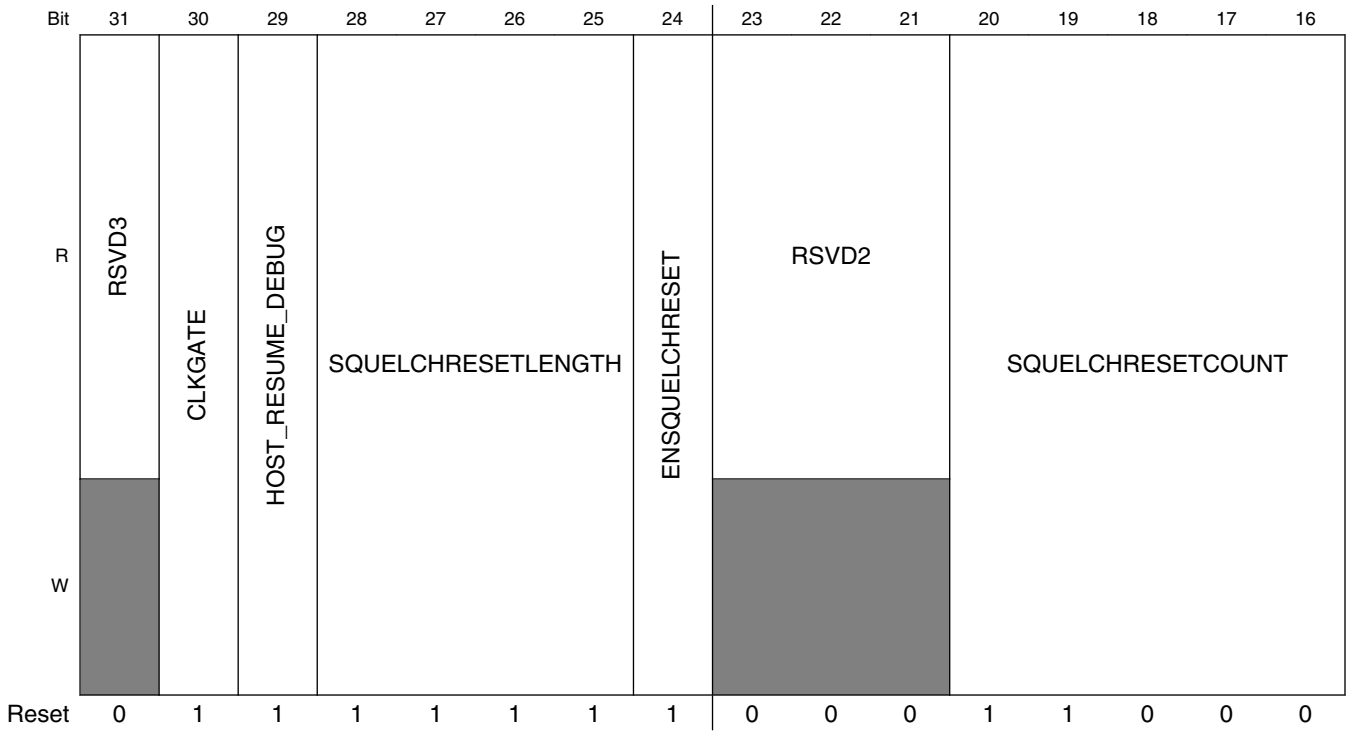
USBPHYx_STATUS field descriptions

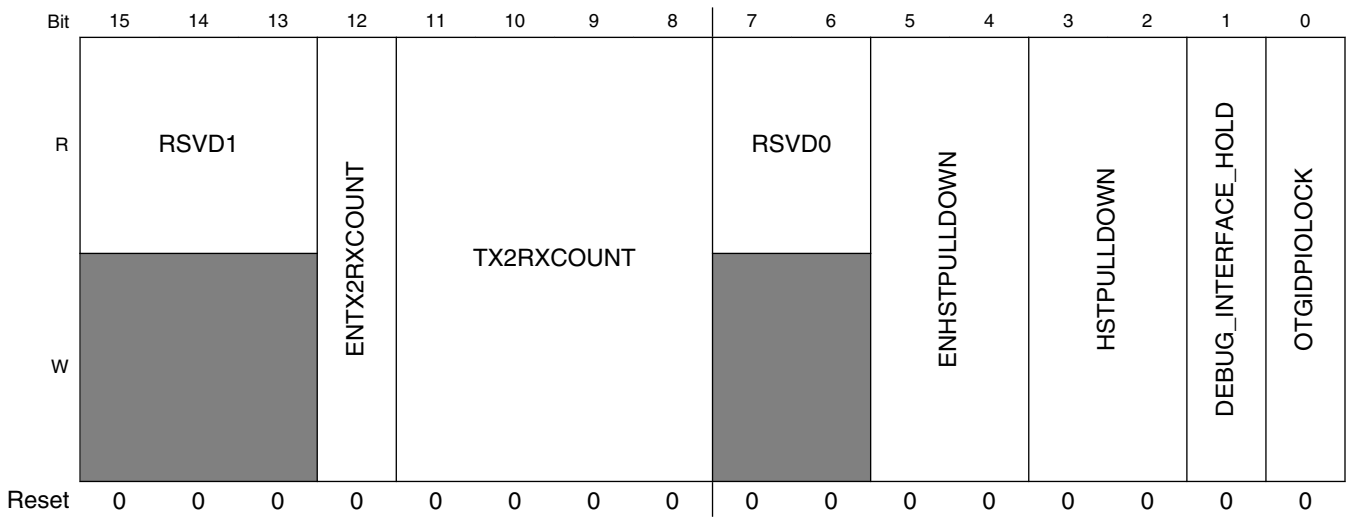
Field	Description
31–11 RSVD4	Reserved.
10 RESUME_STATUS	Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.
9 RSVD3	Reserved.
8 OTGID_STATUS	Indicates the results of ID pin on MiniAB plug. False (0) is when ID resistance is less than Ra_Plug_ID, indicating host (A) side. True (1) is when ID resistance is greater than Rb_Plug_ID, indicating device (B) side.
7 RSVD2	Reserved.
6 DEVPLUGIN_STATUS	Indicates that the device has been connected on the USB_DP and USB_DM lines.
5–4 RSVD1	Reserved.
3 HOSTDISCONDETECT_ STATUS	Indicates that the device has disconnected while in high-speed host mode.
RSVD0	Reserved.

55.3.6 USB PHY Debug Register (USBPHYx_DEBUGn)

This register is used to debug the USB PHY.

Address: Base address + 50h offset + (4d × i), where i=0d to 3d





USBPHYx_DEBUG*n* field descriptions

Field	Description
31 RSVD3	Reserved.
30 CLKGATE	Gate Test Clocks. Clear to 0 for running clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29 HOST_RESUME_DEBUG	Choose to trigger the host resume SE0 with HOST_FORCE_LS_SE0 = 0 or UTMI_SUSPEND = 1.
28–25 SQUELCHRESETLENGTH	Duration of RESET in terms of the number of 480-MHz cycles.
24 ENSQUELCHRESET	Set bit to allow squelch to reset high-speed receive.
23–21 RSVD2	Reserved.
20–16 SQUELCHRESETCOUNT	Delay in between the detection of squelch to the reset of high-speed RX.
15–13 RSVD1	Reserved.
12 ENTX2RXCOUNT	Set this bit to allow a countdown to transition in between TX and RX.
11–8 TX2RXCOUNT	Delay in between the end of transmit to the beginning of receive. This is a Johnson count value and thus will count to 8.
7–6 RSVD0	Reserved.
5–4 ENHSTPULLDOWN	Set bit 5 to 1 to override the control of the USB_DP 15-KOhm pulldown. Set bit 4 to 1 to override the control of the USB_DM 15-KOhm pulldown. Clear to 0 to disable.
3–2 HSTPULLDOWN	Set bit 3 to 1 to pull down 15-KOhm on USB_DP line. Set bit 2 to 1 to pull down 15-KOhm on USB_DM line. Clear to 0 to disable.
1 DEBUG_INTERFACE_HOLD	Use holding registers to assist in timing for external UTMI interface.
0 OTGIDPIOLOCK	Once OTG ID from USBPHYx_STATUS_OTGID_STATUS, use this to hold the value. This is to save power for the comparators that are used to determine the ID status.

55.3.7 UTMI Debug Status Register 0 (USBPHYx_DEBUG0_STATUS)

The UTMI Debug Status Register 0 holds multiple views for counters and status of state machines. This is used in conjunction with the USBPHYx_DEBUG1_DBG_ADDRESS field to choose which function to view. The default is described in the bit fields below and is used to count errors.

Address: Base address + 60h offset

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

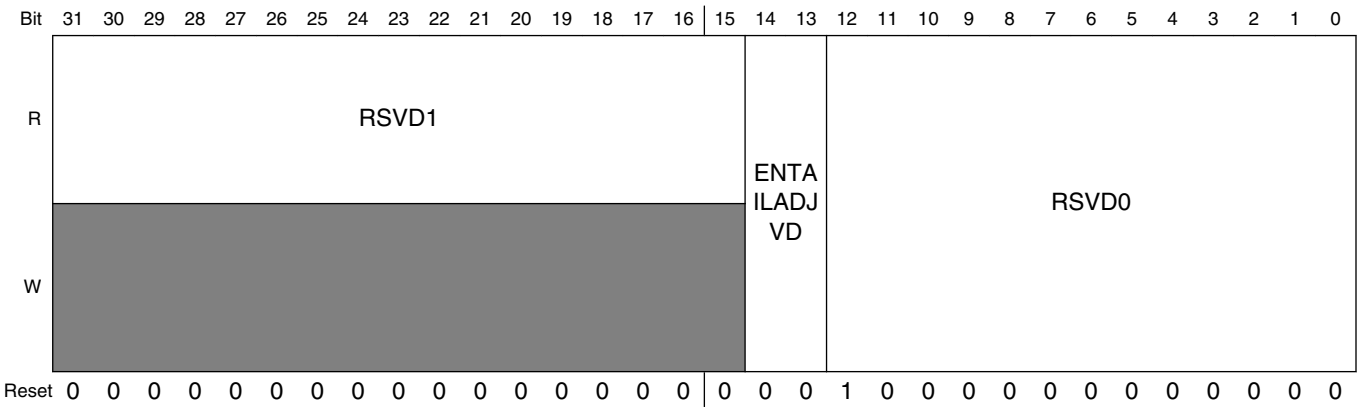
USBPHYx_DEBUG0_STATUS field descriptions

Field	Description
31–26 SQUELCH_ COUNT	Running count of the squelch reset instead of normal end for HS RX.
25–16 UTMI_ RXERROR_ FAIL_COUNT	Running count of the UTMI_RXERROR.
LOOP_BACK_ FAIL_COUNT	Running count of the failed pseudo-random generator loopback. Each time entering testmode, counter goes to 900D and will count up for every detected packet failure in digital/analog loopback tests.

55.3.8 UTMI Debug Status Register 1 (USBPHYx_DEBUG1*n*)

Chooses the muxing of the debug register to be shown in USBPHYx_DEBUG0_STATUS.

Address: Base address + 70h offset + (4d × i), where i=0d to 3d



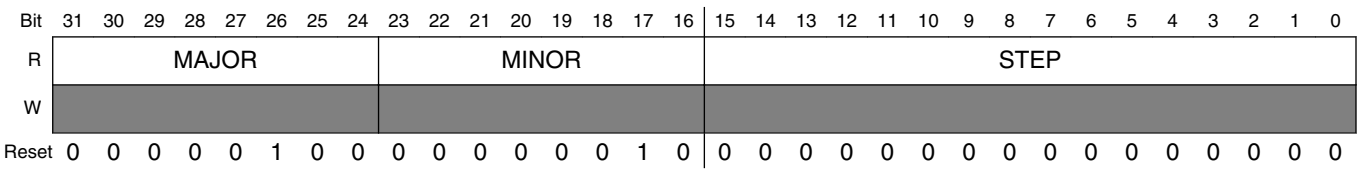
USBPHYx_DEBUG1*n* field descriptions

Field	Description
31–15 RSVD1	Reserved.
14–13 ENTAILADJVD	Delay increment of the rise of squelch: 00 = Delay is nominal 01 = Delay is +20% 10 = Delay is -20% 11 = Delay is -40%
RSVD0	Reserved. **Note:** This bit should remain clear.

55.3.9 UTMI RTL Version (USBPHYx_VERSION)

Fields for RTL Version.

Address: Base address + 80h offset



USBPHYx_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

55.4 USB Analog Memory Map/Register Definition

USB_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A0	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT)	32	R/W	0010_0004h	55.4.1/3484
1A4	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_SET)	32	R/W	0010_0004h	55.4.1/3484
1A8	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_CLR)	32	R/W	0010_0004h	55.4.1/3484
1AC	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_TOG)	32	R/W	0010_0004h	55.4.1/3484
1B0	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT)	32	R/W	0000_0000h	55.4.2/3485
1B4	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_SET)	32	R/W	0000_0000h	55.4.2/3485
1B8	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_CLR)	32	R/W	0000_0000h	55.4.2/3485
1BC	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_TOG)	32	R/W	0000_0000h	55.4.2/3485

Table continues on the next page...

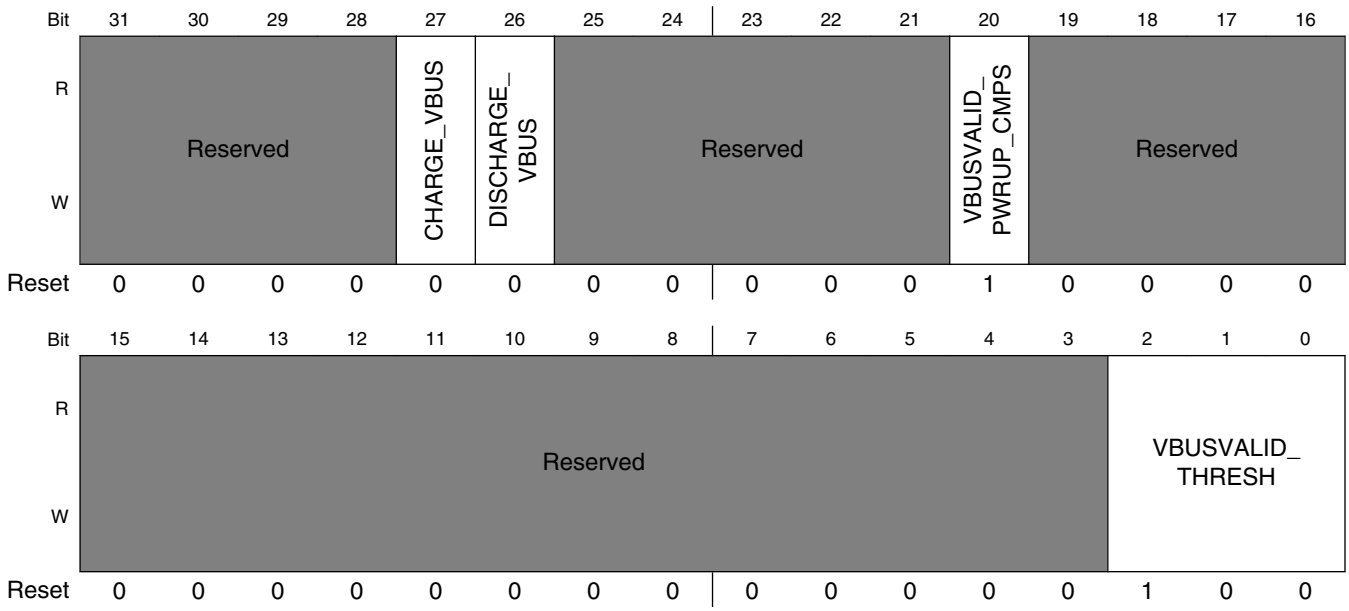
USB_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C0	USB VBUS Detect Status Register (USB_ANALOG_USB1_VBUS_DETECT_STAT)	32	R	0000_0000h	55.4.3/3487
1D0	USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)	32	R	0000_0000h	55.4.4/3489
1F0	USB Misc Register (USB_ANALOG_USB1_MISC)	32	R/W	0000_0002h	55.4.5/3490
1F4	USB Misc Register (USB_ANALOG_USB1_MISC_SET)	32	R/W	0000_0002h	55.4.5/3490
1F8	USB Misc Register (USB_ANALOG_USB1_MISC_CLR)	32	R/W	0000_0002h	55.4.5/3490
1FC	USB Misc Register (USB_ANALOG_USB1_MISC_TOG)	32	R/W	0000_0002h	55.4.5/3490
200	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT)	32	R/W	0010_0004h	55.4.6/3491
204	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_SET)	32	R/W	0010_0004h	55.4.6/3491
208	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_CLR)	32	R/W	0010_0004h	55.4.6/3491
20C	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_TOG)	32	R/W	0010_0004h	55.4.6/3491
210	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT)	32	R/W	0000_0000h	55.4.7/3493
214	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_SET)	32	R/W	0000_0000h	55.4.7/3493
218	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_CLR)	32	R/W	0000_0000h	55.4.7/3493
21C	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_TOG)	32	R/W	0000_0000h	55.4.7/3493
220	USB VBUS Detect Status Register (USB_ANALOG_USB2_VBUS_DETECT_STAT)	32	R	0000_0000h	55.4.8/3495
230	USB Charger Detect Status Register (USB_ANALOG_USB2_CHRG_DETECT_STAT)	32	R	0000_0000h	55.4.9/3497
250	USB Misc Register (USB_ANALOG_USB2_MISC)	32	R/W	0000_0002h	55.4.10/ 3498
254	USB Misc Register (USB_ANALOG_USB2_MISC_SET)	32	R/W	0000_0002h	55.4.10/ 3498
258	USB Misc Register (USB_ANALOG_USB2_MISC_CLR)	32	R/W	0000_0002h	55.4.10/ 3498
25C	USB Misc Register (USB_ANALOG_USB2_MISC_TOG)	32	R/W	0000_0002h	55.4.10/ 3498
260	Chip Silicon Version (USB_ANALOG_DIGPROG)	32	R	0064_0000h	55.4.11/ 3499

55.4.1 USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT*n*)

This register defines controls for USB VBUS detect.

Address: 0h base + 1A0h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB1_VBUS_DETECT*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved.
27 CHARGE_VBUS	USB OTG charge VBUS.
26 DISCHARGE_ VBUS	USB OTG discharge VBUS.
25–21 -	This field is reserved. Reserved.
20 VBUSVALID_ PWRUP_CMPS	Powers up comparators for vbus_valid detector.
19–3 -	This field is reserved. Reserved.
VBUSVALID_ THRESH	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 000 **4V0** — 4.0V

Table continues on the next page...

USB_ANALOG_USB1_VBUS_DETECT*n* field descriptions (continued)

Field	Description
	001 **4V1** — 4.1V 010 **4V2** — 4.2V 011 **4V3** — 4.3V 100 **4V4** — 4.4V (default) 101 **4V5** — 4.5V 110 **4V6** — 4.6V 111 **4V7** — 4.7V

55.4.2 USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT*n*)

This register defines controls for USB charger detect.

Address: 0h base + 1B0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								Reserved	Reserved		EN_B	CHK_CHRG_B	CHK_CONTACT	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

USB_ANALOG_USB1_CHRG_DETECT*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.

Table continues on the next page...

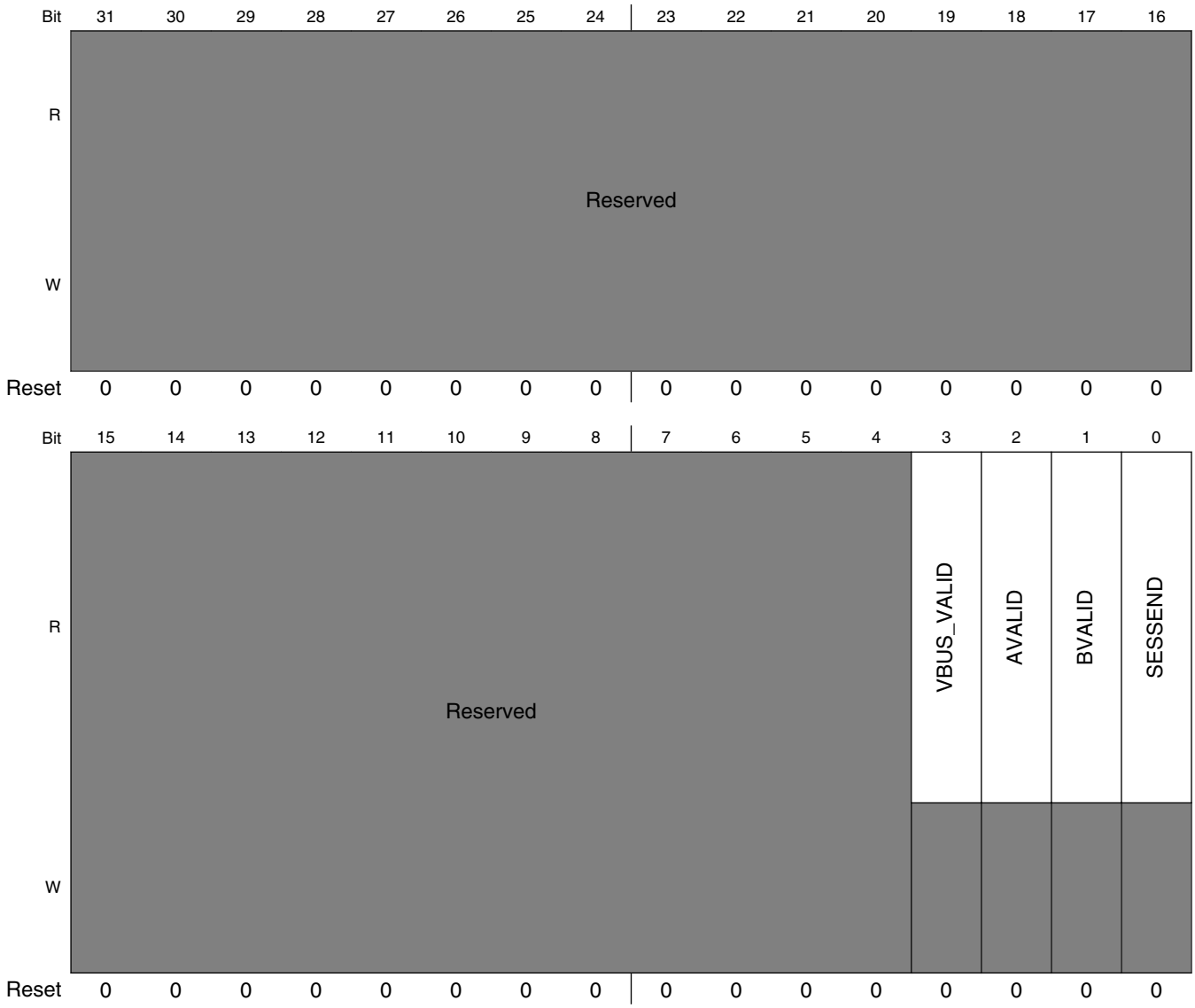
USB_ANALOG_USB1_CHRG_DETECT*n* field descriptions (continued)

Field	Description
23 -	This field is reserved. Reserved.
22–21 -	This field is reserved. Reserved.
20 EN_B	Control the charger detector. 0 **ENABLE** — Enable the charger detector. 1 **DISABLE** — Disable the charger detector.
19 CHK_CHRG_B	0 **CHECK** — Check whether a charger (either a dedicated charger or a host charger) is connected to USB port. 1 **NO_CHECK** — Do not check whether a charger is connected to the USB port.
18 CHK_CONTACT	0 **NO_CHECK** — Do not check the contact of USB plug. 1 **CHECK** — Check whether the USB plug has been in contact with each other
-	This field is reserved. Reserved.

55.4.3 USB VBUS Detect Status Register (USB_ANALOG_USB1_VBUS_DETECT_STAT)

This register defines fields for USB VBUS Detect status.

Address: 0h base + 1C0h offset = 1C0h



USB_ANALOG_USB1_VBUS_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.

Table continues on the next page...

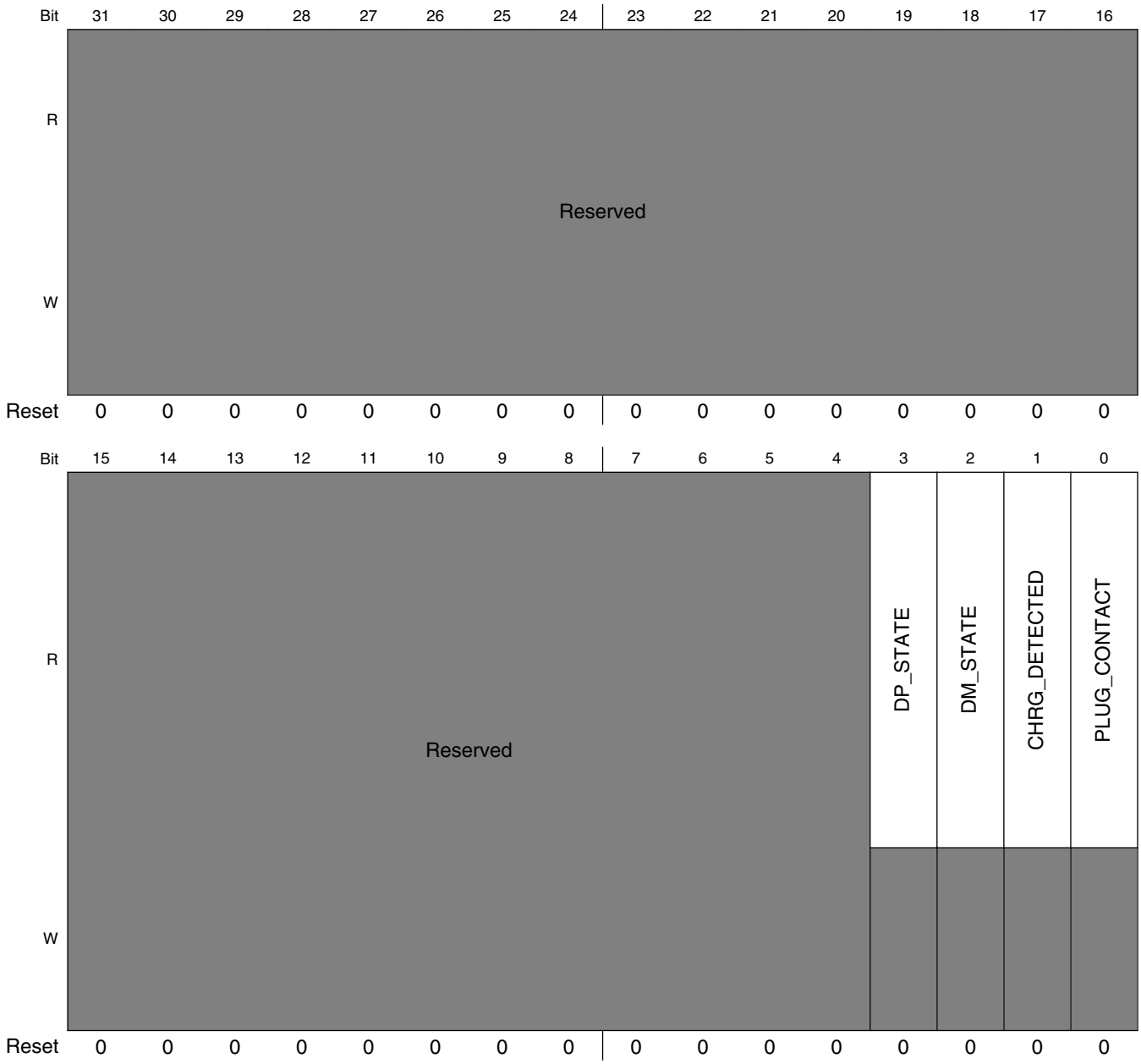
USB_ANALOG_USB1_VBUS_DETECT_STAT field descriptions (continued)

Field	Description
3 VBUS_VALID	VBus valid for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software.
2 AVALID	Indicates VBus is valid for a A-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
1 BVALID	Indicates VBus is valid for a B-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
0 SESSEND	Session End for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the SESSEND bit below. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V is present, 1 if VDD5V is not present.

55.4.4 USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)

This register defines fields for USB charger detect status.

Address: 0h base + 1D0h offset = 1D0h



USB_ANALOG_USB1_CHRG_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.
3 DP_STATE	DP line state output of the charger detector.
2 DM_STATE	DM line state output of the charger detector.
1 CHRG_ DETECTED	State of charger detection. This bit is a read only version of the state of the analog signal. 0 **CHARGER_NOT_PRESENT** — The USB port is not connected to a charger. 1 **CHARGER_PRESENT** — A charger (either a dedicated charger or a host charger) is connected to the USB port.
0 PLUG_ CONTACT	State of the USB plug contact detector. 0 **NO_CONTACT** — The USB plug has not made contact. 1 **GOOD_CONTACT** — The USB plug has made good contact.

55.4.5 USB Misc Register (USB_ANALOG_USB1_MISC*n*)

This register defines controls for USB.

Address: 0h base + 1F0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	EN_CLK_UTMI	Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

USB_ANALOG_USB1_MISC*n* field descriptions

Field	Description
31 -	This field is reserved. Reserved.
30 EN_CLK_UTMI	Enables the clk to the UTMI block.
29–2 -	This field is reserved. Reserved.
1 EN_DEGLITCH	Enable the deglitching circuit of the USB PLL output.
0 HS_USE_ EXTERNAL_R	Use external resistor to generate the current bias for the high speed transmitter. This bit should not be changed unless recommended by Freescale.

55.4.6 USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT*n*)

This register defines controls for USB VBUS detect.

Address: 0h base + 200h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				CHARGE_VBUS	DISCHARGE_VBUS	Reserved					VBUSVALID_PWRUP_CMPS	Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

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

USB_ANALOG_USB2_VBUS_DETECT*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved.

Table continues on the next page...

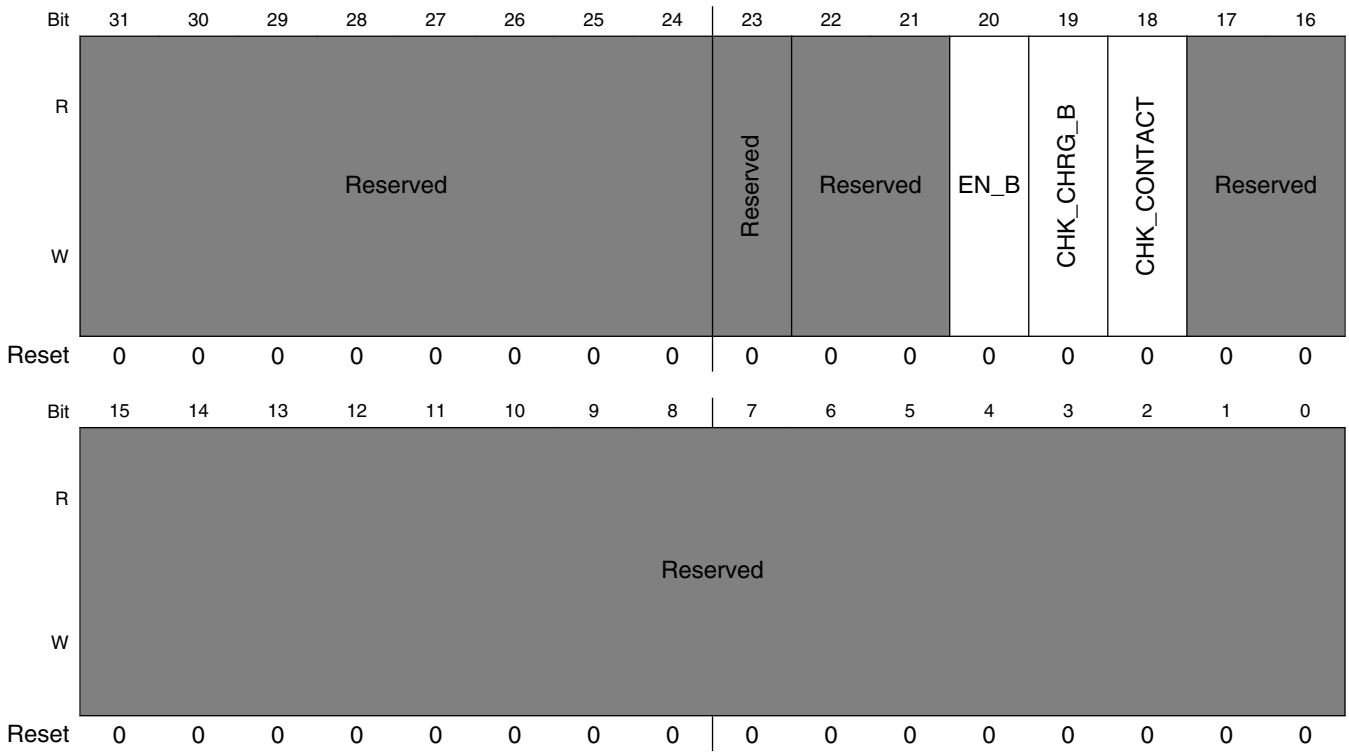
USB_ANALOG_USB2_VBUS_DETECT*n* field descriptions (continued)

Field	Description
27 CHARGE_VBUS	USB OTG charge VBUS.
26 DISCHARGE_ VBUS	USB OTG discharge VBUS.
25–21 -	This field is reserved. Reserved.
20 VBUSVALID_ PWRUP_CMPS	Powers up comparators for vbus_valid detector.
19–3 -	This field is reserved. Reserved.
VBUSVALID_ THRESH	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 000 **4V0** — 4.0V 001 **4V1** — 4.1V 010 **4V2** — 4.2V 011 **4V3** — 4.3V 100 **4V4** — 4.4V (default) 101 **4V5** — 4.5V 110 **4V6** — 4.6V 111 **4V7** — 4.7V

55.4.7 USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT*n*)

This register defines controls for USB charger detect.

Address: 0h base + 210h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB2_CHRG_DETECT*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23 -	This field is reserved. Reserved.
22–21 -	This field is reserved. Reserved.
20 EN_B	Control the charger detector. 0 **ENABLE** — Enable the charger detector. 1 **DISABLE** — Disable the charger detector.
19 CHK_CHRG_B	0 **CHECK** — Check whether a charger (either a dedicated charger or a host charger) is connected to USB port. 1 **NO_CHECK** — Do not check whether a charger is connected to the USB port.

Table continues on the next page...

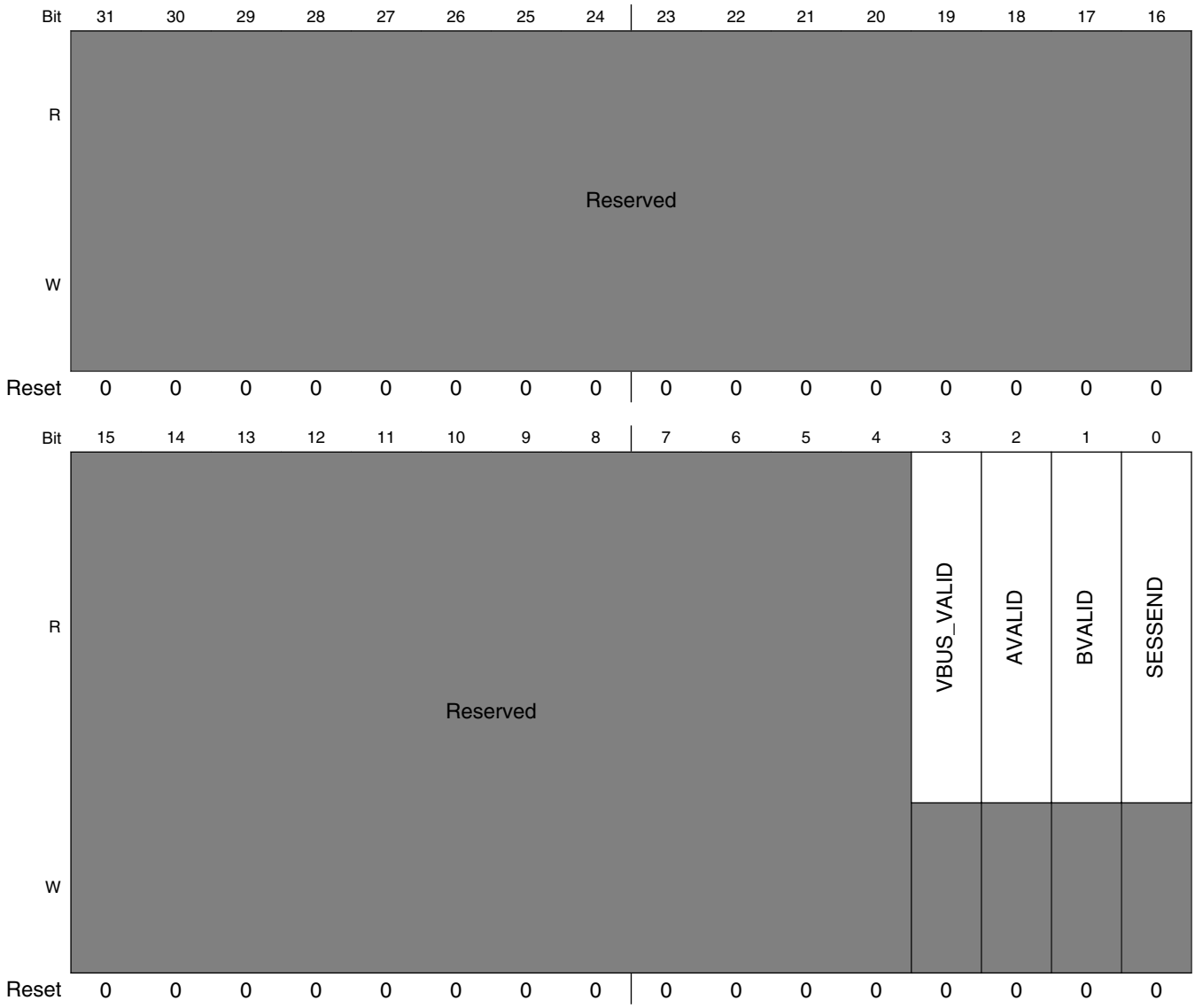
USB_ANALOG_USB2_CHRG_DETECT*n* field descriptions (continued)

Field	Description
18 CHK_CONTACT	0 **NO_CHECK** — Do not check the contact of USB plug. 1 **CHECK** — Check whether the USB plug has been in contact with each other
-	This field is reserved. Reserved.

55.4.8 USB VBUS Detect Status Register (USB_ANALOG_USB2_VBUS_DETECT_STAT)

This register defines fields for USB VBUS Detect status.

Address: 0h base + 220h offset = 220h



USB_ANALOG_USB2_VBUS_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.

Table continues on the next page...

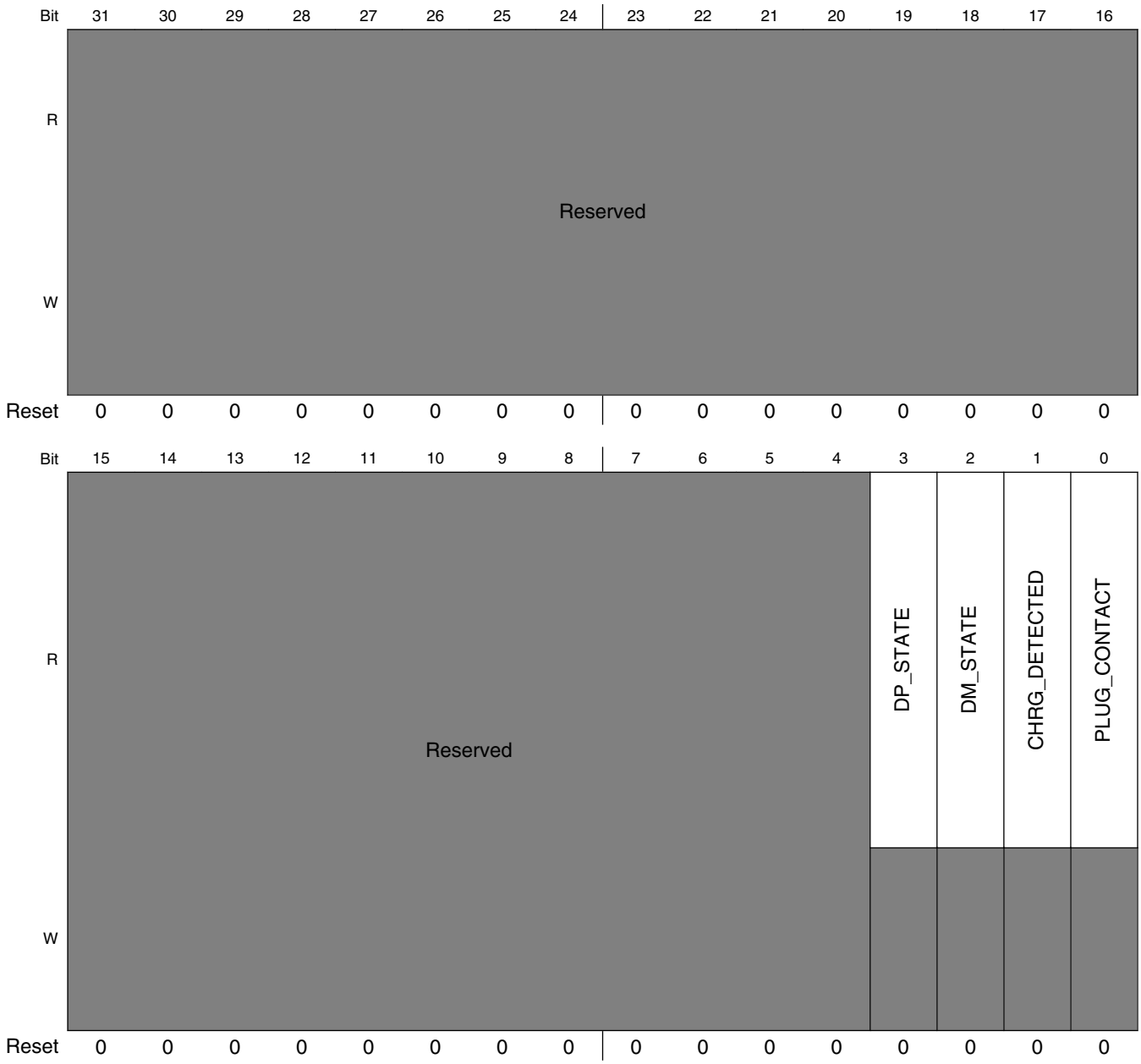
USB_ANALOG_USB2_VBUS_DETECT_STAT field descriptions (continued)

Field	Description
3 VBUS_VALID	VBus valid for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software.
2 AVALID	Indicates VBus is valid for a A-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
1 BVALID	Indicates VBus is valid for a B-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
0 SESSEND	Session End for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the SESSEND bit below. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V is present, 1 if VDD5V is not present.

55.4.9 USB Charger Detect Status Register (USB_ANALOG_USB2_CHRG_DETECT_STAT)

This register defines fields for USB charger detect status.

Address: 0h base + 230h offset = 230h



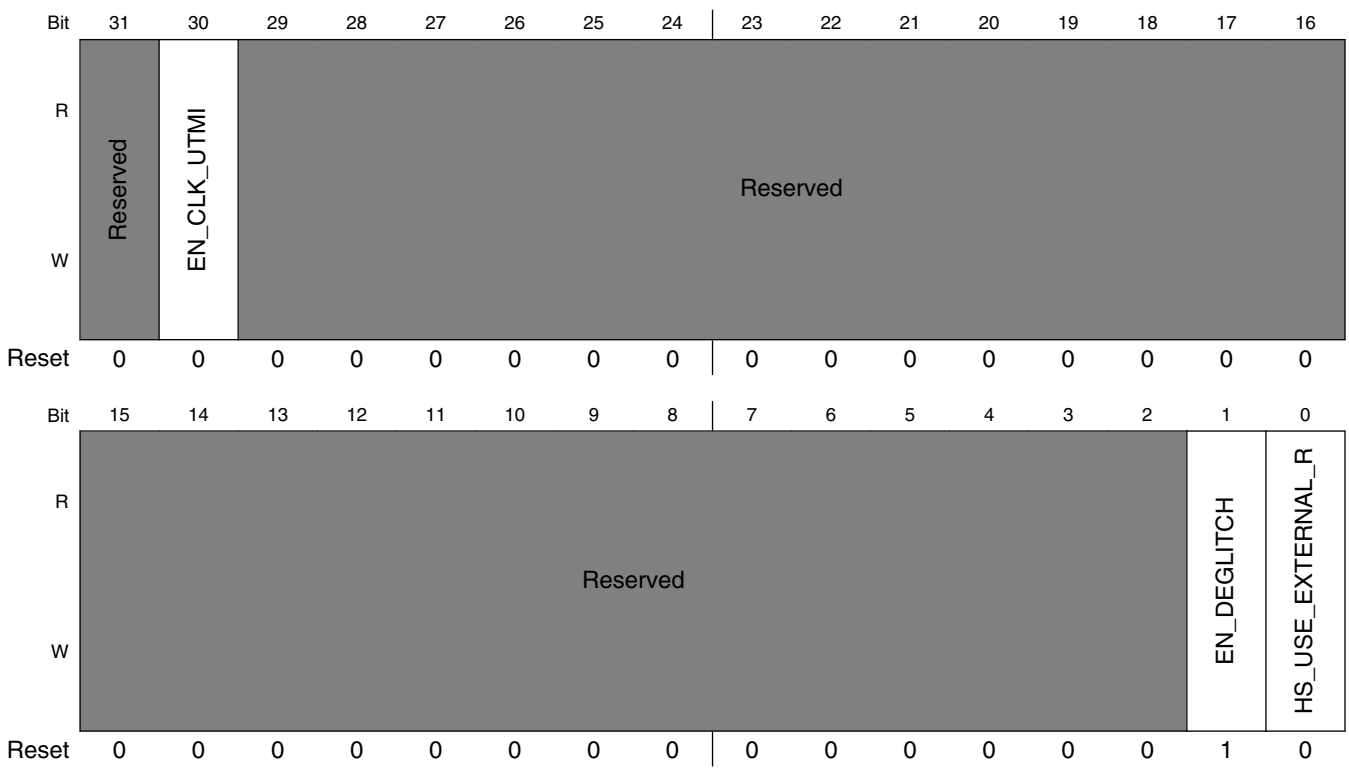
USB_ANALOG_USB2_CHRG_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.
3 DP_STATE	DP line state output of the charger detector.
2 DM_STATE	DM line state output of the charger detector.
1 CHRG_ DETECTED	State of charger detection. This bit is a read only version of the state of the analog signal. 0 **CHARGER_NOT_PRESENT** — The USB port is not connected to a charger. 1 **CHARGER_PRESENT** — A charger (either a dedicated charger or a host charger) is connected to the USB port.
0 PLUG_ CONTACT	State of the USB plug contact detector. 0 **NO_CONTACT** — The USB plug has not made contact. 1 **GOOD_CONTACT** — The USB plug has made good contact.

55.4.10 USB Misc Register (USB_ANALOG_USB2_MISC*n*)

This register defines controls for USB.

Address: 0h base + 250h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB2_MISC*n* field descriptions

Field	Description
31 -	This field is reserved. Reserved.
30 EN_CLK_UTMI	Enables the clk to the UTMI block.
29–2 -	This field is reserved. Reserved.
1 EN_DEGLITCH	Enable the deglitching circuit of the USB PLL output.
0 HS_USE_ EXTERNAL_R	Use external resistor to generate the current bias for the high speed transmitter. This bit should not be changed unless recommended by Freescale.

55.4.11 Chip Silicon Version (USB_ANALOG_DIGPROG)

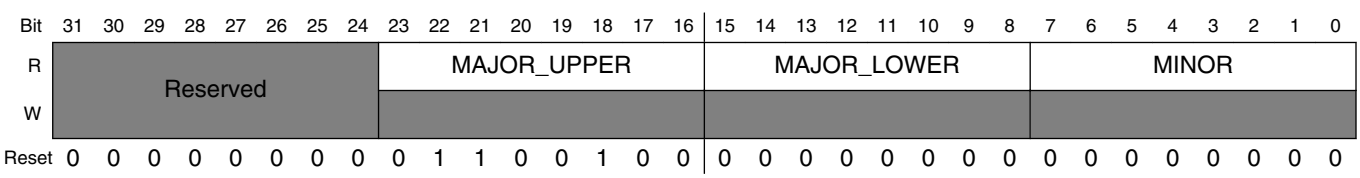
The DIGPROG register returns the digital program ID for the silicon.

For silicon revision 1.0, USB_ANALOG_DIGPROG = 0x0064_0000

For silicon revision 1.1, USB_ANALOG_DIGPROG = 0x0064_0001

For silicon revision 1.2, USB_ANALOG_DIGPROG = 0x0064_0002

Address: 0h base + 260h offset = 260h



USB_ANALOG_DIGPROG field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23–16 MAJOR_UPPER	MAJOR upper byte-Read-only value representing the chip type. 0x64 i.MX 6UltraLite
15–8 MAJOR_LOWER	MAJOR lower byte - Read-only value representing a major silicon revision. 0x00 silicon revision 1.x 0x01 silicon revision 2.x
MINOR	MINOR lower byte - Read-only value representing a minor silicon revision. 0x00 silicon revision x.0 0x01 silicon revision x.1 0x02 silicon revision x.2

USB_ANALOG_DIGPROG field descriptions (continued)

Field	Description

Chapter 56
Ultra Secured Digital Host Controller (uSDHC)

56.1 Overview

The Ultra Secured Digital Host Controller (uSDHC) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 56-1.

The uSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards.

It handles the SD/SDIO/MMC protocols at the transmission level.

The following are brief descriptions of the cards supported by the uSDHC:

The Multi Media Card (MMC) is a universal low cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly-emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, the following figure does not show cards with reduced size or mini cards.

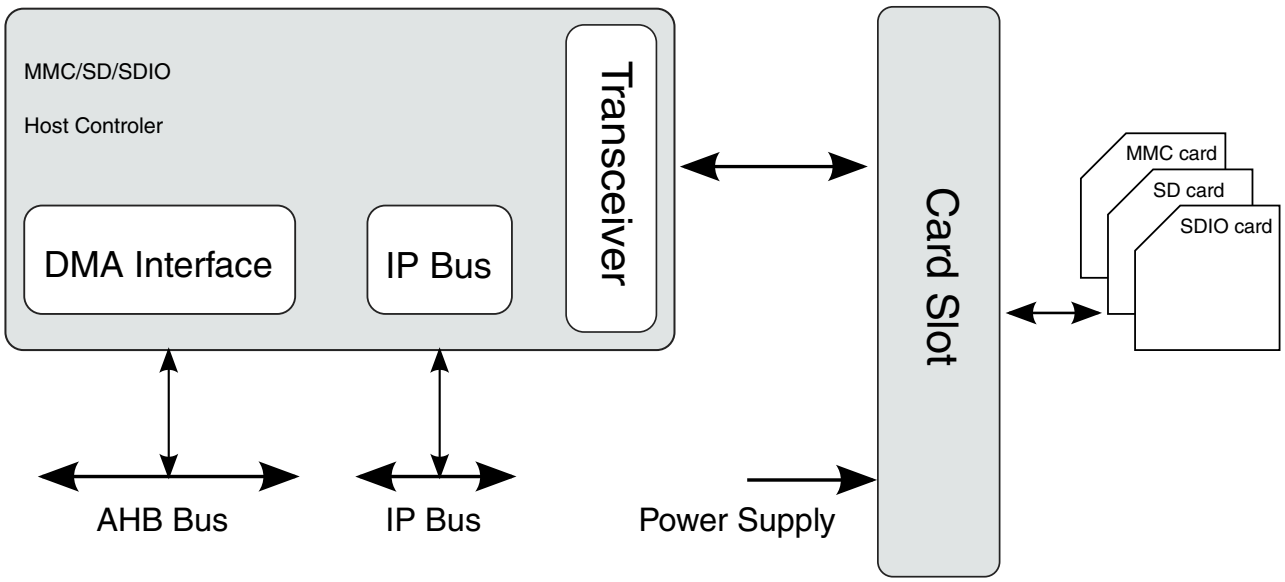


Figure 56-1. System Connection of the uSDHC

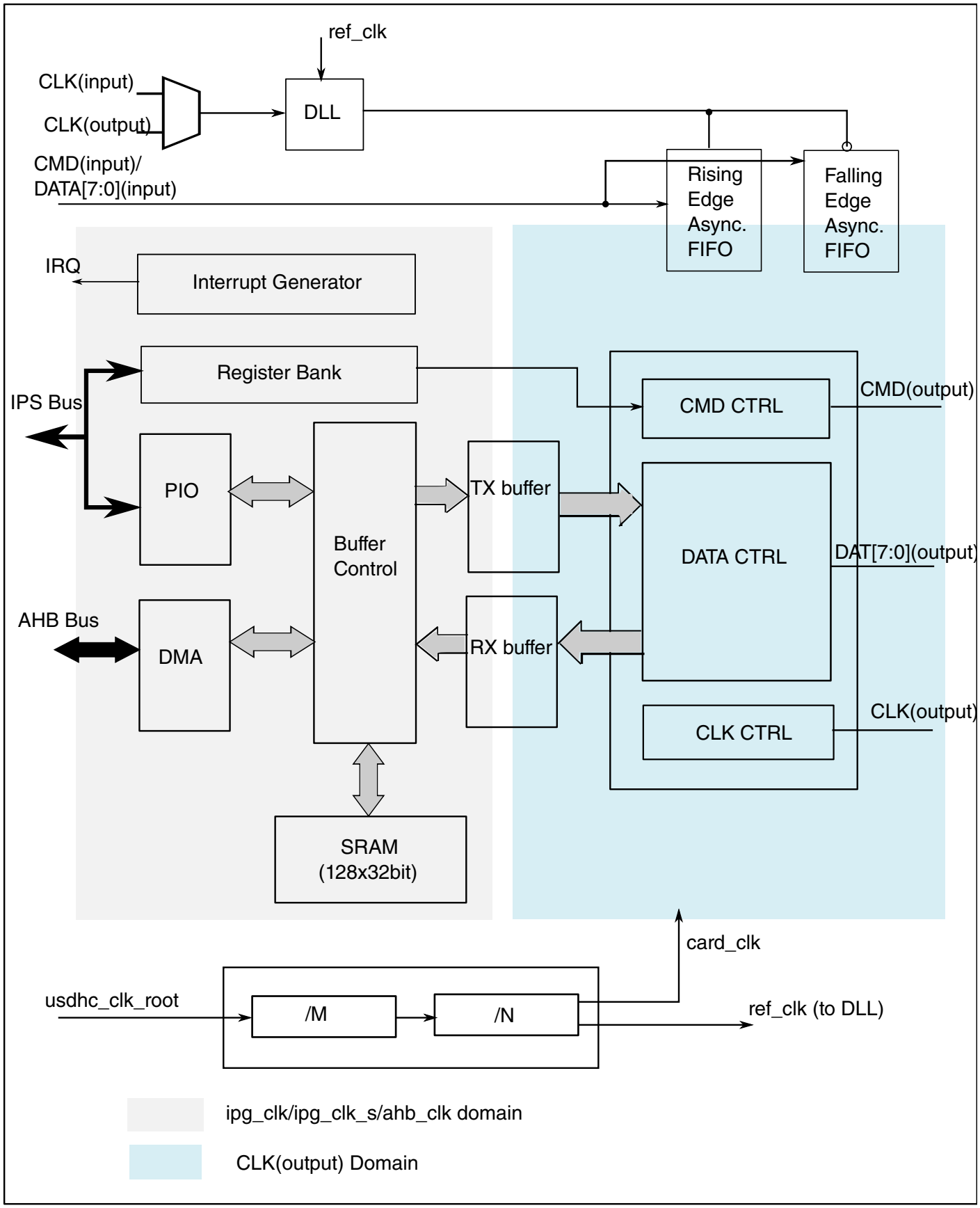


Figure 56-2. ultra Secure Digital Host Controller Block Diagram

56.1.1 Features

The features of the uSDHC module include the following:

- Conforms to the SD Host Controller Standard Specification version 3.0
- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 3.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- Card bus clock frequency up to 208 MHz
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes
    - Up to 832 Mbps of data transfer for SDIO cards using 4 parallel data lines in SDR(Single Data Rate) mode
    - Up to 400 Mbps of data transfer for SDIO card using 4 parallel data lines in DDR(Dual Data Rate) mode
    - Up to 832 Mbps of data transfer for SDXC cards using 4 parallel data lines in SDR(Single Data Rate) mode
    - Up to 400 Mbps of data transfer for SDXC card using 4 parallel data lines in DDR(Dual Data Rate) mode
    - Up to 416 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR(Single Data Rate) mode
    - Up to 832 Mbps of data transfer for MMC cards using 8 parallel data lines in DDR(Dual Data Rate) mode
- Supports single block/multi-block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal and external DMA capabilities
- Support voltage selection by configuring vendor specific register bit
- Supports Advanced DMA to perform linked memory access

56.1.2 Modes and Operations

56.1.2.1 Data transfer Modes

The uSDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification Mode (up to 400 kHz)
- MMC full speed mode (up to 26 MHz)
- MMC high speed mode (up to 52 MHz)
- MMC HS200 mode(up to 200 MHz)
- MMC DDR mode (52 MHz both edges)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)
- SD/SDIO UHS-I mode(up to 208 MHz in SDR mode, up to 50 MHz in DDR mode)

56.2 External Signals

The following table describes the external signals of USDHC:

Table 56-1. USDHC External Signals

Signal	Description	Pad	Mode	Direction
SD1_CD_B	Card detection pin If not used(for the embedded memory),tie low to indicate there is a card attached.	CSI_DATA05	ALT8	I
		GPIO1_IO03	ALT4	
		UART1_RTS_B	ALT2	
SD1_CLK	Clock for MMC/SD/SDIO card	SD1_CLK	ALT0	O
SD1_CMD	CMD line connect to card	SD1_CMD	ALT0	IO
SD1_DATA0	DATA0 line in all modes Also used to detect busy sta	SD1_DATA0	ALT0	IO
SD1_DATA1	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	SD1_DATA1	ALT0	IO
SD1_DATA2	DATA2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	SD1_DATA2	ALT0	IO

Table continues on the next page...

Table 56-1. USDHC External Signals (continued)

Signal	Description	Pad	Mode	Direction
SD1_DATA3	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	SD1_DATA3	ALT0	IO
SD1_DATA4	DATA7 line in 8-bit mode, not used in other modes	NAND_READY_B	ALT1	IO
SD1_DATA5		NAND_CE0_B	ALT1	IO
SD1_DATA6		NAND_CE1_B	ALT1	IO
SD1_DATA7		NAND_CLE	ALT1	IO
SD1_LCTL	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	ENET1_RX_DATA0	ALT8	O
SD1_RESET_B	Card hardware reset signal, active LOW	CSI_DATA06	ALT8	O
		GPIO1_IO04	ALT4	
		GPIO1_IO09	ALT6	
		NAND_WP_B	ALT1	
SD1_TESTER_TRIGGER	-	UART1_RX_DATA	ALT7	IO
SD1_VSELECT	IO power voltage selection signal	CSI_DATA07	ALT8	O
		ENET1_RX_EN	ALT8	
		GPIO1_IO05	ALT4	
SD1_WP	Card write protect detect If not used(for the embedded memory), tie low to indicate it's not write protected.	CSI_DATA04	ALT8	I
		GPIO1_IO02	ALT4	
		UART1_CTS_B	ALT2	
SD2_CD_B	Card detection pin If not used(for the embedded memory),tie low to indicate there is a card attached.	CSI_MCLK	ALT1	I
		GPIO1_IO07	ALT4	
		UART1_RTS_B	ALT8	
SD2_CLK	Clock for MMC/SD/SDIO card	CSI_VSYNC	ALT1	O
		LCD_DATA19	ALT8	
		NAND_RE_B	ALT1	
SD2_CMD	CMD line connect to card	CSI_HSYNC	ALT1	IO
		LCD_DATA18	ALT8	
		NAND_WE_B	ALT1	
SD2_DATA0	DATA0 line in all modes Also used to detect busy sta	CSI_DATA00	ALT1	IO
		LCD_DATA20	ALT8	
		NAND_DATA00	ALT1	
SD2_DATA1	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	CSI_DATA01	ALT1	IO
		LCD_DATA21	ALT8	
		NAND_DATA01	ALT1	
SD2_DATA2	DATA2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	CSI_DATA02	ALT1	IO
		LCD_DATA22	ALT8	

Table continues on the next page...

Table 56-1. USDHC External Signals (continued)

Signal	Description	Pad	Mode	Direction
		NAND_DATA02	ALT1	
SD2_DATA3	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	CSI_DATA03	ALT1	IO
		LCD_DATA23	ALT8	
		NAND_DATA03	ALT1	
SD2_DATA4	DATA7 line in 8-bit mode, not used in other modes	CSI_DATA04	ALT1	IO
		LCD_DATA14	ALT8	
		NAND_DATA04	ALT1	
SD2_DATA5		CSI_DATA05	ALT1	IO
		LCD_DATA15	ALT8	
		NAND_DATA05	ALT1	
SD2_DATA6		CSI_DATA06	ALT1	IO
		LCD_DATA16	ALT8	
		NAND_DATA06	ALT1	
SD2_DATA7		CSI_DATA07	ALT1	IO
		LCD_DATA17	ALT8	
		NAND_DATA07	ALT1	
SD2_LCTL	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	ENET1_RX_DATA1	ALT8	IO
SD2_RESET_B	Card hardware reset signal, active LOW	GPIO1_IO09	ALT4	O
		LCD_DATA13	ALT8	
		NAND_ALE	ALT1	
SD2_TESTER_TRIGGER	-	UART1_CTS_B	ALT7	IO
SD2_VSELECT	IO power voltage selection signal	ENET1_TX_DATA0	ALT8	IO
		GPIO1_IO08	ALT4	
SD2_WP	Card write protect detect If not used(for the embedded memory), tie low to indicate it's not write protected.	CSI_PIXCLK	ALT1	I
		GPIO1_IO06	ALT4	
		UART1_CTS_B	ALT8	

56.2.1 Signals Overview

The uSDHC has 14 associated I/O signals.

- The CLK is an internally generated clock used to drive the MMC, SD, SDIO cards.
- The CMD I/O is used to send commands and receive responses to and from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the uSDHC and the card.

- The CD and WP are card detection and write protection signals directly routed from the socket. These two signals are active low (0). A low on CD# means that a card is inserted, and a high on WP means that the write protect switch is active.
- LCTL is an output signal used to drive an external LED to indicate that the SD interface is busy.
- RST is an output signal used to reset the MMC card.
- VSELECT is an output signal used to change the voltage of the external power supplier.

CD, WP, LCTL, RST and VSELECT are all optional for system implementation. If the uSDHC needs to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.

56.3 Clocks

The table found here describes the clock sources for uSDHC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 56-2. uSDHC Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	AHB bus clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_perclk	usdhc_clk_root	Base clock
ipg_clk_s	ipg_clk_root	Peripheral access clock for register accesses

56.4 Functional Description

The following sections provide a brief functional description of the major system blocks, including the Data Buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock & reset manager and clock generator.

56.4.1 Data Buffer

The uSDHC uses one configurable data buffer to transfer data between the system bus (IP Bus or AHB Bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock).

The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable and can be from 1 to 128 words. The burst lengths for read and write are also configurable and can be from 1 to 31 words.

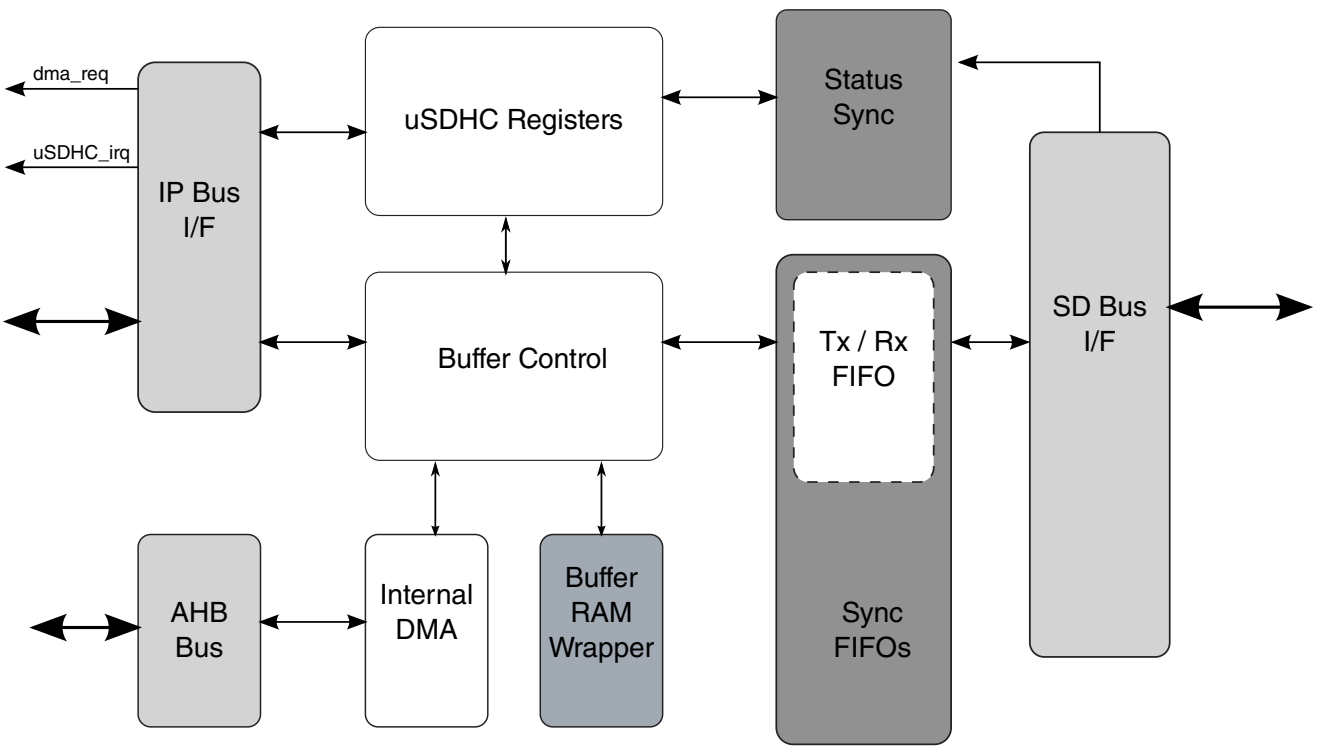


Figure 56-3. uSDHC Buffer Scheme

There are 3 transfer modes to access the data buffer:

- CPU polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the Host Driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar.
- External DMA mode:
  - For a read operation, when there are more words received in the buffer than the amount set in the RD_WML register, a DMA request is sent out to inform the external DMA to fetch the data. The request will be immediately de-asserted when there is an access on the Buffer Data Port register. If the number of words

in the buffer after the current burst meets or exceeds RD_WML value, the DMA request is asserted again. For instance, if there are twice as many words in the buffer as there are in the RD_WML value, there are two successive DMA requests with only one cycle of de-assertion between. The write operation is similar. Note the accesses CPU polling mode and external DMA mode both use the IP bus, and if the external DMA is enabled, in both modes an external DMA request is sent when the buffer is ready.

- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus. For internal DMA access mode, the external DMA request will never be sent out.

For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

- Burst length configured in the burst length field of the Watermark Level register
- Watermark Level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 Kbyte address boundary defined in the AHB protocol

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, byte order is swapped before the data is stored in the buffer.

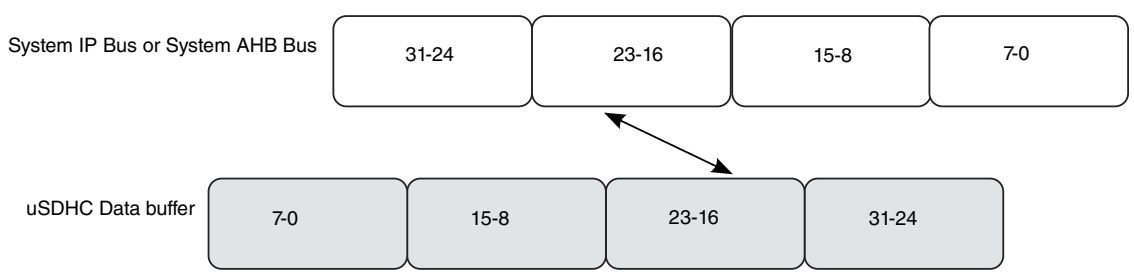


Figure 56-4. Data Swap between System Bus and uSDHC Data Buffer in Byte Little Endian Mode

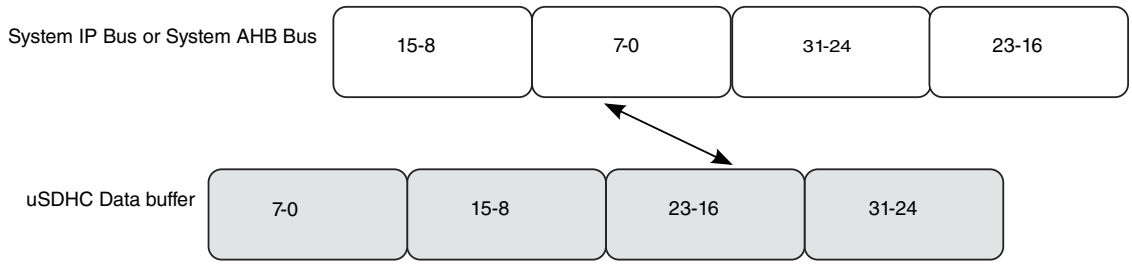


Figure 56-5. Data Swap between System Bus and uSDHC Data Buffer in Half Word Big Endian Mode

56.4.1.1 Write Operation Sequence

There are three ways to write data into the buffer when the user transfers data to the card:

- External DMA through the uSDHC DMA request signal
- Processor core polling through the BWR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When the internal DMA is not used, (the DMAEN bit in the Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register, and is ready for receiving new data. At the same time, the uSDHC sets the BWR bit. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the write operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

The uSDHC will not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the Host System does not write data in time, the uSDHC will stop the CLK to avoid the data buffer under-run situation.

56.4.1.2 Read Operation Sequence

There are three ways to read data from the buffer when the user transfers data to the card:

- External DMA through the uSDHC DMA request signal
- Processor core polling through the BRR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, that is available and ready for system fetching data. At the same time, the uSDHC sets the BRR bit. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the read operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

For any write transfer mode, the uSDHC will not start data transmission until the number of words set in the RD_WML register are in the buffer. If the buffer is full and the Host System does not read data in time, the uSDHC will stop the CLK to avoid the data buffer over-run situation.

56.4.1.3 Data Buffer and Block Size

The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way. In the uSDHC, the only data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly.

For both read and write, the watermark level can be from 1 to 128 words. For both read and write the burst length can be from 1 to 31 words. The Host Driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes).

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned) stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write two times for each block. For each block the ending byte will be abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

56.4.1.4 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the Host Driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See the figure below for an example showing the dividing of large data transfers, assuming a kind of WLAN SDIO card that only supports a block size up to 64 bytes. Although the uSDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than 64 bytes, so the data must be divided (see example below).

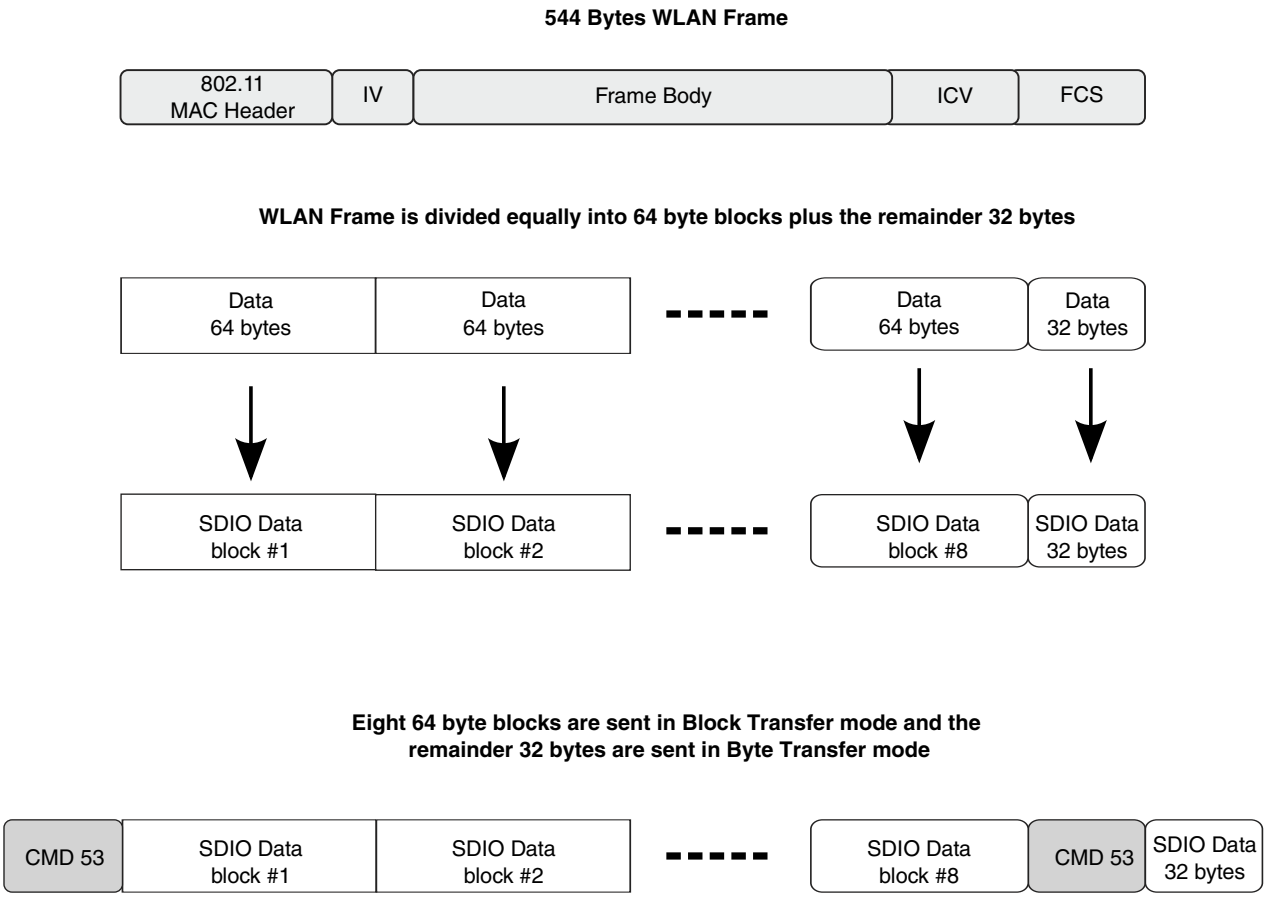


Figure 56-6. Example for Dividing Large Data Transfers

56.4.1.5 External DMA Request

When the internal DMA is not in use and external DMA is enabled, the Data Buffer will generate a DMA request to the system. During a write operation, when the number of WR_WML words can be held in the buffer free space, the signal uSDHC_dreq_b is asserted to 0, informing the Host System of a DMA write.

The BWR bit in the Interrupt Status register is also set, as long as the BWRSEN bit in the Interrupt Status Enable register is set. The DMA request is de-asserted after several accesses to the Data Port register are made while the buffer's free space can't meet the watermark condition (free space> write watermark level).

On read operation, when the number of RD_WML words are already in the buffer, the signal uSDHC_dreq_b is asserted to 0, informing the Host System for a DMA read. The BRR bit in the Interrupt Status register is also set, as long as the BRRSEN bit in the

Interrupt Status Enable register is set. The DMA request is de-asserted after several accesses to the Data Port register are made while the buffer's data can't meet the watermark condition (the number of data in buffer > read watermark level).

If the DMA burst length can't change during a data transfer for an external DMA transfer, the watermark level (read or write) must be a divisor of the block size. If it is not, transferring the block may cause buffer under-run (read operation) or over-run (write operation). For example, if the block size is 512 bytes, the watermark level of read (or write) must be a power of two between 1 and 128. For processor core polling access there is no such issue, as the last access in the block transfer can be controlled by software. The watermark level can be any value, even larger than the block size (but no greater than 128 words) because the actual number of bytes transferred by the software can be controlled and does not exceed the block size in each transfer.

The uSDHC also supports non-word aligned block size, as long as the card supports that block size. In this case, the watermark level should be set as the number of words. For example, if the block size is 31 bytes, the watermark level can be set to any number of words. For this case, the BLKSIZE bits of the Block Attribute register will be set as 1fh. For the CPU polling access, the burst length can be 1 to 128 words, without restriction. This is because the software will transfer 8 words, and the uSDHC will also set the BWR or BRR bits when the remaining data does not violate data buffer. See DMA Burst Length for more details about the dynamic watermark level of the data buffer.

For the above example, even though 8 words are transferred via the Data Port register, the uSDHC will transfer only 31 bytes over the SD Bus, as required by the BLKSIZE bits. In this data transfer, with non-word aligned block size, the endian mode should be set cautiously or invalid data will be transferred to and from the card.

56.4.2 DMA AHB Interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the uSDHC_dreq_b will not be asserted during the transfer, but the BWR and BRR bits will be set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register.

See the figure below for an illustration of the DMA AHB interface block.

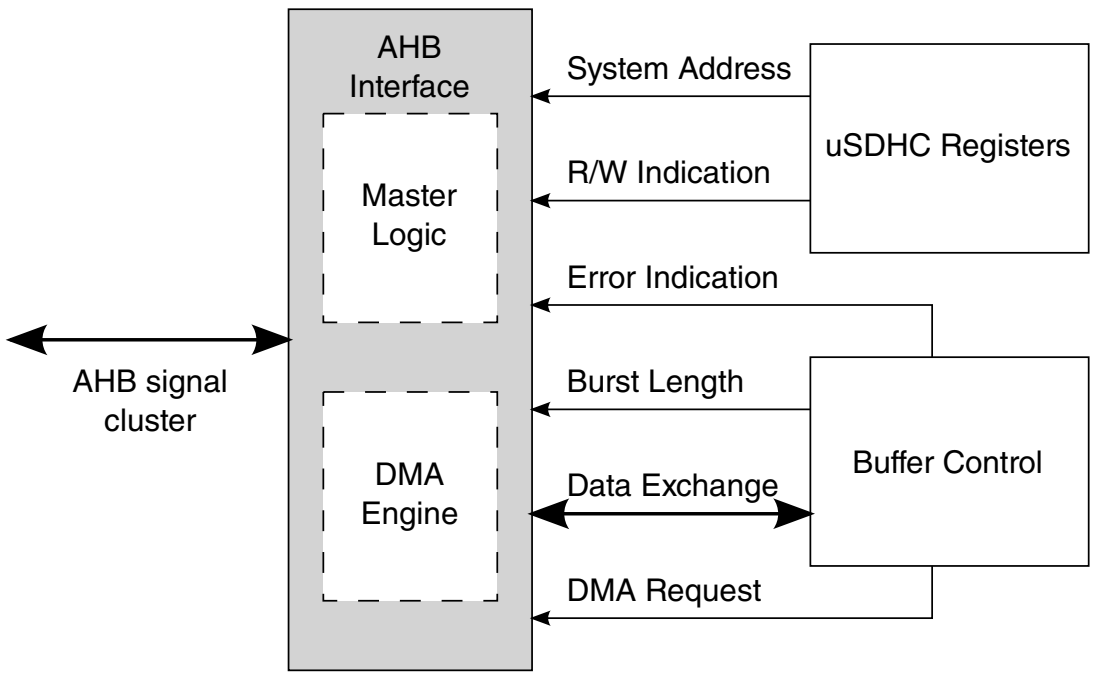


Figure 56-7. DMA AHB Interface Block

56.4.2.1 Internal DMA Request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the Data Buffer block will send a DMA request to AHB interface. Meanwhile, the external DMA request signal (uSDHC_dreq_b) is disabled.

The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to the uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The Data Buffer de-asserts the request if the data buffer space(for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA,

the Data Buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer (which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The uSDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuffed byte.

56.4.2.2 DMA Burst Length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block.

See the example in Internal DMA Request. After 6 words are read, the burst length will be 2 words, then the next burst length will be 6 words. This is because the next block starts, which is 31 bytes, more than 6 words. The Host Driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

56.4.2.3 AHB Master Interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state.At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register will be generated to host CPU to report a bus error condition.

Once the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and re-start the transfer from this address to recover the corrupted block. DMA operation will resume when the interrupt is serviced by software.

56.4.2.4 ADMA Engine

In the SD Host Controller Standard, a new DMA transfer algorithm called the ADMA (Advanced DMA) is defined. For Simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the Host Driver.

The ADMA defines the programmable descriptor table in the system memory. The Host Driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized since the Host MCU intervention would not be needed during long DMA based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in Host Controller. ADMA1 can support data transfer of 4KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors define in SD Host Controller Standard, and if 'End' flag is detected in the descriptor, ADMA will stop after this descriptor is processed.

56.4.2.4.1 ADMA Concept and Descriptor Format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Set data length descriptor.
- Set data address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Rsv descriptor.
- Set data length & address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

See Figure 56-8 for the format of the descriptor table for ADMA1.

Figure 56-11 explains the ADMA2 format. ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it will ignore the high 32-bit. Address field shall be set on word aligned(lower 2-bit is always set to 0). Data length is in byte unit.

ADMA will start read/write operation after it reaches the Tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last Set type descriptor before Tran type descriptor. Every Tran type will trigger a transfer, and the transfer data length is extracted from the most recent Set type descriptor. If there is no Set type descriptor after the previous Trans descriptor, the data length will be the value for previous transfer, or 0 if no Set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address, so only a Tran type descriptor can start a data transfer

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 56-8. Format of the ADMA1 Descriptor Table

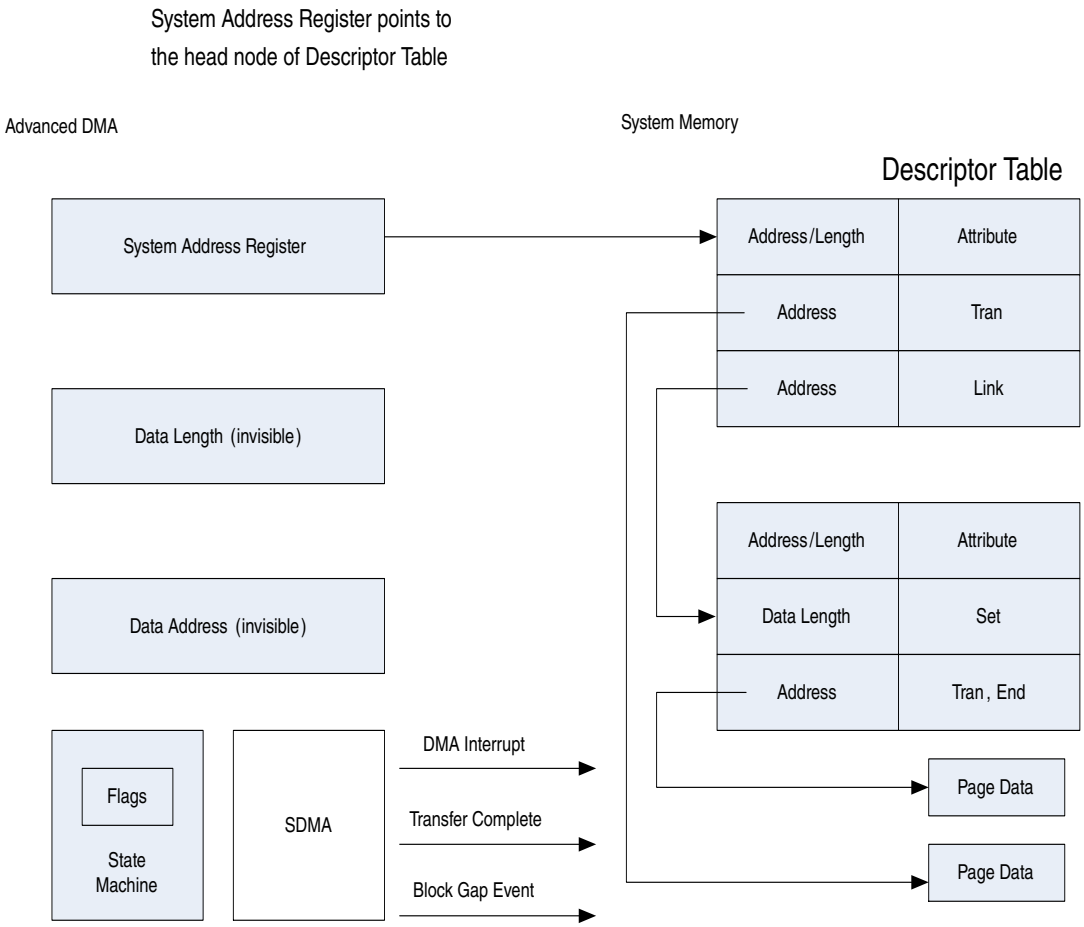


Figure 56-9. Concept and Access Method of ADMA1 Descriptor Table

Address Field		Length		Reserved		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 56-10. Format of the ADMA2 Descriptor Table

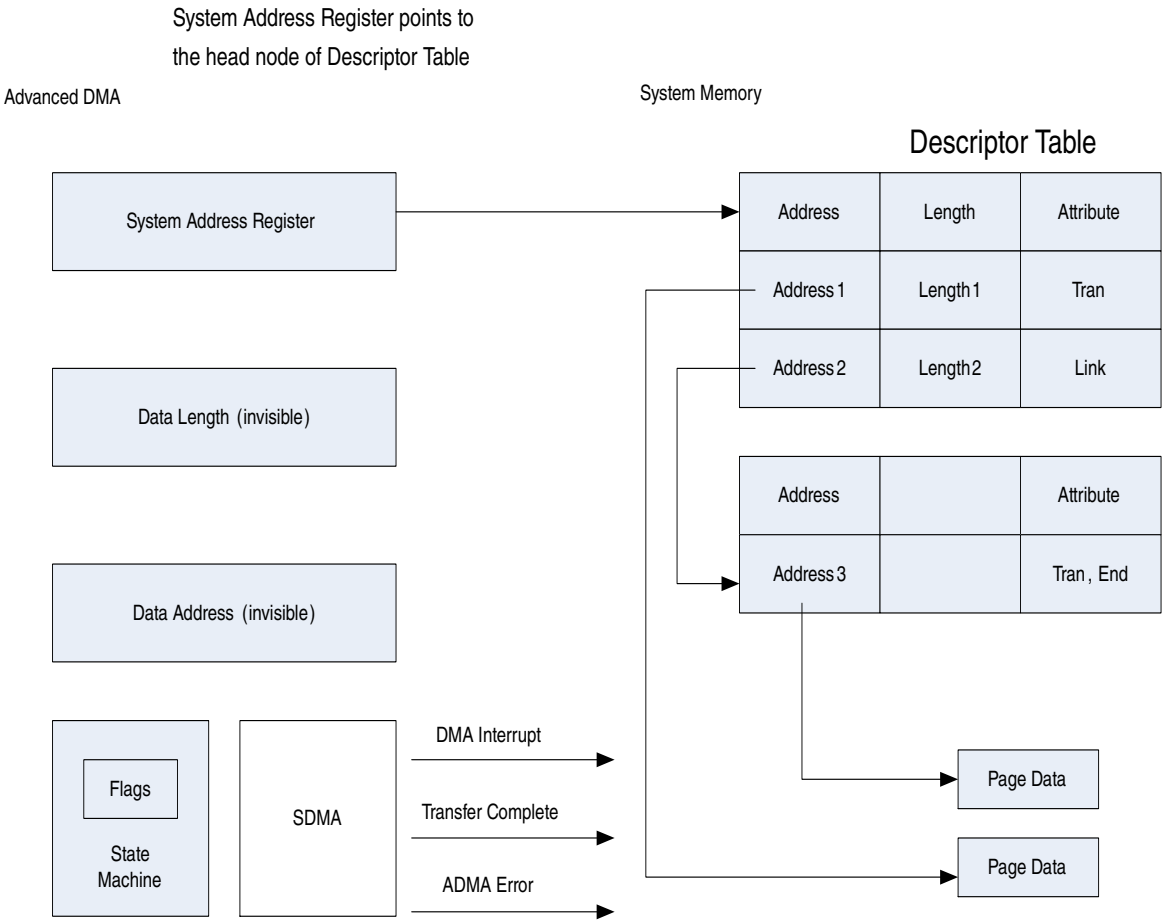


Figure 56-11. Concept and Access Method of ADMA2 Descriptor Table

56.4.2.4.2 ADMA Interrupt

If the interrupt flag descriptor is set, ADMA will generate an interrupt according to various types of descriptors:

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

56.4.2.4.3 ADMA Error

The ADMA will stop whenever any error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

An ADMA descriptor error will be generated when it fails to detect a 'Valid' flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in buffer must be equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

When BLKCNTEN bit is not set, then whole data length set in descriptor should be a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors will occur.

56.4.3 Register Bank with IP Bus Interface

Register accesses via the IP Bus interface are actually on the Register Bank.

See Figure 56-12 below for the block diagram.

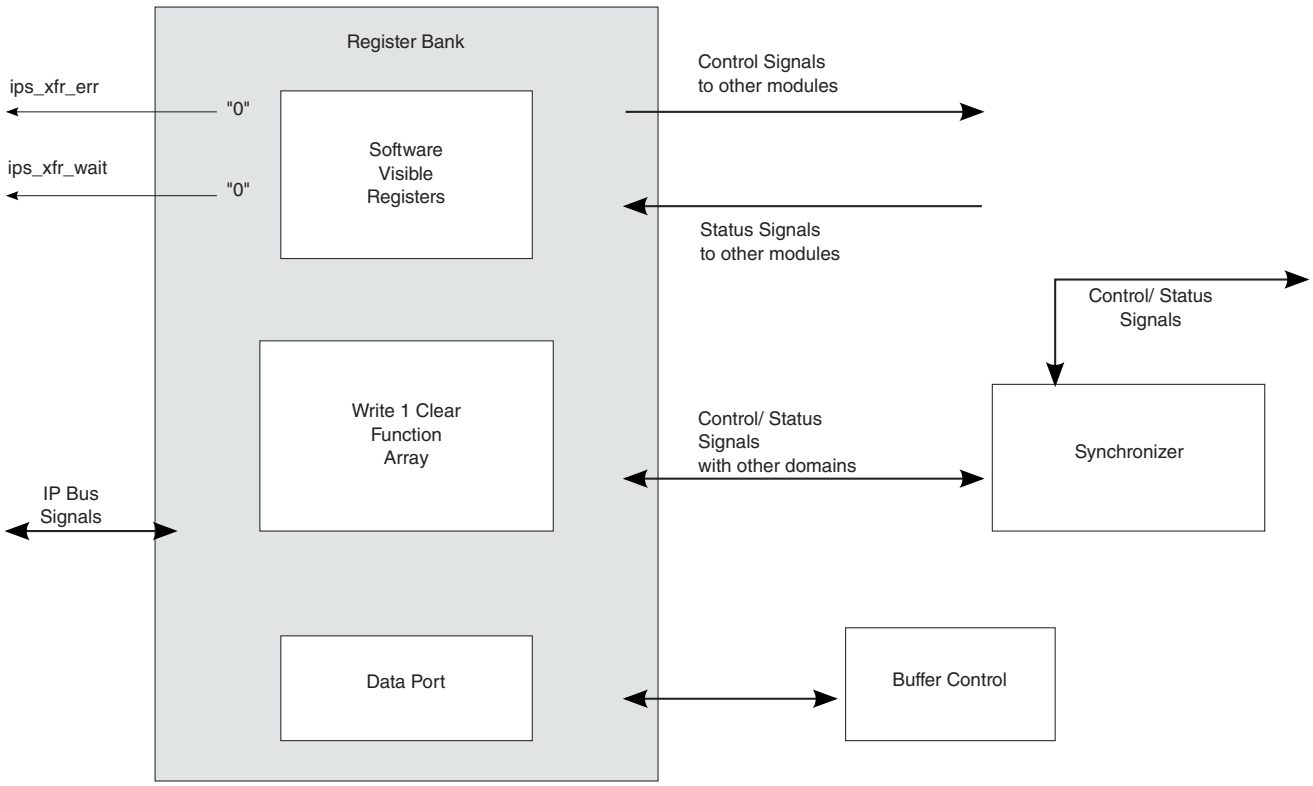


Figure 56-12. Register Bank Diagram

Only 32-bit access is allowed, and no partial read / write is supported, thus all accesses are word aligned.

56.4.3.1 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs.

The SD Protocol Unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD Protocol Unit consists of four sub modules:

1. SD control misc.

2. Command control.
3. Data control.
4. Clock control

56.4.3.2 SD control misc

In the SD control misc unit, the card detect(include the CD_B and DATA3 used as Card Detection), write protection and card interrupt are implemented.

This module monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the Register Bank. The driver can use this for debug purposes.

The module also detects the WP (Write Protect) line. If WP is active, writes to the register bank will be ignored.

This module also drives the LCTL output signal when the LCTL bit is set by the driver.

56.4.3.3 SD Clock control

If the internal data buffer is near full(for read) or near empty(for write), the SD clock must be gated off to avoid buffer over/under-run, this module will assert the gate of the output SD clock to shut the clock off. After the buffer has space(for read) or has data(for write), the clock gate of this module will open and the SD clock will be active again.

56.4.3.4 Command control

The Command Control module deals with the transactions on the CMD line.

See the figure below for an illustration of the structure for the Command CRC Shift Register.

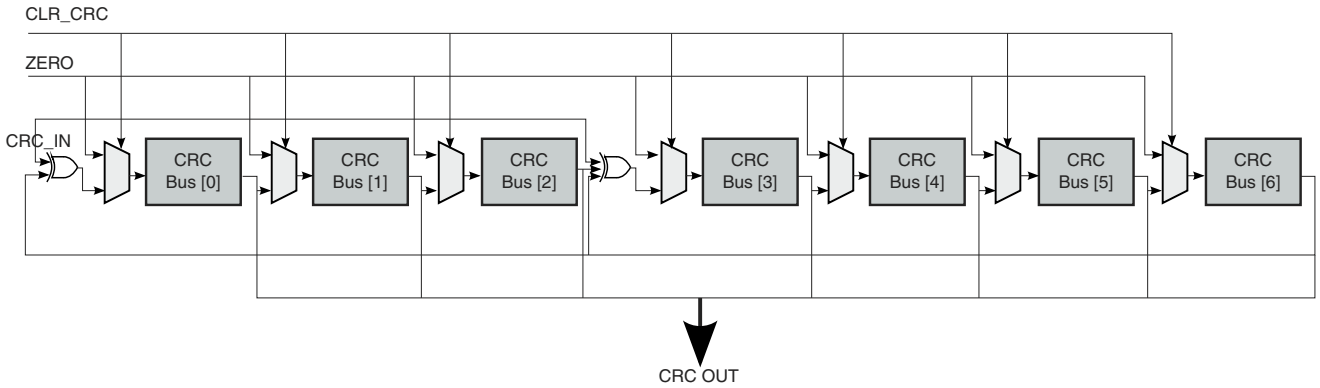


Figure 56-13. Command CRC Shift Register

The CRC polynomials for the CMD are as follows:

```
Generator polynomial: G(x) = x^7 + x^3 + 1
M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0
CRC[6:0] = Remainder [(M(x) * x^7) / G(x)]
```

56.4.3.5 Data control

The Data Agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line, and generates the Read Wait state by the request from the Transceiver.

The CRC polynomials for the DATA are as follows:

```
Generator polynomial: G(x) = x^16 + x^12 + x^5 +1
M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0
CRC[15:0] = Remainder [(M(x) * x^16) / G(x)]
```

56.4.4 Clock & Reset Manager

This module controls all the reset signals within the uSDHC.

There are four kinds of reset signals within uSDHC:

1. Hardware reset.
2. Software reset for all logic.
3. Software reset for the data logic.
4. Software reset for the command logic.

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

56.4.5 Clock Generator

The Clock Generator generates the card CLK by peripheral source clock in two stages.

Refer to the figure below for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

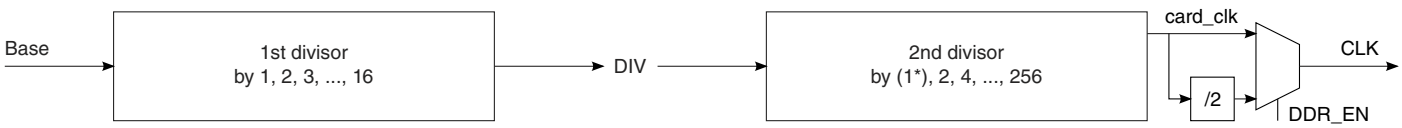


Figure 56-14. Two Stages of the Clock Divider

The first stage outputs an intermediate clock (DIV), which can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler, and outputs the actual internal working clock (card_clk). This clock is the driving clock for all sub modules of the SD Protocol Unit, and the sync FIFOs (see Figure 56-3) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can be DIV, DIV/2, DIV/4,..., or DIV/256. Thus the highest frequency of the card_clk is Base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of Base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

Please note, in SDR mode and DDR mode, the CLK are different.

- In SDR mode, CLK is equal to the internal working clock(card_clk).

- In DDR mode, CLK is equal to the card_clk/2.

56.4.6 SDIO Card Interrupt

Information on Interrupts in 1-bit Mode, Interrupts in 4-bit Mode, and Card Interrupt Handling are detailed in the sections below.

56.4.6.1 Interrupts in 1-bit Mode

In this case the DATA1 pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DATA1 low from the SDIO card, until the interrupt service is finished to clear the interrupt.

56.4.6.2 Interrupt in 4-bit Mode

Since the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will only be sent by the card and recognized by the host during a specific time. This is known as the Interrupt Period. The uSDHC will only sample the level on Pin 8 during the Interrupt Period. At all other times, the host will ignore the level on Pin 8, and treat it as the data signal. The definition of the Interrupt Period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the Interrupt Period becomes active two clock cycles after the completion of a data packet. This Interrupt Period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the Interrupt Period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the Interrupt Period. For this case, the Interrupt Period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line will be held low for one clock cycle with the last clock cycle pulling DATA1 high. On completion of the Interrupt Period, the card releases the DATA1 line into the high Z state. The uSDHC samples the DATA1 during the Interrupt Period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

56.4.6.3 Card Interrupt Handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, the uSDHC clears the interrupt request to the Host System. The Host Driver should clear this bit before servicing the SDIO Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Card Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. In 1-bit mode, the uSDHC will detect the SDIO Interrupt with or without the SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the Interrupt Period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the

SDIO status has been set, and the Host Driver needs to service this interrupt, so the SDIO bit in the Interrupt Control Register of SDIO card will be cleared. This is required to clear the SDIO interrupt status latched in the uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The Host Driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1, and the uSDHC starts sampling the interrupt signal again.

See the figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

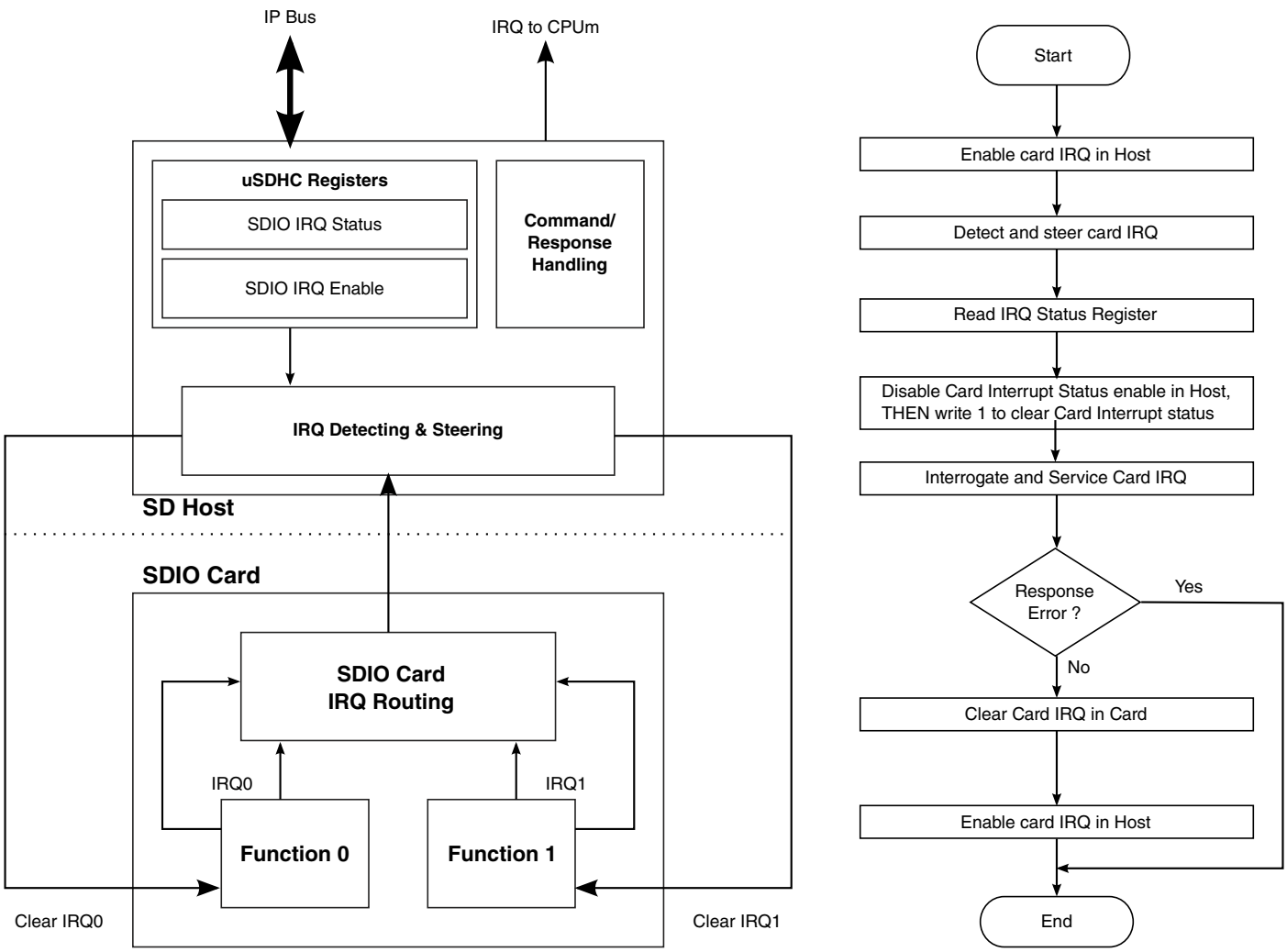


Figure 56-15. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure

56.4.7 Card Insertion and Removal Detection

The uSDHC uses either the DATA3 pin or the CD_B pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DATA3 will be pulled to a low voltage level by default.

When any card is inserted to or removed from the socket, the uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt. When the DATA3 pin is not used for card detection (for example, it is implemented in GPIO), the CD_B pin must be connected for card detection. Whether DATA3 is configured for card detection or not, the CD_B pin is always a reference for card detection. Whether the DATA3 pin or the CD_B pin is used to detect card insertion, the uSDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

56.4.8 Power Management and Wake Up Events

When there is no operation between the uSDHC and the card through the SD bus, the user can completely disable the ipg_clk and ipg_perclk in the chip level clock control module to save power. When the user needs to use the uSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the uSDHC are disabled, for instance, when the system is in low power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The uSDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

1. Card Removal Interrupt
2. Card Insertion Interrupt
3. Interrupt from SDIO card

The uSDHC offers a power management feature. By clearing the clock enabled bits in the System Control Register, the clocks are gated in the low position to the uSDHC. For maximum power saving, the user can disable all the clocks to the uSDHC when there is no operation in progress.

These three wake up events (or wakeup interrupts) can also be used to wake up the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to the uSDHC are disabled or when the whole system is in low power mode, the corresponding wakeup enabled bit needs to be

set. Refer to Protocol Control (uSDHC_PROT_CTRL) for more information on the uSDHC Protocol Control register.

56.4.8.1 Setting Wake Up Events

For the uSDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters sleep mode.

Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active
- Data and Command lines are not active
- No interrupts are pending
- Internal data buffer is empty

56.4.9 MMC fast boot

The Embedded MultiMediaCard (eMMC4.3) specification adds a fast boot feature which requires hardware support. There are two types of fast boot mode, boot operation, and alternative boot operation in the eMMC4.3 specification. Each type also has with-acknowledge and without-acknowledge modes.

In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

NOTE

For the eMMC4.3 card setting, please see the eMMC4.3 specification.

56.4.9.1 Boot operation

NOTE

For the purposes of this documentation, fast boot is called "normal fast boot mode".

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line LOW to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode with the CMD line HIGH.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

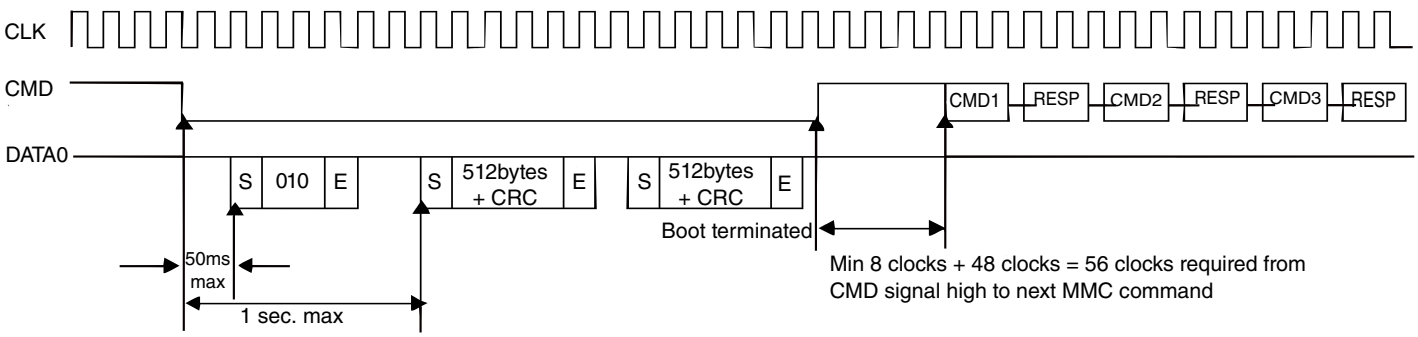


Figure 56-16. MultiMediaCard state diagram (normal boot mode)

56.4.9.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

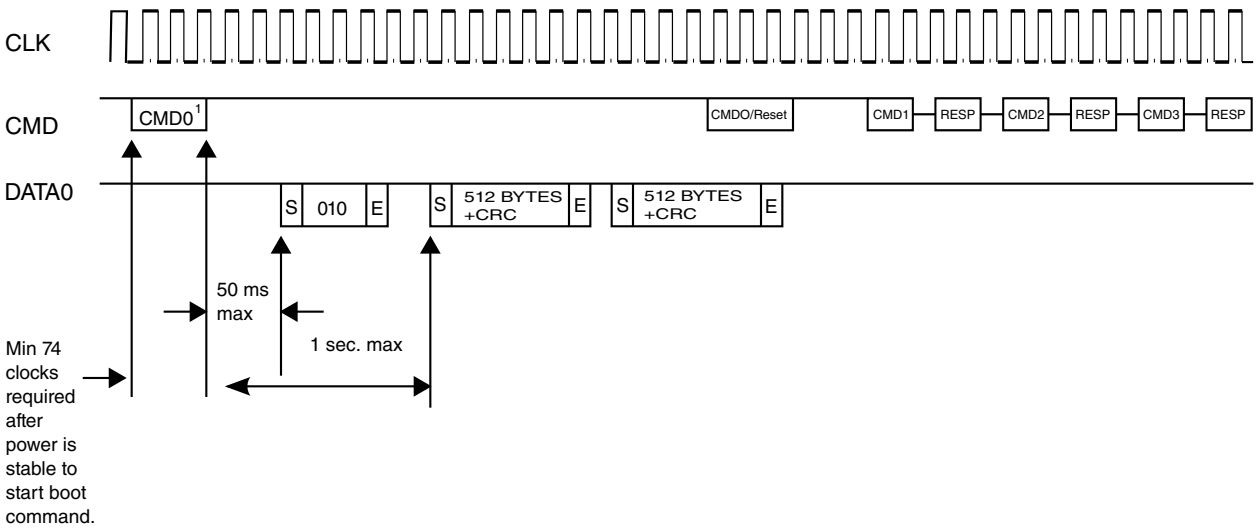
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, theslave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



NOTE 1. CMD0 with argument 0xFFFFFFFA

Figure 56-17. MultiMediaCard state diagram (alternative boot mode)

56.5 Initialization/Application of uSDHC

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, ALL_SEND_CID. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DATA I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

56.5.1 Command Send & Response Receive Basic Operation

Assuming the data type WORD is an unsigned 32-bit integer, the below flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
          set MSBSEL bit;
          if (finite block number) {
                    set BCEN bit;
                    if (auto12 command is to use) set AC12EN bit;
          }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the Standby State, no response to the Host when CMD2 is sent. The Host Driver will deal with "fake" errors like this with caution.

56.5.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards.

56.5.2.1 Card Detect

See the figure below for a flow diagram showing the detection of MMC, SD and SDIO cards using the uSDHC.

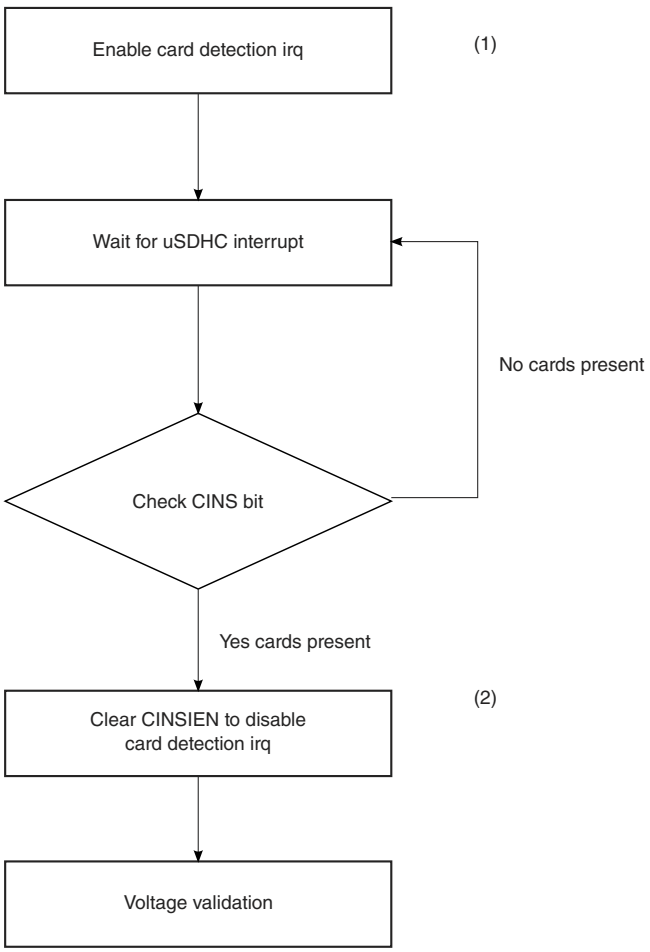


Figure 56-18. Flow Diagram for Card Detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

56.5.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) which is driven by POR (Power On Reset)

- Software reset (Host Only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the Host Controller, respectively
- Card reset (Card Only). The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards, SD Memory cards. This command sets each card into the Idle State regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both the uSDHC and the card.

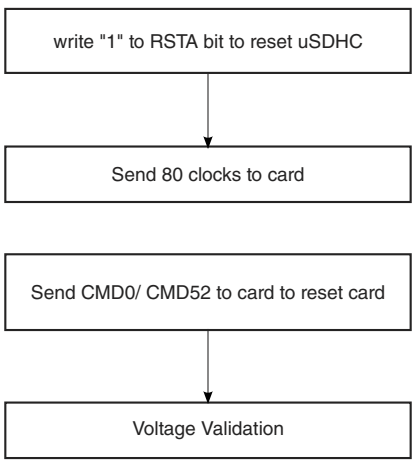


Figure 56-19. Flow Chart for Reset of the uSDHC and SD I/O Card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

56.5.2.3 Voltage Validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for Vdd are defined in the Operation Conditions Register (OCR) and may not cover the whole range.

Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
          if (0 < number of IO functions) {
                    label the card as SDIO;
                    IORDY = 0;
                    while (!(IORDY in IO OCR response)) { // set voltage range for each IO
function
                              send_command(IO_SEND_OP_COND, <voltage range>, <other
parameter>);
                              wait_for_response(IO_SEND_OP_COND);
                    } // end of while ...
          } // end of if (0 < ...
          if (memory part is present inside SDIO card) Label the card as SDCombo; // this is
an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range
is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific
CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
          send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage
range
for memory part or SD card
          wait_for_response(SD_APP_OP_COND); // voltage range is set
          if (card type is UNKNOWN) label the card as SD;
          return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
          deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card  if (card is already labelled as SDCombo)
{ //
change label
```

```
                    re-label the card as SDIO;
                    ignore the error or report it;
                    return; // card is identified as SDIO card
          } // of if (card is ...
          send_command(SEND_OP_COND, <voltage range>, <...>);
          if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
either
                    label the card as UNKNOWN;
                    return;
          } // of if (RESP_TIMEOUT ...
} // of else
}
```

56.5.2.4 Card Registry

Card registry for the MMC and SD/SDIO/SD Combo cards are different. For the SD Card, the Identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the Card spec). At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions.

The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the Inactive State. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification State.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. Once the RCA is received, the card changes its state to the Standby State. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards

(the cards in Ready State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For operation as MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
          if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                    send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
publish its
RCA
                    retrieve RCA from response;
          } // end if (card is labelled as SDCombo ...
          else if (card is labelled as SD) { // for SD card
                    send_command(ALL_SEND_CID, <...>);
                    if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                    send_command(SET_RELATIVE_ADDR, <...>);
                    retrieve RCA from response;
          } // else if (card is labelled as SD ...
          else if (card is labelled as MMC) {
                    send_command(ALL_SEND_CID, <...>);
                    rca = 0x1; // arbitrarily set RCA, 1 here for example
                    send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper
16
bits
          } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

56.5.3 Card Access

Information about Block Write, Block Read, Suspense Resume, ADMA Usage, Transfer Error, and Card Interrupt are detailed in the sections below.

56.5.3.1 Block Write

Information on Normal Write, DDR Write, and Write with Pause are detailed in the sections below.

56.5.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the DATA line. The transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DATA line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DATA to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:

- For SD/MMC cards, use SET_BLOCKLEN (CMD16)
- For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

56.5.3.1.2 DDR Write

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described as below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

56.5.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the Host Driver, the Driver can set the Stop At Block Gap Request(SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Like in the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The Driver shall read the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card. This is because when such a command is sent, the uSDHC thinks the System will switch to another function on the SDIO card, and flush the data buffer. The uSDHC takes the Resume Command as a normal command with data transfer, and it is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to

send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

56.5.3.2 Block Read

Information about Normal Read, DDR Read, Read with Pause, and DLL (Delay Line) in Read Path are detailed in the sections below.

56.5.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

56.5.3.2.2 DDR Read

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

56.5.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks.

Before setting the SABGREQ bit, make sure the RWCTL bit in the Protocol Control register is set, otherwise the uSDHC will not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit once the Read Wait capability of the SDIO card is recognized.

Like in the flow described in Normal Read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

5. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
6. Set the uSDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the uSDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. No matter if the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the uSDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC will automatically send the CMD12 to mark the end of multi-block transfer.

56.5.3.2.4 DLL (Delay Line) in Read Path

The DLL(Delay Line) is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage and temperature (PVT).

The reasons why the DLL is needed for uSDHC are 1.) the path of read data traveling from card to host varies. 2.) in SD/MMC DDR mode the minimum input setup and hold time are both at 2.5 ns. The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in override mode. The override value set is the number of delay cells. In override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL will automatically adjust the number of delay cells according to target value set by user and PVT changes. Be aware that target value is in units of 1/32 of the clock reference period. If the card_clk is 100Mhz, then the reference clock period is 10ns, setting target vaule of 16 means 5ns =(16/32)*10ns. Software can disable automatic update by setting dll_gate_update bit. Please refer to Figure 56-20.

Since the user may change the frequency of the card_clk from time to time by changing SDCLKFS[7:0]/DVS[3:0], the software must adjust the delay value to ensure it works correctly when the reference clock (card_clk) is changed. The following is the correct flow, which should be ignored if DLL_CTRL_ENABLE is not set.

Step 1: Set DLL_CTRL_RESET bit

Step 2: Configure the SDCLKFS[7:0] and DVS[3:0]

Step 3: Wait until SDSTB is asserted

Step 4: clear DLL_CTRL_RESET bit

Step 5: Wait until both DLL_STS_SLV_LOCK and DLL_STS_REF_LOCK are asserted

Step 6: set DLL_CTRL_SLV_FORCE_UPD

Step 7: clear DLL_CTRL_SLV_FORCE_UPD

NOTE:Software should make sure the DLL_CTRL_SLV_FORCE_UPD is lasted for at least one card_clk. So software may need to add some delay between step6 and step7.

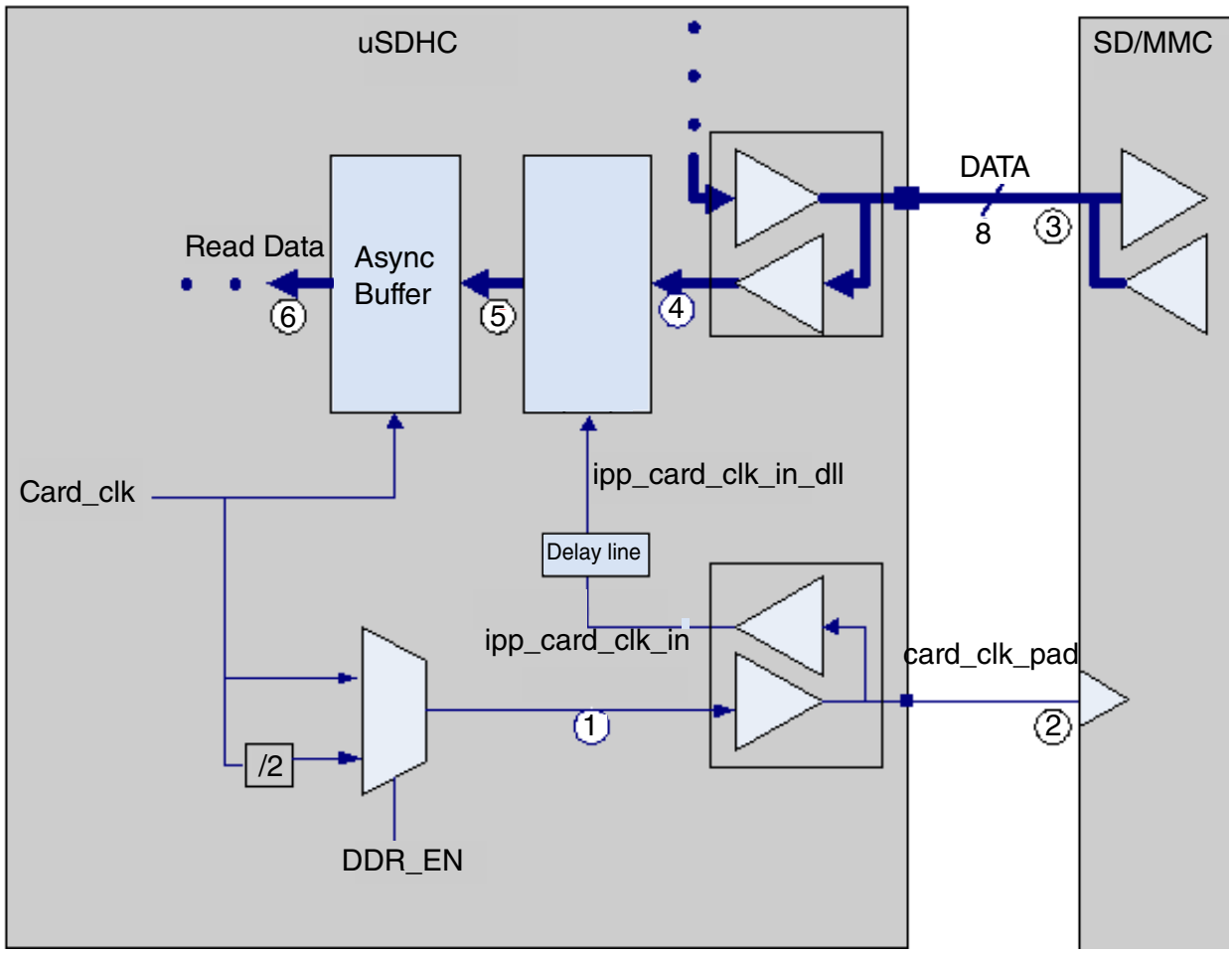


Figure 56-20. DLL(Delay Line) in Read Path

56.5.3.3 Suspend Resume

The uSDHC supports the Suspend Resume operations of SDIO cards, although slightly differently than the suggested implementation of Suspend in the SDIO card specification.

56.5.3.3.1 Suspend

After setting the SABGREQ bit, the Host Driver may send a Suspend command to switch to another function of the SDIO card. The uSDHC does not monitor the content of the response, so it doesn't know if the Suspend command succeeded or not. Accordingly, it doesn't de-assert Read Wait for read pause. To solve this problem, the Driver shall not mark the Suspend command as a "Suspend", (i.e. setting the CMDTYP bits to 01). Instead, the Driver shall send this command as if it were a normal command, and only

when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform the uSDHC that the current transfer is suspended. As shown in the following sequence for Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

56.5.3.3.2 Resume

To resume the data transfer, a Resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation above.
2. Send the Resume command. In the Transfer Type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer will be resumed.

56.5.3.4 ADMA Usage

To use the ADMA in a data transfer, the Host Driver must prepare the correct descriptor chain prior to sending the read/write command.

The steps to prepare the correct descriptor chain are:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4kB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.

4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

56.5.3.5 Transfer Error

Information about CRC, Internal DMA, Transfer ADMA, and Auto CMD12 Errors are detailed in the sections below.

56.5.3.5.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one.

For a multi-block transfer, the Host Driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, the uSDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 will be sent by the uSDHC. In this case, the Driver shall re-send or re-obtain the last block with a single block transfer.

56.5.3.5.2 Internal DMA Error

During the data transfer with internal Simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA Error interrupt is sent to the Host System. When acknowledged by such an interrupt, the Driver shall calculate the start address of data block in which the error occurs.

The start address can be calculated by either:

1. Reading the DMA System Address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the

next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.
2. Reading the BLKCNT field of the Block Attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register does not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and re-start the transfer from the corrupted block to recover from the error.

56.5.3.5.3 Transfer ADMA Error

There are 3 kinds of possible ADMA errors. The AHB transfer, invalid descriptor, and data-length mismatch errors. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set.

For acknowledging the status, the Host Driver should recover the error as shown below and re-transfer from the place of interruption.

1. AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
2. Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and re-create the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
3. Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

56.5.3.5.4 Auto CMD12 Error

After the last block of the multi block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, the uSDHC automatically sends a CMD12 to the card to stop the transfer.

When errors with this command occur, it is recommended to the Driver to deal with the situations in the following manner:

1. Auto CMD12 response time-out. It is not certain whether the command is accepted by the card or not. The Driver should clear the Auto CMD12 error status bits and re-send the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error. Since card responds to the CMD12, the card will abort the transfer. The Driver may ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The command is not sent, so the Driver shall send a CMD12 manually.

56.5.3.6 Card Interrupt

The external cards can inform the Host Controller by means of some special signals. For the SDIO card, it can be the low level on the DATA1 line during some special period. The uSDHC only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the uSDHC, and the Host System is informed by the uSDHC asserting the uSDHC interrupt line, the interrupt service from the Host Driver is called.

As the interrupt source is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared by written. Refer to Card Interrupt Handling for the card interrupt handling flow.

56.5.4 Switch Function

A switch command shall be issued by the Host Driver to enable new features added to the SD/MMC spec. SD/MMC cards can transfer data at bus widths other than 1-bit. Different speed mode are also defined. To enable these features, a switch command shall be issued by the Host Driver.

For SDIO cards, the high speed mode/DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards , the high speed mode/HS200 are queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit ,DDR4-bit and DDR8-bit width of the MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudocode examples do not show current capability check, which is recommended in the function switch process.

56.5.4.1 Query, Enable and Disable SDIO High Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

56.5.4.2 Query, Enable and Disable SD High Speed Mode/SDR50/ SDR104/DDR50

```
enable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
wait data transfer done bit is set;
check if the bit x of received 512 bits is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
send CMD6, with argument 0x80FFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104 x=3; DDR50 x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz for high speed mode, 100Mhz for SDR50, 200Mhz for SDR104, 50Mhz for
DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
```

```
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

56.5.4.3 Query, Enable and Disable MMC High Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

56.5.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate), x=5;8-
bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

56.5.5 ADMA Operation

Code for ADMA1 Operation and ADMA2 Operation can be found here.

56.5.5.1 ADMA1 Operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB aligned);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

56.5.5.2 ADMA2 Operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

56.5.6 Fast Boot Operation

56.5.6.1 Normal fast boot flow

1. Software must configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software must configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode or not. If the data will be sent through DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack timeout value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set DTW (data transfer width). If in DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept at the default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.
7. DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot timeout happens (ack times out or the first data read times out), an interrupt will be triggered, and software must configure MMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.
9. If there is no timeout, software needs to determine when the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This will make CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin normal initialization process.
10. Reset the host and then can begin the normal process.

56.5.6.2 Alternative fast boot flow

1. Software needs to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the DTW (data transfer width). If in ddr fast boot mode, DTW only can be configure to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. Note DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
7. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot timeout (ack data timeout in 50ms or data timeout in 1s), the host will send out the interrupt and software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process..
8. If there is no time out, software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the MMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. Reset the host and then begin the normal process.

56.5.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during MMC fast boot.

In fast boot, the host can use ADMA2 (Advanced DMA2) with two destinations.

The detail flow is described below:

1. Software needs to configure INIT_ACTIVE bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (BLK_CNT - VALUE1).Here VALUE1 is the value of the block count that needs to transfer the first time, so that that the hostwill stop at the block gap when the uSDHC controller gets VAULE1 blocks from the device. Also configure bits[3-0] to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (BLK_CNT) to the max value (16'hffff).
4. Software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software enbable ADMA2 by configuring Protocol Control Register bits [9-8].
6. Software need to set at least three pairs ADMA2 descriptor in boot memory (ie, in IRAM, at least 6 word). The first pair descriptor define the start address (ie, IRAM)and data length(ie,512byte*VALUE1) of fisrt part boot code. Software also need to set the second pair descriptor, the second start address (any value that is writeable), data length is suggest to set 1~2word(record as VALUE2). Note: the second couple desc also transfer useful data even at lease 1 word. Because our ADMA2 can't support 0 data_length data transfer descriptor.
7. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot, and don't need set in normal fast boot.
8. Software needs to configure Transfer Type Register to start the boot process . CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. DMAEN is configured as 1 in DMA mode. And if BCEN is configured as 1, better to configure blk no in Bock Attributes Register to the max value. And if in ddr fast boot mode, DDR_EN need to be set to 1.
9. When the step 8 is configured, boot process will begin, the first VALUE1 block number data has transfer. Software need to polling TC bit (bit1 in Interrupt Status Register) to determine first transfer is end. Also software need to polling BGE bit (bit2 in Interrupt Status Register) to determine if first transfer stop at block gap.

10. When TC, BGE bit is 1, . SW can analyzes the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of boot code(VALUE3 the remain boot code block). Remeber set the last descriptor with END.
11. Software needs to configure MMC Boot Register (offset 0xc4) again. Set bit 6 to 1(enable boot); and bit 5 to 0(normal fast boot), to 1(alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 for enable automatically stop at block gap feature. Also configure bit31-bit16 to set the *(BLK_CNT - (VALUE1+1+VALUE3))*, that host will stop at block gap when *the uSDHC controller gets (VALUE1+1+VALUE3)) blocks from device totally include the blocks received in step 9*. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequence. Please note, Software doesn't need to configure the *BLK_CNT* again, because it's counted down automatically by the uSDHC controller.
12. Software needs to clear TC and BGE bit. And software needs to clear SABGREQ(bit 16 in Protocol control register), and set CREQ(bit17 Protocol control register) to 1 to resume the data transfer. Host will transfer the VALUE2 and VALUE3 data to the destination that is set by descriptor.
13. Software need to polling BGE bit to determine if the fast boot is over.

Note:

1. When ADMA boot flow is started, for uSDHC, it is like a normal ADMA read operation. So setting ADMA2 descriptor as the normal ADMA2 transfer.
2. Need a few words length memory to keep descriptor.
3. For the 1~2 word data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

56.6 Commands for MMC/SD/SDIO

A table containing the list of commands for the MMC/SD/SDIO cards can be found here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. broadcast commands (bc), no response.
2. broadcast commands with response (bcr), response from all cards simultaneously.
3. addressed (point-to-point) commands (ac), no data transfer on the DATA.
4. addressed (point-to-point) data transfer commands (adtc).

Response: a response is a token which is sent from the card to the host as an answner to a previously received command. A response is transferred serially on the CMD line.

Table 56-3. Commands for MMC/SD/SDIO Cards

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.

Table continues on the next page...

Table 56-3. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation

Table continues on the next page...

Table 56-3. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
					after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This

Table continues on the next page...

Table 56-3. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
					command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57-59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62-63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.

Table continues on the next page...

Table 56-3. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default(one write block).
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on CD_B/DATA3 of the card.
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards . The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 2.
4. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

The Access Bits for the EXT_CSD Access Modes are shown below.

Table 56-4. EXT_CSD Access Modes

Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the 1 bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the 1 bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

56.7 Software Restrictions

56.7.1 Initialization Active

The driver cannot set INITA bit in System Control register when any of the command line or data lines is active, so the driver must ensure both CDIHB and CIHB bits are cleared.

56.7.2 Software Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block.

For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

56.7.3 Suspend Operation

In order to suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended trasnfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such 'suspend' command, uSDHC will regard the current transfer is aborted and change BLKCNT register to its original value, instead of keeping the remained number of blocks.

56.7.4 Data Length Setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

56.7.5 (A)DMA Address Setting

To configure ADMA1/ADMA2/DMA address register, when TC bit is set, the register will always update itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring ADMA1/ADMA2/DMA address register.

56.7.6 Data Port Access

Data Port does not support parallel access. For example, during an external DMA access, it is not allowed to write any data to the Data Port by CPU; or during a CPU read operation, it is also prohibited to write any data to the Data Port, by either CPU or external DMA. Otherwise the data would be corrupted inside the uSDHC buffer.

56.7.7 Change Clock Frequency

uSDHC does not automatically gate off the card clock when the Host Driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing clock divisor value(SDCLKFS or DVS in System Control Register) or setting RSTA bit.

Also before changing the clock divisor value, Host Driver should make sure the SDSTB bit is high.

56.7.8 Multi-block Read

For pre-defined multi-block read operation, i.e., the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode.

In this case, the card may not respond to this extra abort command and uSDHC will get Response Timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

56.8 uSDHC Memory Map/Register Definition

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map for the uSDHC. All these registers only support 32-bit accesses.

NOTE

The uSDHC registers are 32-bit wide and only support 32-bit access.

uSDHC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_0000	DMA System Address (uSDHC1_DS_ADDR)	32	R/W	0000_0000h	56.8.1/3568
219_0004	Block Attributes (uSDHC1_BLK_ATT)	32	R/W	0000_0000h	56.8.2/3569
219_0008	Command Argument (uSDHC1_CMD_ARG)	32	R/W	0000_0000h	56.8.3/3570
219_000C	Command Transfer Type (uSDHC1_CMD_XFR_TYP)	32	R/W	0000_0000h	56.8.4/3571
219_0010	Command Response0 (uSDHC1_CMD_RSP0)	32	R	0000_0000h	56.8.5/3574
219_0014	Command Response1 (uSDHC1_CMD_RSP1)	32	R	0000_0000h	56.8.6/3575
219_0018	Command Response2 (uSDHC1_CMD_RSP2)	32	R	0000_0000h	56.8.7/3575
219_001C	Command Response3 (uSDHC1_CMD_RSP3)	32	R	0000_0000h	56.8.8/3576
219_0020	Data Buffer Access Port (uSDHC1_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	56.8.9/3577
219_0024	Present State (uSDHC1_PRES_STATE)	32	R	0000_8080h	56.8.10/ 3578
219_0028	Protocol Control (uSDHC1_PROT_CTRL)	32	R/W	0880_0020h	56.8.11/ 3583
219_002C	System Control (uSDHC1_SYS_CTRL)	32	R/W	8080_800Fh	56.8.12/ 3588
219_0030	Interrupt Status (uSDHC1_INT_STATUS)	32	w1c	0000_0000h	56.8.13/ 3591
219_0034	Interrupt Status Enable (uSDHC1_INT_STATUS_EN)	32	R/W	0000_0000h	56.8.14/ 3597
219_0038	Interrupt Signal Enable (uSDHC1_INT_SIGNAL_EN)	32	R/W	0000_0000h	56.8.15/ 3600
219_003C	Auto CMD12 Error Status (uSDHC1_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	56.8.16/ 3603
219_0040	Host Controller Capabilities (uSDHC1_HOST_CTRL_CAP)	32	R	07F3_B407h	56.8.17/ 3606
219_0044	Watermark Level (uSDHC1_WTMK_LVL)	32	R/W	0810_0810h	56.8.18/ 3609
219_0048	Mixer Control (uSDHC1_MIX_CTRL)	32	R/W	8000_0000h	56.8.19/ 3610

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_0050	Force Event (uSDHC1_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	56.8.20/ 3612
219_0054	ADMA Error Status Register (uSDHC1_ADMA_ERR_STATUS)	32	R	0000_0000h	56.8.21/ 3615
219_0058	ADMA System Address (uSDHC1_ADMA_SYS_ADDR)	32	R/W	0000_0000h	56.8.22/ 3617
219_0060	DLL (Delay Line) Control (uSDHC1_DLL_CTRL)	32	R/W	0000_0200h	56.8.23/ 3618
219_0064	DLL Status (uSDHC1_DLL_STATUS)	32	R	0000_0000h	56.8.24/ 3620
219_0068	CLK Tuning Control and Status (uSDHC1_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	56.8.25/ 3621
219_00C0	Vendor Specific Register (uSDHC1_VEND_SPEC)	32	R/W	2000_7809h	56.8.26/ 3623
219_00C4	MMC Boot Register (uSDHC1_MMC_BOOT)	32	R/W	0000_0000h	56.8.27/ 3626
219_00C8	Vendor Specific 2 Register (uSDHC1_VEND_SPEC2)	32	R/W	0000_0006h	56.8.28/ 3627
219_00CC	Tuning Control Register (uSDHC1_TUNING_CTRL)	32	R/W	0021_2800h	56.8.29/ 3629
219_4000	DMA System Address (uSDHC2_DS_ADDR)	32	R/W	0000_0000h	56.8.1/3568
219_4004	Block Attributes (uSDHC2_BLK_ATT)	32	R/W	0000_0000h	56.8.2/3569
219_4008	Command Argument (uSDHC2_CMD_ARG)	32	R/W	0000_0000h	56.8.3/3570
219_400C	Command Transfer Type (uSDHC2_CMD_XFR_TYP)	32	R/W	0000_0000h	56.8.4/3571
219_4010	Command Response0 (uSDHC2_CMD_RSP0)	32	R	0000_0000h	56.8.5/3574
219_4014	Command Response1 (uSDHC2_CMD_RSP1)	32	R	0000_0000h	56.8.6/3575
219_4018	Command Response2 (uSDHC2_CMD_RSP2)	32	R	0000_0000h	56.8.7/3575
219_401C	Command Response3 (uSDHC2_CMD_RSP3)	32	R	0000_0000h	56.8.8/3576
219_4020	Data Buffer Access Port (uSDHC2_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	56.8.9/3577
219_4024	Present State (uSDHC2_PRES_STATE)	32	R	0000_8080h	56.8.10/ 3578
219_4028	Protocol Control (uSDHC2_PROT_CTRL)	32	R/W	0880_0020h	56.8.11/ 3583
219_402C	System Control (uSDHC2_SYS_CTRL)	32	R/W	8080_800Fh	56.8.12/ 3588
219_4030	Interrupt Status (uSDHC2_INT_STATUS)	32	w1c	0000_0000h	56.8.13/ 3591
219_4034	Interrupt Status Enable (uSDHC2_INT_STATUS_EN)	32	R/W	0000_0000h	56.8.14/ 3597
219_4038	Interrupt Signal Enable (uSDHC2_INT_SIGNAL_EN)	32	R/W	0000_0000h	56.8.15/ 3600

Table continues on the next page...

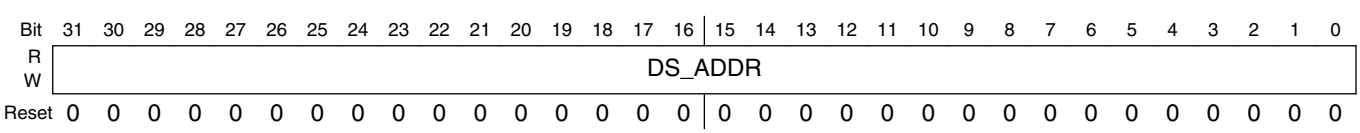
uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_403C	Auto CMD12 Error Status (uSDHC2_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	56.8.16/ 3603
219_4040	Host Controller Capabilities (uSDHC2_HOST_CTRL_CAP)	32	R	07F3_B407h	56.8.17/ 3606
219_4044	Watermark Level (uSDHC2_WTMK_LVL)	32	R/W	0810_0810h	56.8.18/ 3609
219_4048	Mixer Control (uSDHC2_MIX_CTRL)	32	R/W	8000_0000h	56.8.19/ 3610
219_4050	Force Event (uSDHC2_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	56.8.20/ 3612
219_4054	ADMA Error Status Register (uSDHC2_ADMA_ERR_STATUS)	32	R	0000_0000h	56.8.21/ 3615
219_4058	ADMA System Address (uSDHC2_ADMA_SYS_ADDR)	32	R/W	0000_0000h	56.8.22/ 3617
219_4060	DLL (Delay Line) Control (uSDHC2_DLL_CTRL)	32	R/W	0000_0200h	56.8.23/ 3618
219_4064	DLL Status (uSDHC2_DLL_STATUS)	32	R	0000_0000h	56.8.24/ 3620
219_4068	CLK Tuning Control and Status (uSDHC2_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	56.8.25/ 3621
219_40C0	Vendor Specific Register (uSDHC2_VEND_SPEC)	32	R/W	2000_7809h	56.8.26/ 3623
219_40C4	MMC Boot Register (uSDHC2_MMC_BOOT)	32	R/W	0000_0000h	56.8.27/ 3626
219_40C8	Vendor Specific 2 Register (uSDHC2_VEND_SPEC2)	32	R/W	0000_0006h	56.8.28/ 3627
219_40CC	Tuning Control Register (uSDHC2_TUNING_CTRL)	32	R/W	0021_2800h	56.8.29/ 3629

56.8.1 DMA System Address (uSDHCx_DS_ADDR)

This register contains the physical system memory address used for DMA transfers.

Address: Base address + 0h offset



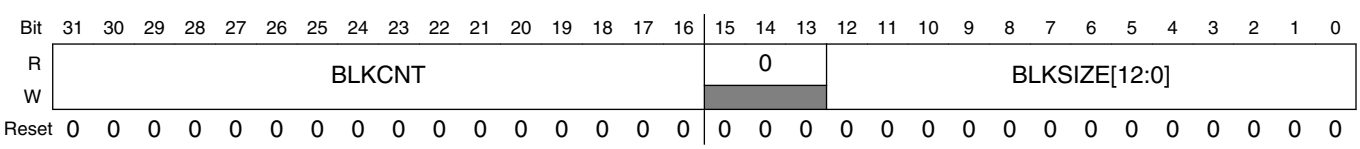
uSDHCx_DS_ADDR field descriptions

Field	Description
DS_ADDR	**DMA System Address / Argument 2** When ACMD23_ARGU2_EN is set to 0, SDMA uses this register as system address and supports only 32-bit addressing mode. Auto CMD23 cannot be used with SDMA. When ACMD23_ARGU2_EN is set to 1, SDMA uses ADMA System Address register (05Fh – 058h) instead of this register to support both 32-bit and 64-bit addressing. This register is used only for Argument2 and SDMA may use Auto CMD23. 1. **SDMA System Address** Since the address must be word (4 bytes) aligned, the least 2 bits are reserved, always 0. When the uSDHC stops a DMA transfer, this register points out the system address of the next contiguous data position. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operation during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The Host driver shall wait, until the DLA bit in the Present State register is cleared, before writing to this register. The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC will automatically change SEQ burst type to NSEQ. Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions. 2. **Argument 2** This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23. If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case.

56.8.2 Block Attributes (uSDHCx_BLK_ATT)

This register is used to configure the number of data blocks and the number of bytes in each block.

Address: Base address + 4h offset



uSDHCx_BLK_ATT field descriptions

Field	Description
31–16 BLKCNT	Blocks Count For Current Transfer:

Table continues on the next page...

uSDHCx_BLK_ATT field descriptions (continued)

Field	Description
	This register is enabled when the Block Count Enable bit in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The Host Driver shall set this register to a value between 1 and the maximum block count. The uSDHC decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred. This register should be accessed only when no transaction is executing (i.e. after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored. When saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when Suspend command is sent out, uSDHC will regard the current transfer is aborted and change BLKCNT register back to its original value instead of keeping the dynamical indicator of remained block count. When restoring transfer content prior to issuing a Resume command, the Host Driver shall restore the previously saved block count. **NOTE:** Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL bit is indicating a single block transfer, the read value of BLKCNT is always 1. FFFF 65535 blocks 0002 2 blocks 0001 1 block 0000 Stop Count
15–13 Reserved	This read-only field is reserved and always has the value 0.
BLKSIZE[12:0]	Transfer Block Size: This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations will be ignored. 1000 4096 Bytes 800 2048 Bytes 200 512 Bytes 1FF 511 Bytes 004 4 Bytes 003 3 Bytes 002 2 Bytes 001 1 Byte 000 No data transfer

56.8.3 Command Argument (uSDHCx_CMD_ARG)

This register contains the SD / MMC Command Argument.

.

Address: Base address + 8h offset

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

uSDHCx_CMD_ARG field descriptions

Field	Description
CMDARG[31:0]	Command Argument The SD / MMC Command Argument is specified as bits 39-8 of the Command Format in the SD or MMC Specification.This register is write protected when the Command Inhibit (CMD) bit in the Present State register is set.

56.8.4 Command Transfer Type (uSDHCx_CMD_XFR_TYP)

This register is used to control the operation of data transfers. The Host Driver shall set this register before issuing a command followed by a data transfer, or before issuing a Resume command. To prevent data loss, the uSDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The Host Driver shall check the Command Inhibit DAT bit (CDIHB) and the Command Inhibit CMD bit (CIHB) in the Present State register before writing to this register. When the CDIHB bit in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

On sending commands with data transfer invovled, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is '1'), otherwise uSDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, i.e., response time-out, uSDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver should issue the command again to re-try the transfer. It is also possible that for some reason the card responds the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 56-5. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

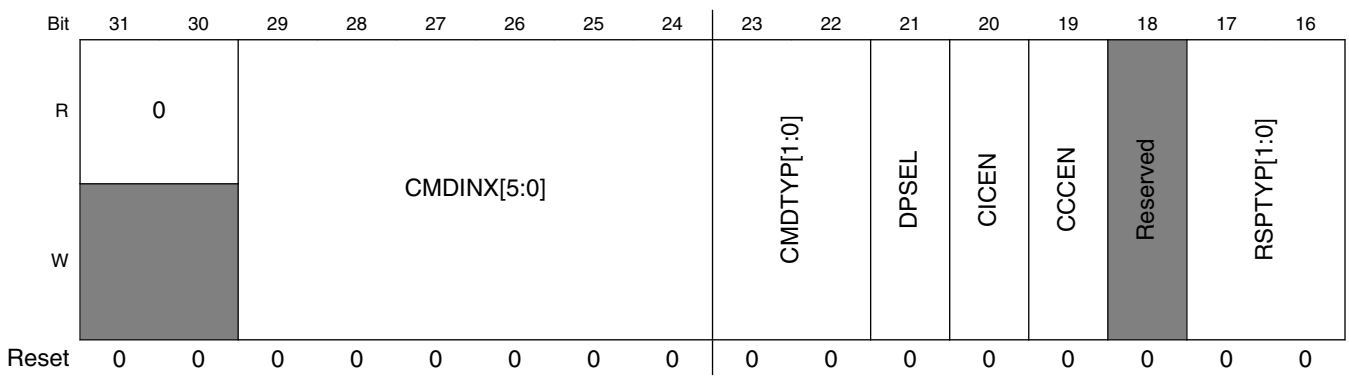
The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, in regards to the Response Type bits as well as the name of the response type.

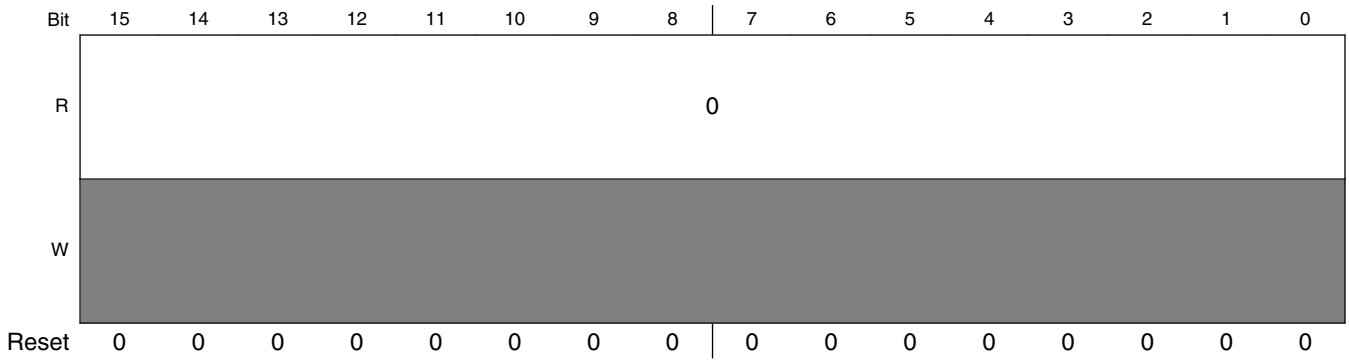
Table 56-6. Relationship Between Parameters and the Name of the Response Type

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3,R4
10	1	1	R1,R5,R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to specify that the uSDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command shall be used with R5b.
- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

Address: Base address + Ch offset





uSDHCx_CMD_XFR_TYP field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.
29–24 CMDINX[5:0]	Command Index These bits shall be set to the command number that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23–22 CMDTYP[1:0]	Command Type There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00b for all other commands. • Suspend Command: If the Suspend command succeeds, the uSDHC shall assume that the card bus has been released and that it is possible to issue the next command which uses the DATA line. Since the uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the Host Driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform the uSDHC that a Suspend command was successfully issued. Refer to Suspend Resume for more details. After the end bit of command is sent, the uSDHC de-asserts Read Wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the uSDHC will maintain its current state, and the Host Driver shall restart the transfer by setting the Continue Request bit in the Protocol Control register. • Resume Command: The Host Driver re-starts the data transfer by restoring the registers saved before sending the Suspend Command and then sends the Resume Command. The uSDHC will check for a pending busy state before starting write transfers. • Abort Command: If this command is set when executing a read transfer, the uSDHC will stop reads to the buffer. If this command is set when executing a write transfer, the uSDHC will stop driving the DATA line. After issuing the Abort command, the Host Driver should issue a software reset (Abort Transaction). 11 Abort CMD12, CMD52 for writing I/O Abort in CCCR 10 Resume CMD52 for writing Function Select in CCCR 01 Suspend CMD52 for writing Bus Suspend in CCCR 00 Normal Other commands
21 DPSEL	Data Present Select This bit is set to 1 to indicate that data is present and shall be transferred using the DATA line. It is set to 0 for the following: • Commands using only the CMD line (e.g. CMD52). • Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b e.g. CMD38)

Table continues on the next page...

uSDHCx_CMD_XFR_TYP field descriptions (continued)

Field	Description
	NOTE: In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, (i.e. the WPSPL bit is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored. 1 Data Present 0 No Data Present
20 CICEN	Command Index Check Enable If this bit is set to 1, the uSDHC will check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked. 1 Enable 0 Disable
19 CCCEN	Command CRC Check Enable If this bit is set to 1, the uSDHC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and Command Transfer Type (uSDHC_CMD_XFR_TYP) .) 1 Enable 0 Disable
18 -	This field is reserved. Reserved
17–16 RSPTYP[1:0]	Response Type Select 00 No Response 01 Response Length 136 10 Response Length 48 11 Response Length 48, check Busy after response
Reserved	This read-only field is reserved and always has the value 0.

56.8.5 Command Response0 (uSDHCx_CMD_RSP0)

This register is used to store part 0 of the response bits from the card.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP0[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

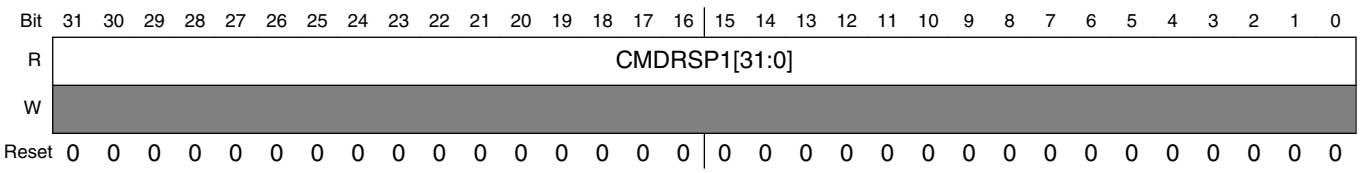
uSDHCx_CMD_RSP0 field descriptions

Field	Description
CMDRSP0[31:0]	Command Response 0 Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

56.8.6 Command Response1 (uSDHCx_CMD_RSP1)

This register is used to store part 1 of the response bits from the card.

Address: Base address + 14h offset



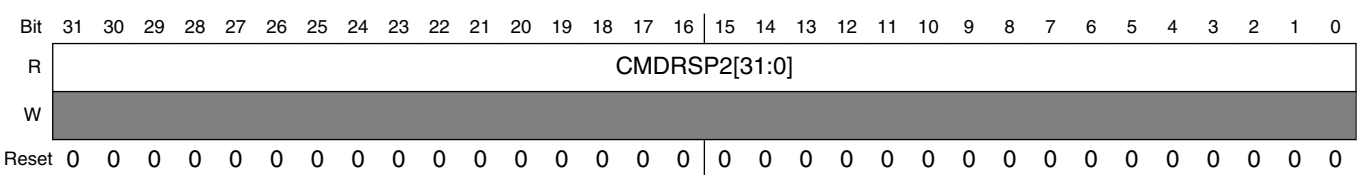
uSDHCx_CMD_RSP1 field descriptions

Field	Description
CMDRSP1[31:0]	Command Response 1 Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

56.8.7 Command Response2 (uSDHCx_CMD_RSP2)

This register is used to store part 2 of the response bits from the card.

Address: Base address + 18h offset



uSDHCx_CMD_RSP2 field descriptions

Field	Description
CMDRSP2[31:0]	Command Response 2

uSDHCx_CMD_RSP2 field descriptions (continued)

Field	Description
	Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

56.8.8 Command Response3 (uSDHCx_CMD_RSP3)

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD Bus to Command Response registers for each response type. In the table, R[ ] refers to a bit range within the response data as transmitted on the SD Bus.

Table 56-7. Response Bit Definition for Each Response Type

Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card Status	R[39:8]	CMDRSP0
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the uSDHC only stores part of the response data in the Command Response registers. This enables the Host Driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by the uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit

range for the CRC check depends on the response length. If the response length is 48, the uSDHC will check R[47:1], and if the response length is 136 the uSDHC will check R[119:1].

Since the uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows the uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP3[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_RSP3 field descriptions

Field	Description
CMDRSP3[31:0]	Command Response 3 Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

56.8.9 Data Buffer Access Port (uSDHCx_DATA_BUFF_ACC_PORT)

This is a 32-bit data port register used to access the internal buffer.

Address: Base address + 20h offset

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

uSDHCx_DATA_BUFF_ACC_PORT field descriptions

Field	Description
DATCONT[31:0]	Data Content The Buffer Data Port register is for 32-bit data access by the ARM platform or the external DMA. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s.

56.8.10 Present State (uSDHCx_PRES_STATE)

The Host Driver can get status of the uSDHC from this 32-bit read only register.

- The Host Driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands shall be issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.
- Note: the reset value of Present State Register depend on testbench connectivity.

Address: Base address + 24h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLSL[7:0]								CLSL	0			WPSPL	CDPL	0	CINST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TSCD	0		RTR	BREN	BWEN	RTA	WTA	SDOFF	PEROFF	HCKOFF	IPGOFF	SDSTB	DLA	CDIHB	CIHB
W																
Reset	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

uSDHCx_PRES_STATE field descriptions

Field	Description
31–24 DLSL[7:0]	DATA[7:0] Line Signal Level This status is used to check the DATA line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this bit field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up. DATA7 Data 7 line signal level DATA6 Data 6 line signal level DATA5 Data 5 line signal level DATA4 Data 4 line signal level DATA3 Data 3 line signal level DATA2 Data 2 line signal level DATA1 Data 1 line signal level DATA0 Data 0 line signal level
23 CLSL	CMD Line Signal Level This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this bit after reset is 1'b1, when the command line is pulled up.
22–20 Reserved	This read-only field is reserved and always has the value 0.
19 WPSPL	Write Protect Switch Pin Level The Write Protect Switch is supported for memory and combo cards.This bit reflects the inverted value of the WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled. 1 Write enabled (WP = 0) 0 Write protected (WP = 1)
18 CDPL	Card Detect Pin Level This bit reflects the inverse value of the CD_B pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed, because of propagation delay. Use of this bit is limited to testing since it must be debounced by software. A software reset does not effect this bit. A write to the Force Event Register does not effect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the CD_B pin (i.e. when a card is inserted in the socket, it is 0 on the CD_B input, and consequently the CDPL reads 1.) 1 Card present (CD_B = 0) 0 No card present (CD_B = 1)
17 Reserved	This read-only field is reserved and always has the value 0.
16 CINST	Card Inserted This bit indicates whether a card has been inserted. The uSDHC debounces this signal so that the Host Driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not effect this bit. The Software Reset For All in the System Control register does not effect this bit. A software reset does not effect this bit.

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	1 Card Inserted 0 Power on Reset or No Card
15 TSCD	Tape Select Change Done This bit indicates the dealy setting is effective after write CLK_TUNE_CTRL_STATUS register. 1 Delay cell select change is finished. 0 Delay cell select change is not finished.
14–13 Reserved	This read-only field is reserved and always has the value 0.
12 RTR	Re-Tuning Request (only for SD3.0 SDR104 mode) Host Controller may request Host Driver to execute re-tuning sequence by setting this bit when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This bit is cleared when a command is issued with setting Execute Tuning bit in MIXER_CTRL register. Changing of this bit from 0 to 1 generates Re-Tuning Event. Refer to Interrupt status registers for more detail. This bit isn't set to 1 if Sampling Clock Select in the MIXER_CTRL register is set to 0 (using fixed sampling clock). 1 Sampling clock needs re-tuning 0 Fixed or well tuned sampling clock
11 BREN	Buffer Read Enable This status bit is used for non-DMA read transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. A change of this bit from 1 to 0 occurs when some reads from the buffer(read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this bit from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 1 Read enable 0 Read disable
10 BWEN	Buffer Write Enable This status bit is used for non-DMA write transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. A change of this bit from 1 to 0 occurs when some writes to the buffer(write DATPORT(Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this bit from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 1 Write enable 0 Write disable
9 RTA	Read Transfer Active This status bit is used for detecting completion of a read transfer. This bit is set for either of the following conditions:

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	• After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. A Transfer Complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, i.e. all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1. 1 Transferring data 0 No valid data
8 WTA	Write Transfer Active This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the uSDHC. This bit is set in either of the following cases: • After the end bit of the write command. • When writing 1 to the Continue Request bit in the Protocol Control register to restart a write transfer. This bit is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple). • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during Write Busy state. 1 Transferring data 0 No valid data
7 SDOFF	SD Clock Gated Off Internally This status bit indicates that the SD Clock is internally gated off, because of buffer over / under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON bit is 0 to stop the SD clock in idle status. Set IPG_PERCLK_SOFT_EN and CARD_CLK_SOFT_EN to 0 also gate off SD clock. This bit is for the Host Driver to debug data transaction on the SD bus. 1 SD Clock is gated off. 0 SD Clock is active.
6 PEROFF	IPG_PERCLK Gated Off Internally This status bit indicates that the IPG_PERCLK is internally gated off. This bit is for the Host Driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, IPG_PERCLK will be gated off, otherwise IPG_PERCLK will be always active. 1 IPG_PERCLK is gated off. 0 IPG_PERCLK is active.
5 HCKOFF	HCLK Gated Off Internally

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	This status bit indicates that the HCLK is internally gated off. This bit is for the Host Driver to debug during a data transfer. 1 HCLK is gated off. 0 HCLK is active.
4 IPGOFF	IPG_CLK Gated Off Internally This status bit indicates that the ipg_clk is internally gated off. This bit is for the Host Driver to debug. 1 IPG_CLK is gated off. 0 IPG_CLK is active.
3 SDSTB	SD Clock Stable This status bit indicates that the internal card clock is stable. This bit is for the Host Driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON bit in System Control register to remove glitches on the card clock when the frequency is changing. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* 1 Clock is stable. 0 Clock is changing frequency and not stable.
2 DLA	Data Line Active This status bit indicates whether one of the DATA lines on the SD Bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. This bit will be cleared in either of the following cases: (1) When the end bit of the last data block is sent from the SD Bus to the uSDHC. (2) When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC will wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function. This bit will remain 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the write command. • When writing to 1 to the Continue Request bit in the Protocol Control register to continue a write transfer. This bit will be cleared in either of the following cases:

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	• When the SD card releases Write Busy of the last data block, the uSDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the uSDHC shall assume the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and as a result of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DATA0 line is released. 1 DATA Line Active 0 DATA Line Inactive
1 CDIHB	Command Inhibit (DATA) This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the uSDHC can issue the next SD / MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. **NOTE:** The SD Host Driver can save registers for a suspend transaction after this bit has changed from 1 to 0. 1 Cannot issue command which uses the DATA line 0 Can issue command which uses the DATA line
0 CIHB	Command Inhibit (CMD) If this status bit is 0, it indicates that the CMD line is not in use and the uSDHC can issue a SD / MMC Command using the CMD line. This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If the uSDHC cannot issue the command because of a command conflict error (Refer to Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete is not set. The Status of issuing an Auto CMD12 does not show on this bit. 1 Cannot issue command 0 Can issue command using only CMD line

56.8.11 Protocol Control (uSDHCx_PROT_CTRL)

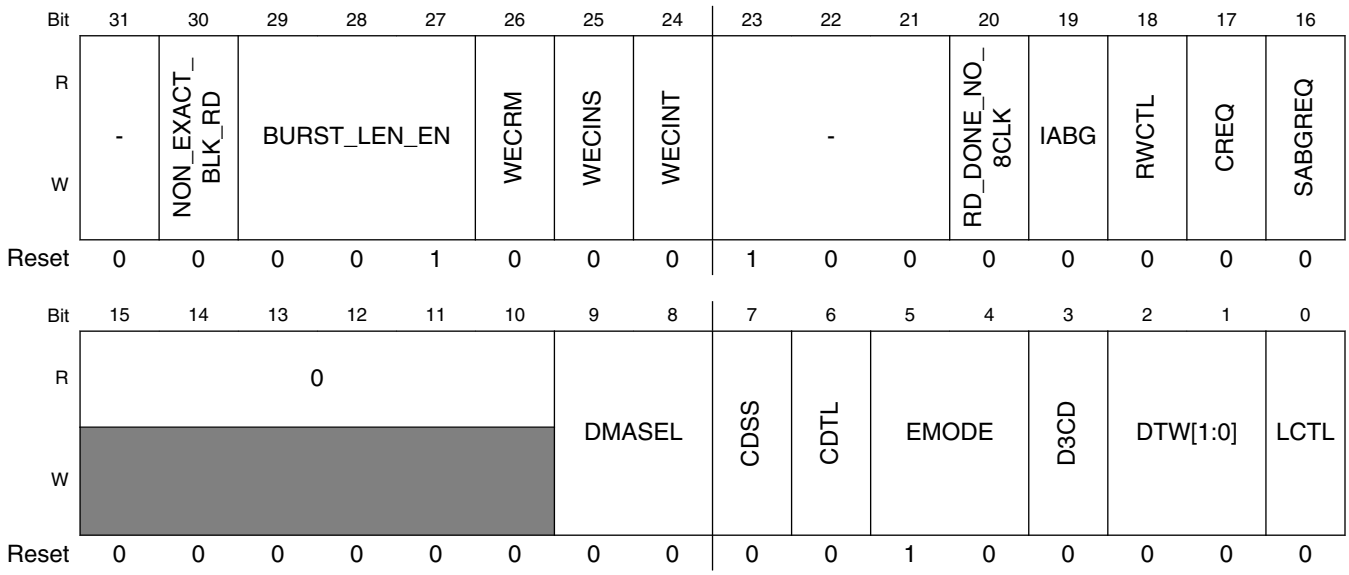
There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the uSDHC issues a Suspend command or the SD card accepts the Suspend command.

1. If the Host Driver does not issue a Suspend command, the Continue Request shall be used to restart the transfer.
2. If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command shall be used to restart the transfer.

3. If the Host Driver issues a Suspend command and the SD card does not accept it, the Continue Request shall be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the Host Driver shall wait for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the Host Driver shall clear the Stop At Block Gap Request before or simultaneously.

Address: Base address + 28h offset



uSDHCx_PROT_CTRL field descriptions

Field	Description
31 -	Reserved. Always write as 0
30 NON_EXACT_BLK_RD	Current block read is non-exact block read. It is only used for SDIO. 1 The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read. 0 The block read is exact block read. Host driver doesn't need to issue abort command to terminate this multi-block read.
29–27 BURST_LEN_EN	BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP This is used to enable / disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLEs on the AXI side. xx1 Burst length is enabled for INCR x1x Burst length is enabled for INCR4 / INCR8 / INCR16 1xx Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP
26 WECRM	Wakeup Event Enable On SD Card Removal

Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	This bit enables a wakeup event, via a Card Removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Removal Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Removal Status and the uSDHC interrupt. 1 Enable 0 Disable
25 WECINS	Wakeup Event Enable On SD Card Insertion This bit enables a wakeup event, via a Card Insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Insertion Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Insertion Status and the uSDHC interrupt. 1 Enable 0 Disable
24 WECINT	Wakeup Event Enable On Card Interrupt This bit enables a wakeup event, via a Card Interrupt, in the Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the Card Interrupt Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Interrupt Status and the uSDHC interrupt. 1 Enable 0 Disable
23–21 -	Reserved. Always write as 3'b100
20 RD_DONE_NO_8CLK	*Read done no 8 clock:* *According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end bit of the last data block. So, by default(RD_DONE_NO_8CLK=0), 8 clocks will be active after the end bit of the last read data transaction.* *However, this 8 clocks should not be active if user wants to use stop at block gap(include the auto stop at block gap in boot mode) feature for read and the RWCTL bit(bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid this 8 clocks. Otherwise, the device may send extra data to uSDHC while uSDHC ignores these data.* *In a summary, this bit should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.*
19 IABG	Interrupt At Block Gap This bit is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0 to avoid an inadvertent interrupt. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. 1 Enabled 0 Disabled
18 RWCTL	Read Wait Control The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise the uSDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SDIO

Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	card insertion, it shall set this bit according to the CCCR of the card. If the card does not support read wait, this bit shall never be set to 1, otherwise DATA line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the uSDHC will stop the SD Clock to pause reading operation. 1 Enable Read Wait Control, and assert Read Wait without stopping SD Clock at block gap when SABGREQ bit is set 0 Disable Read Wait Control, and stop SD Clock at block gap when SABGREQ bit is set
17 CREQ	Continue Request This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. When a Suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer. The uSDHC automatically clears this bit, therefore it is not necessary for the Host Driver to set this bit to 0. If both Stop At Block Gap Request and this bit are 1, the continue request is ignored. 1 Restart 0 No effect
16 SABGREQ	Stop At Block Gap Request This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the Transfer Complete is set to 1, indicating a transfer completion, the Host Driver shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC will honor the Stop At Block Gap Request for write transfers, but for read transfers it requires that the SDIO card support Read Wait. Therefore, the Host Driver shall not set this bit during read transfers unless the SDIO card supports Read Wait and has set the Read Wait Control to 1, otherwise the uSDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the Host Driver writes data to the Data Port register, the Host Driver shall set this bit after all block data is written. If this bit is set to 1, the Host Driver shall not write data to the Data Port register after a block is sent. Once this bit is set, the Host Driver shall not clear this bit before the Transfer Complete bit in Interrupt Status Register is set, otherwise the uSDHCs behavior is undefined. This bit effects Read Transfer Active, Write Transfer Active, DATA Line Active and Command Inhibit (DATA) in the Present State register. 1 Stop 0 Transfer
15–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 DMASEL	DMA Select This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00 No DMA or Simple DMA is selected 01 ADMA1 is selected 10 ADMA2 is selected 11 reserved
7 CDSS	Card Detect Signal Selection This bit selects the source for the card detection.

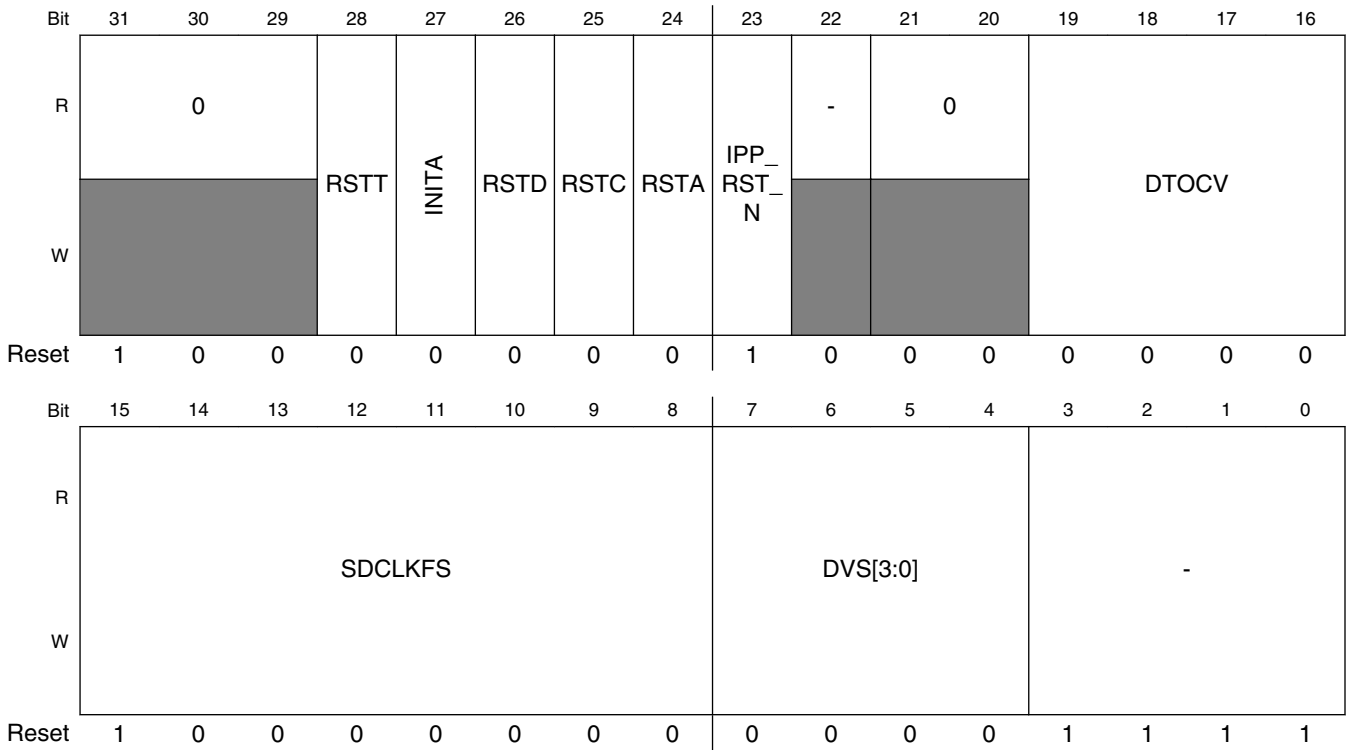
Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	1 Card Detection Test Level is selected (for test purpose). 0 Card Detection Level is selected (for normal purpose).
6 CDTL	Card Detect Test Level This is bit is enabled while the Card Detection Signal Selection is set to 1 and it indicates card insertion. 1 Card Detect Test Level is 1, card inserted 0 Card Detect Test Level is 0, no card inserted
5–4 EMODE	Endian Mode The uSDHC supports all three endian modes in data transfer. Refer to Data Buffer for more details. 00 Big Endian Mode 01 Half Word Big Endian Mode 10 Little Endian Mode 11 Reserved
3 D3CD	DATA3 as Card Detection Pin If this bit is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 may set the card into the SPI mode, which the uSDHC does not support. 1 DATA3 as Card Detection Pin 0 DATA3 does not monitor Card Insertion
2–1 DTW[1:0]	Data Transfer Width This bit selects the data width of the SD bus for a data transfer. The Host Driver shall set it to match the data width of the card. Possible Data transfer Width is 1-bit, 4-bits or 8-bits. 10 8-bit mode 01 4-bit mode 00 1-bit mode 11 Reserved
0 LCTL	LED Control This bit, fully controlled by the Host Driver, is used to caution the user not to remove the card while the card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. When the software issues multiple SD commands, setting the bit once before the first command is sufficient: it is not necessary to reset the bit between commands. 1 LED on 0 LED off

56.8.12 System Control (uSDHCx_SYS_CTRL)

Address: Base address + 2Ch offset



uSDHCx_SYS_CTRL field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 RSTT	Reset Tuning When set this bit to 1, it will reset tuning circuit. After tuning circuits are reset, bit value is 0. Clearing execute_tuning bit in AUTOCMD12_ERR_STATUS will also set this bit to 1 to reset tuning circuit
27 INITA	Initialization Active When this bit is set, 80 SD-Clocks are sent to the card. After the 80 clocks are sent, this bit is self cleared. This bit is very useful during the card power-up period when 74 SD-Clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the Present State Register are set, writing 1 to this bit is ignored (i.e. when command line or data lines are active, write to this bit is not allowed). On the otherhand, when this bit is set, i.e., during intialization active period, it is allowed to issue command, and the command bit stream will appear on the CMD pad after all 80 clock cycles are done. So when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs send 80 cycles to the card and does not want to wait till this bit is self cleared.
26 RSTD	Software Reset For DATA Line

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
	Only part of the data circuit is reset. DMA circuit is also reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Data Port register • Buffer is cleared and initialized. • Present State register • Buffer Read Enable • Buffer Write Enable • Read Transfer Active • Write Transfer Active • DATA Line Active • Command Inhibit (DATA) Protocol Control register • Continue Request • Stop At Block Gap Request Interrupt Status register • Buffer Read Ready • Buffer Write Ready • DMA Interrupt • Block Gap Event • Transfer Complete **NOTE:** When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC uSDHC_INT_STATUS register is set. 1 Reset 0 No Reset
25 RSTC	Software Reset For CMD Line Only part of the command circuit is reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Present State register Command Inhibit (CMD) • Interrupt Status register Command Complete 1 Reset 0 No Reset
24 RSTA	Software Reset For ALL This reset effects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the Host Driver shall set this bit to 1 to reset the uSDHC. The uSDHC shall reset this bit to 0 when the capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All does not affect the value of the Capabilities registers. After this bit is set, it is recommended that the Host Driver reset the external card and re-initialize it. After this bit is set, SW should wait for self-clear. In tuning process, after every CMD19 is finished, this bit will be set to retest the uSDHC. **NOTE:** When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC uSDHC_INT_STATUS register is set. 1 Reset 0 No Reset
23 IPP_RST_N	This register's value will be output to CARD from pad directly for hardware reset of the card if the card supports this feature.
22 -	Reserved

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
21–20 Reserved	This read-only field is reserved and always has the value 0.
19–16 DTOCV	Data Timeout Counter Value This value determines the interval by which DAT line timeouts are detected. Refer to the Data Timeout Error bit in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value. The Host Driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 1111 SDCLK x 2^{28} 1110 SDCLK x 2^{27} 0001 SDCLK x 2^{14} 0000 SDCLK x 2^{13}
15–8 SDCLKFS	SDCLK Frequency Select This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register. *In Single Data Rate mode(DDR_EN bit of MIXERCTRL is '0')* Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4 01h) Base clock divided by 2 00h) Base clock divided by 1 *While in Dual Data Rate mode(DDR_EN bit of MIXERCTRL is '1')* *Only the following settings are allowed:* *80h) Base clock divided by 512* *40h) Base clock divided by 256* *20h) Base clock divided by 128* *10h) Base clock divided by 64* *08h) Base clock divided by 32* *04h) Base clock divided by 16* *02h) Base clock divided by 8* *01h) Base clock divided by 4* *00h) Base clock divided by 2* *When S/W changes the DDR_EN bit, SDCLKFS may need to be changed also!*

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
	In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD Clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this bit field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD Clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD Clock frequency is 50 MHz and shall never exceed this limit. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* *If setting SDCLKFS and DVS can generate same clock frequency,(For example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.) SDCLKFS is highly recommended.*
7–4 DVS[3:0]	Divisor This register is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* The setting are as following: 0000 Divide-by-1 0001 Divide-by-2 ..... 1110 Divide-by-15 1111 Divide-by-16
-	Reserved. Always write as 1.

56.8.13 Interrupt Status (uSDHCx_INT_STATUS)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For Card Interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the Card Driver services the interrupt condition, otherwise the CINT bit will be asserted again.

The table below shows the relationship between the Command Timeout Error and the Command Complete.

Table 56-8. uSDHC Status for Command Timeout Error/Command Complete Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the Transfer Complete and the Data Timeout Error.

Table 56-9. uSDHC Status for Data Timeout Error/Transfer Complete Bit Combinations

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data Transfer Complete

The table below shows the relationship between the Command CRC Error and Command Timeout Error.

Table 56-10. uSDHC Status for Command CRC Error/Command Timeout Error Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	No error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Address: Base address + 30h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE	0	TNE	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				w1c		w1c		w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TP	0	RTE	0			CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W		w1c		w1c				w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_INT_STATUS field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DMAE	DMA Error Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Since any error corrupts the whole data block, the Host Driver shall re-start the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 1 Error 0 No Error
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNE	Tuning Error: (only for SD3.0 SDR104 mode) This bit is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, Host Driver needs to abort a command executing and perform tuning.
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12E	Auto CMD12 Error Occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1, not only when the errors in Auto CMD12 occur, but also when the Auto CMD12 is not executed due to the previous command error. 1 Error 0 No Error
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBE	Data End Bit Error Occurs either when detecting 0 at the end bit position of read data, which uses the DATA line, or at the end bit position of the CRC. This bit will be not asserted in tuning process. 1 Error 0 No Error

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
21 DCE	Data CRC Error Occurs when detecting a CRC error when transferring read data, which uses the DATA line, or when detecting the Write CRC status having a value other than 010. This bit will be not asserted in tuning process. 1 Error 0 No Error
20 DTOE	Data Timeout Error Occurs when detecting one of following time-out conditions. • Busy time-out for R1b, R5b type • Busy time-out after Write CRC status • Read Data time-out. This bit will be not asserted in tuning process. 1 Time out 0 No Error
19 CIE	Command Index Error Occurs if a Command Index error occurs in the command response. This bit will be not asserted in tuning process. 1 Error 0 No Error
18 CEBE	Command End Bit Error Occurs when detecting that the end bit of a command response is 0. This bit will be not asserted in tuning process. 1 End Bit Error Generated 0 No Error
17 CCE	Command CRC Error Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this bit is set when detecting a CRC error in the command response. • The uSDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the uSDHC shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict. This bit will be not asserted in tuning process. 1 CRC Error Generated. 0 No Error
16 CTOE	Command Timeout Error Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the uSDHC detects a CMD line conflict, in which case a Command CRC Error shall also be set (as shown in Interrupt Status (uSDHC_INT_STATUS)), this bit shall be set without waiting for 64 SDCLK cycles. This is because the command will be aborted by the uSDHC.

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	This bit will be not asserted in tuning process. 1 Time out 0 No Error
15 Reserved	This read-only field is reserved and always has the value 0.
14 TP	Tuning Pass:(only for SD3.0 SDR104 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTE	Re-Tuning Event: (only for SD3.0 SDR104 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. Host Controller requests Host Driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 1 Re-Tuning should be performed 0 Re-Tuning is not required
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINT	Card Interrupt This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the uSDHC will detect the Card Interrupt without the SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the Host System. Writing this bit to 1 can clear this bit, but as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. When this status has been set, and the Host Driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the Host System. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal may not be asserted), write 1 to clear this bit, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again. 1 Generate Card Interrupt 0 No Card Interrupt
7 CRM	Card Removal This status bit is set if the Card Inserted bit in the Present State register changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. In order to leave it cleared, clear the Card Removal Status Enable bit in Interrupt Status Enable register. 1 Card removed 0 Card state unstable or inserted
6 CINS	Card Insertion This status bit is set if the Card Inserted bit in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted. In order to leave it cleared, clear the Card Inserted Status Enable bit in Interrupt Status Enable register. 1 Card inserted 0 Card state unstable or removed
5 BRR	Buffer Read Ready This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Read Enable bit in the Present State register for additional information. This bit indicates that cmd19 is finished in tuning process. 1 Ready to read buffer 0 Not ready to read buffer
4 BWR	Buffer Write Ready This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Write Enable bit in the Present State register for additional information. 1 Ready to write buffer: 0 Not ready to write buffer
3 DINT	DMA Interrupt Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set. 1 DMA Interrupt is generated 0 No DMA Interrupt
2 BGE	Block Gap Event If the Stop At Block Gap Request bit in the Protocol Control register is set, this bit is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1. In the case of a Read Transaction: This bit is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD Bus timing). The Read Wait must be supported in order to use this function. In the case of Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 1 Transaction stopped at block gap 0 No block gap event
1 TC	Transfer Complete This bit is set when a read or write transfer is completed. In the case of a Read Transaction: This bit is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request bit in the Protocol Control register (after valid data has been read to the Host System).

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	In the case of a Write Transaction: This bit is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request bit in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this bit is set when busy is deasserted. This bit will be not asserted in tuning process. 1 Transfer complete 0 Transfer not complete
0 CC	Command Complete This bit is set when you receive the end bit of the command response (except Auto CMD12). Refer to the Command Inhibit (CMD) in the Present State register. This bit will be not asserted in tuning process. 1 Command complete 0 Command not complete

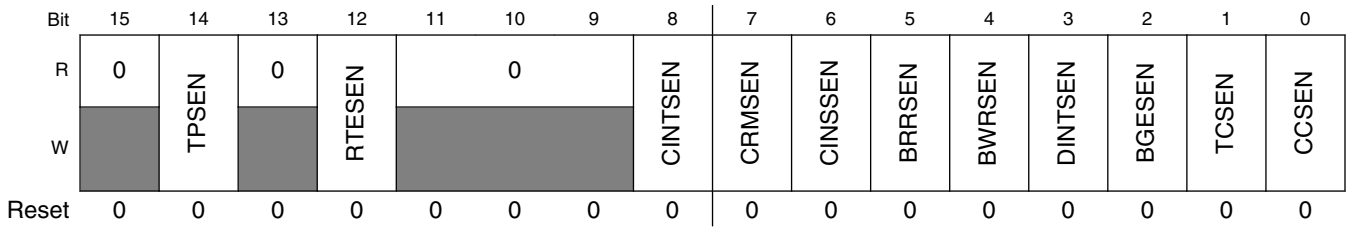
56.8.14 Interrupt Status Enable (uSDHCx_INT_STATUS_EN)

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any bit is cleared, the corresponding Interrupt Status bit is also cleared (i.e. when the bit in this register is cleared, the corresponding bit in Interrupt Status Register is always 0).

- Depending on IABG bit setting, uSDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the Card Interrupt, asserted from the card, to the time the Host System is informed.
- To detect a CMD line conflict, the Host Driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Address: Base address + 34h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAESEN	0	TNESEN	0	AC12ESEN	0	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



uSDHCx_INT_STATUS_EN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DMAESEN	DMA Error Status Enable 1 Enabled 0 Masked
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNESEN	Tuning Error Status Enable 1 Enabled 0 Masked
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12ESEN	Auto CMD12 Error Status Enable 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBESEN	Data End Bit Error Status Enable 1 Enabled 0 Masked
21 DCESEN	Data CRC Error Status Enable 1 Enabled 0 Masked
20 DTOESEN	Data Timeout Error Status Enable 1 Enabled 0 Masked
19 CIESEN	Command Index Error Status Enable 1 Enabled 0 Masked
18 CEBESEN	Command End Bit Error Status Enable 1 Enabled 0 Masked

Table continues on the next page...

uSDHCx_INT_STATUS_EN field descriptions (continued)

Field	Description
17 CCESEN	Command CRC Error Status Enable 1 Enabled 0 Masked
16 CTOESEN	Command Timeout Error Status Enable 1 Enabled 0 Masked
15 Reserved	This read-only field is reserved and always has the value 0.
14 TPSEN	Tuning Pass Status Enable 1 Enabled 0 Masked
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTESEN	Re-Tuning Event Status Enable 1 Enabled 0 Masked
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINTSEN	Card Interrupt Status Enable If this bit is set to 0, the uSDHC will clear the interrupt request to the system. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 1 Enabled 0 Masked
7 CRMSEN	Card Removal Status Enable 1 Enabled 0 Masked
6 CINSSEN	Card Insertion Status Enable 1 Enabled 0 Masked
5 BRRSEN	Buffer Read Ready Status Enable 1 Enabled 0 Masked
4 BWRSEN	Buffer Write Ready Status Enable 1 Enabled 0 Masked
3 DINTSEN	DMA Interrupt Status Enable

Table continues on the next page...

uSDHCx_INT_STATUS_EN field descriptions (continued)

Field	Description
	1 Enabled 0 Masked
2 BGESEN	Block Gap Event Status Enable 1 Enabled 0 Masked
1 TCSEN	Transfer Complete Status Enable 1 Enabled 0 Masked
0 CCSEN	Command Complete Status Enable 1 Enabled 0 Masked

56.8.15 Interrupt Signal Enable (uSDHCx_INT_SIGNAL_EN)

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

Address: Base address + 38h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				0		0		0							
W				DMAEIEN		TNEIEN		AC12EIEN		DEBEIEN	DCEIEN	DTOEIEN	CIEIEN	CEBEIEN	CCEIEN	CTOEIEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		0		0											
W		TPIEN		RTEIEN				CINTIEN	CRMIEN	CINSIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_INT_SIGNAL_EN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_INT_SIGNAL_EN field descriptions (continued)

Field	Description
28 DMAEIEN	DMA Error Interrupt Enable 1 Enable 0 Masked
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNEIEN	Tuning Error Interrupt Enable 1 Enabled 0 Masked
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12EIEN	Auto CMD12 Error Interrupt Enable 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBEIEN	Data End Bit Error Interrupt Enable 1 Enabled 0 Masked
21 DCEIEN	Data CRC Error Interrupt Enable 1 Enabled 0 Masked
20 DTOEIEN	Data Timeout Error Interrupt Enable 1 Enabled 0 Masked
19 CIEIEN	Command Index Error Interrupt Enable 1 Enabled 0 Masked
18 CEBEIEN	Command End Bit Error Interrupt Enable 1 Enabled 0 Masked
17 CCEIEN	Command CRC Error Interrupt Enable 1 Enabled 0 Masked
16 CTOEIEN	Command Timeout Error Interrupt Enable 1 Enabled 0 Masked
15 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_INT_SIGNAL_EN field descriptions (continued)

Field	Description
14 TPIEN	Tuning Pass Interrupt Enable 1 Enabled 0 Masked
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTEIEN	Re-Tuning Event Interrupt Enable 1 Enabled 0 Masked
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINTIEN	Card Interrupt Interrupt Enable 1 Enabled 0 Masked
7 CRMIEN	Card Removal Interrupt Enable 1 Enabled 0 Masked
6 CINSIEN	Card Insertion Interrupt Enable 1 Enabled 0 Masked
5 BRRIEN	Buffer Read Ready Interrupt Enable 1 Enabled 0 Masked
4 BWRIEN	Buffer Write Ready Interrupt Enable 1 Enabled 0 Masked
3 DINTIEN	DMA Interrupt Enable 1 Enabled 0 Masked
2 BGEIEN	Block Gap Event Interrupt Enable 1 Enabled 0 Masked
1 TCIEN	Transfer Complete Interrupt Enable 1 Enabled 0 Masked
0 CCIEN	Command Complete Interrupt Enable 1 Enabled 0 Masked

56.8.16 Auto CMD12 Error Status (uSDHCx_AUTOCMD12_ERR_STATUS)

When the Auto CMD12 Error Status bit in the Status register is set, the Host Driver shall check this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in bit 04-01.This register is valid only when the Auto CMD12 Error status bit is set.

The table bwlow shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 56-11. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12

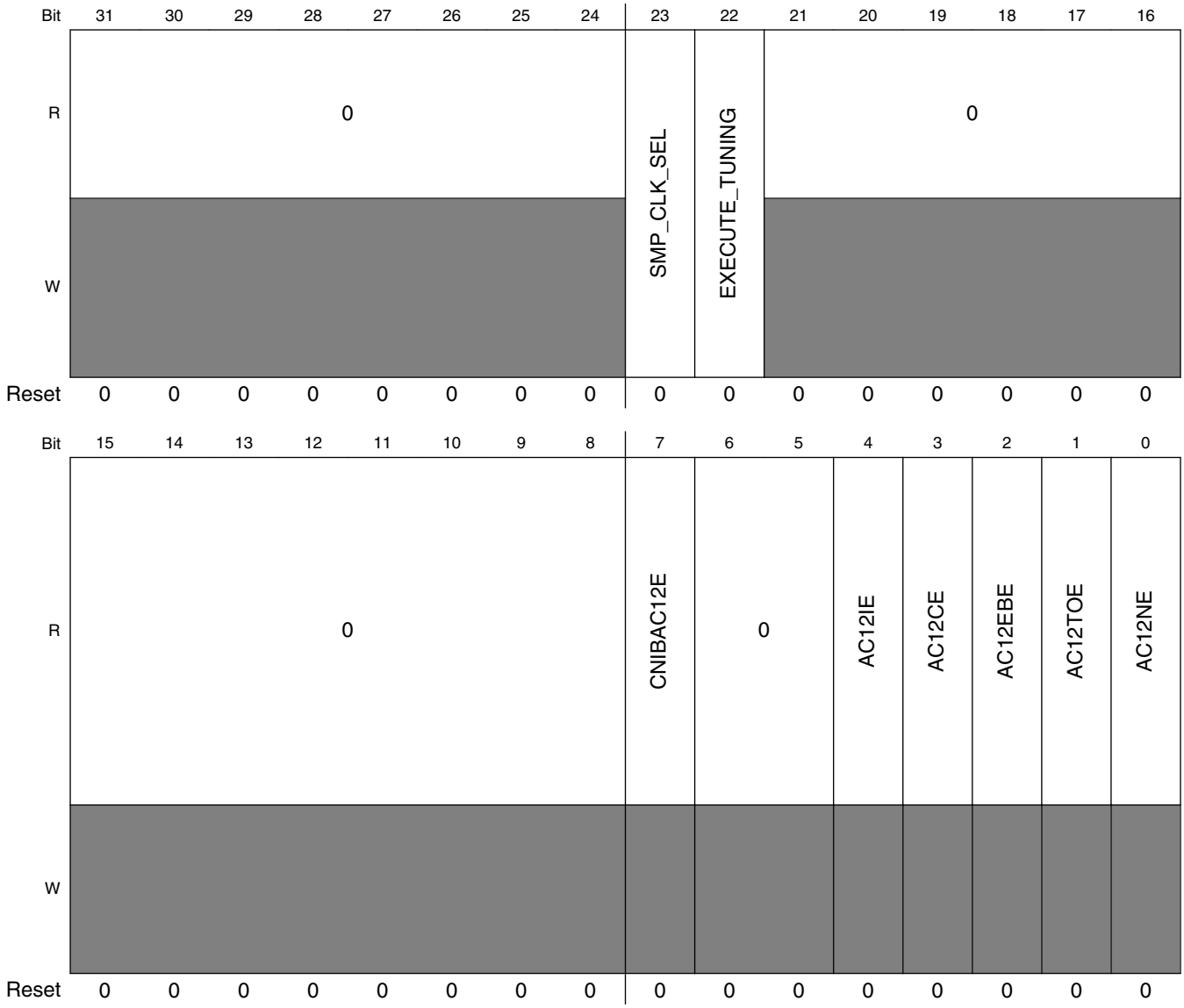
Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Type of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

1. When the uSDHC is going to issue an Auto CMD12.
   - Set bit 0 to 1 if the Auto CMD12 can't be issued due to an error in the previous command
   - Set bit 0 to 0 if the Auto CMD12 is issued
2. At the end bit of an Auto CMD12 response.
   - Check errors correspond to bits 1-4.
   - Set bits 1-4 corresponding to detected errors.
   - Clear bits 1-4 corresponding to detected errors
3. Before reading the Auto CMD12 Error Status bit 7.
   - Set bit 7 to 1 if there is a command that can't be issued
   - Clear bit 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, bit 7 shall be sampled when the driver is not writing to the Command register. So it is suggested to read this register only when the AC12E bit in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error bits (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Address: Base address + 3Ch offset



uSDHC*x*_AUTOCMD12_ERR_STATUS field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.
23 SMP_CLK_SEL	Sample Clock Select When std_tuning_en bit is set, this bit is used to select sampling clock to receive CMD and DATA. Otherwise, this bit is reserved. This bit is set by ty tuning procedure and valid after the completion of tuning(When Execute Tuning is cleared). Setting 1 means that tuning is completed successfully and setting 0 means that tuning is failed. Writing 1 to this bit is meaningless and ignored. A tuning circuit is reset by writing to 0. This bit can be cleared with setting Execute Tuning. Once the tuning circuit is reset, it will take time to complete tuning sequence. Therefore, Host Driver should keep this bit to 1 to perform re-tuning sequence to complete re-tuning sequence in a short time. Change of this bit is not allowed while the Host controller us receiving response or a read data block.

Table continues on the next page...

uSDHCx_AUTOCMD12_ERR_STATUS field descriptions (continued)

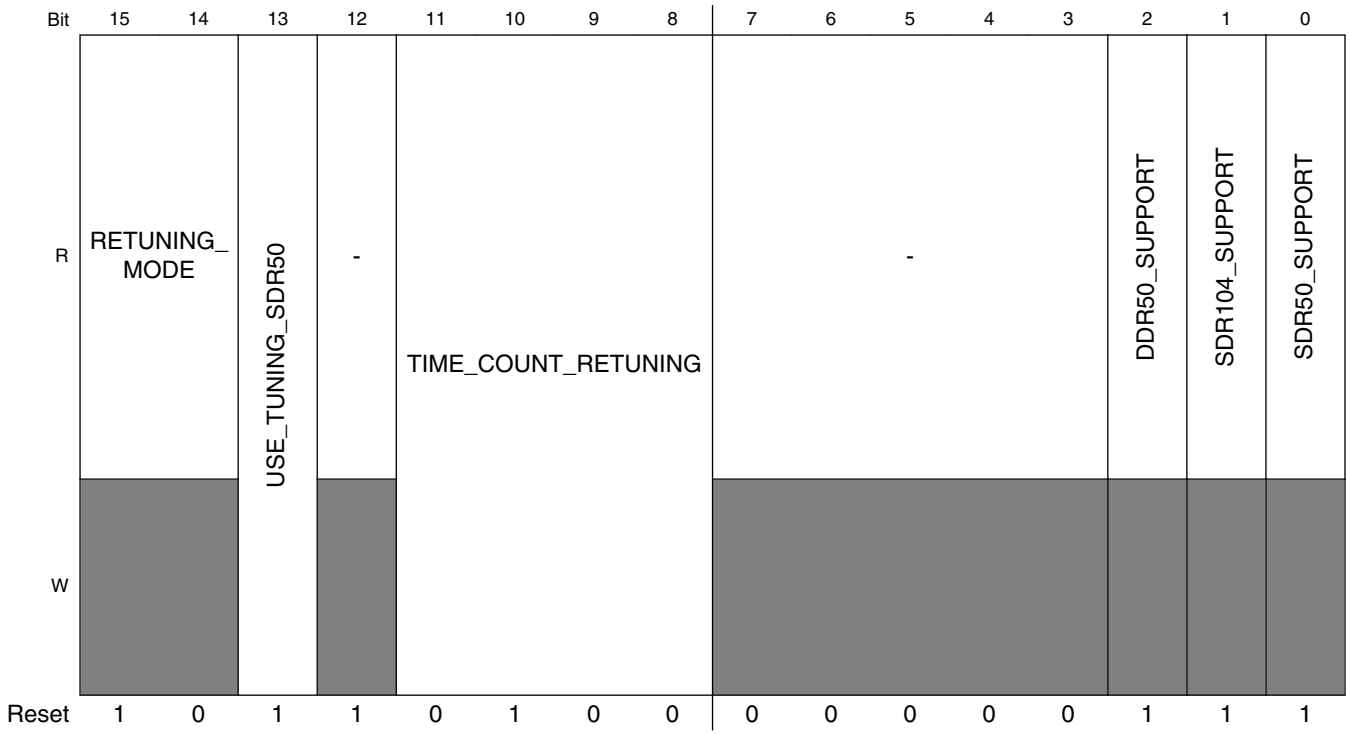
Field	Description
	1 Tuned clock is used to sample data 0 Fixed clock is used to sample data
22 EXECUTE_ TUNING	Execute Tuning When std_tuning_en bit is set, this bit is used to start tuning procedure. Otherwise, this bit is reserved. This bit is set to start tuning procedure and automatically cleared when runing procedure is completed. The result of tuning is indicated to sam_clk_sel bit. Tuning procedure is aborted by writing 0.
21–8 Reserved	This read-only field is reserved and always has the value 0.
7 CNIBAC12E	Command Not Issued By Auto CMD12 Error Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register. 1 Not Issued 0 No error
6–5 Reserved	This read-only field is reserved and always has the value 0.
4 AC12IE	Auto CMD12 / 23 Index Error Occurs if the Command Index error occurs in response to a command. 1 Error, the CMD index in response is not CMD12/23 0 No error
3 AC12CE	Auto CMD12 / 23 CRC Error Occurs when detecting a CRC error in the command response. 1 CRC Error Met in Auto CMD12/23 Response 0 No CRC error
2 AC12EBE	Auto CMD12 / 23 End Bit Error Occurs when detecting that the end bit of command response is 0 which should be 1. 1 End Bit Error Generated 0 No error
1 AC12TOE	Auto CMD12 / 23 Timeout Error Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to1, the other error status bits (2-4) have no meaning. 1 Time out 0 No error
0 AC12NE	Auto CMD12 Not Executed If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means the uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning. 1 Not executed 0 Executed

56.8.17 Host Controller Capabilities (uSDHCx_HOST_CTRL_CAP)

This register provides the Host Driver with information specific to the uSDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset.

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					VS18	VS30	VS33	SRS	DMAS	HSS	ADMAS	0	MBL[2:0]		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1



uSDHCx_HOST_CTRL_CAP field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26 VS18	Voltage Support 1.8 V This bit shall depend on the Host System ability. 1 1.8V supported 0 1.8V not supported
25 VS30	Voltage Support 3.0 V This bit shall depend on the Host System ability. 1 3.0V supported 0 3.0V not supported
24 VS33	Voltage Support 3.3V This bit shall depend on the Host System ability. 1 3.3V supported 0 3.3V not supported
23 SRS	Suspend / Resume Support This bit indicates whether the uSDHC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the Host Driver shall not issue either Suspend or Resume commands. 1 Supported 0 Not supported

Table continues on the next page...

uSDHCx_HOST_CTRL_CAP field descriptions (continued)

Field	Description
22 DMAS	DMA Support This bit indicates whether the uSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 1 DMA Supported 0 DMA not supported
21 HSS	High Speed Support This bit indicates whether the uSDHC supports High Speed mode and the Host System can supply a SD Clock frequency from 25 MHz to 50 MHz. 1 High Speed Supported 0 High Speed Not Supported
20 ADMAS	ADMA Support This bit indicates whether the uSDHC supports the ADMA feature. 1 Advanced DMA Supported 0 Advanced DMA Not supported
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 MBL[2:0]	Max Block Length This value indicates the maximum block size that the Host Driver can read and write to the buffer in the uSDHC. The buffer shall transfer block size without wait cycles. 000 512 bytes 001 1024 bytes 010 2048 bytes 011 4096 bytes
15–14 RETUNING_ MODE	Retuning Mode This bit selects retuning method. 00 Mode 1 01 Mode 2 10 Mode 3 11 Reserved
13 USE_TUNING_ SDR50	Use Tuning for SDR50 This bit is set to 1. Host controller requires tuning to operate SDR50 1 SDR50 requires tuning 0 SDR does not require tuning
12 -	Reserved
11–8 TIME_COUNT_ RETUNING	Time Counter for Retuning This bit indicates an initial value of the Retuning Timer for Re-Tuning Mode1 and 3.Setting to 0 disables Retuning Timer.

Table continues on the next page...

uSDHCx_HOST_CTRL_CAP field descriptions (continued)

Field	Description
7–3 -	
2 DDR50_ SUPPORT	DDR50 support This bit indicates support of DDR50 mode.
1 SDR104_ SUPPORT	SDR104 support This bit indicates support of SDR104 mode.
0 SDR50_ SUPPORT	SDR50 support This bit indicates support of SDR50 mode.

56.8.18 Watermark Level (uSDHCx_WTMK_LVL)

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also Configurable. There value can range from 1 to 31 words.

Address: Base address + 44h offset

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

uSDHCx_WTMK_LVL field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–24 WR_BRST_ LEN[4:0]	Write Burst Length [1] The number of words the uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).
23–16 WR_WML[7:0]	Write Watermark Level The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15–13 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_WTMK_LVL field descriptions (continued)

Field	Description
12–8 RD_BRST_ LEN[4:0]	Read Burst Length [2] The number of words the uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).
RD_WML[7:0]	Read Watermark Level The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

1. Due to system restriction, the actual burst length may not exceed 16.

2. Due to system restriction, the actual burst length may not exceed 16.

56.8.19 Mixer Control (uSDHCx_MIX_CTRL)

This register is used to DMA and data transfer. To prevent data loss, The software should check if data transfer is active before writing this register. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

Table 56-12. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

Address: Base address + 48h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-	-	-	Reserved			FBCLK_SEL	AUTO_TUNE_EN	SMP_CLK_SEL	EXE_TUNE	0					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								AC23EN	NIBBLE_POS	MSBSEL	DTDSEL	DDR_EN	AC12EN	BCEN	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_MIX_CTRL field descriptions

Field	Description
31 -	Reserved. Always write as 1
30 -	Reserved. Always write as 0.
29 -	Reserved. Always write as 0.
28–26 -	This field is reserved. Reserved
25 FBCLK_SEL	Feedback Clock Source Selection (Only used for SD3.0, SDR104 mode) 1 Feedback clock comes from the ipp_card_clk_out 0 Feedback clock comes from the loopback CLK
24 AUTO_TUNE_EN	Auto Tuning Enable (Only used for SD3.0, SDR104 mode) 1 Enable auto tuning 0 Disable auto tuning
23 SMP_CLK_SEL	When STD_TUNING_EN is 0, this bit is used to select Tuned clock or Fixed clock to sample data / cmd (Only used for SD3.0, SDR104 mode) 1 Tuned clock is used to sample data / cmd 0 Fixed clock is used to sample data / cmd
22 EXE_TUNE	Execute Tuning: (Only used for SD3.0, SDR104 mode) When STD_TUNING_EN is 0, this bit is set to 1 to indicate the Host Driver is starting tuning procedure. Tuning procedure is aborted by writing 0. 1 Execute Tuning 0 Not Tuned or Tuning Completed
21–8 Reserved	This read-only field is reserved and always has the value 0.
7 AC23EN	Auto CMD23 Enable When this bit is set to 1, the Host Controller issues a CMD23 automatically before issuing a command specified in the Command Register.
6 NIBBLE_POS	In DDR 4-bit mode nibble position indictation. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.
5 MSBSEL	Multi / Single Block Select This bit enables multiple block DATA line data transfers. For any other commands, this bit can be set to 0. If this bit is 0, it is not necessary to set the Block Count register. (Refer to Command Transfer Type (uSDHC_CMD_XFR_TYP)). 1 Multiple Blocks 0 Single Block
4 DTDSEL	Data Transfer Direction Select This bit defines the direction of DATA line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the uSDHC and is set to 0 for all other commands.

Table continues on the next page...

uSDHCx_MIX_CTRL field descriptions (continued)

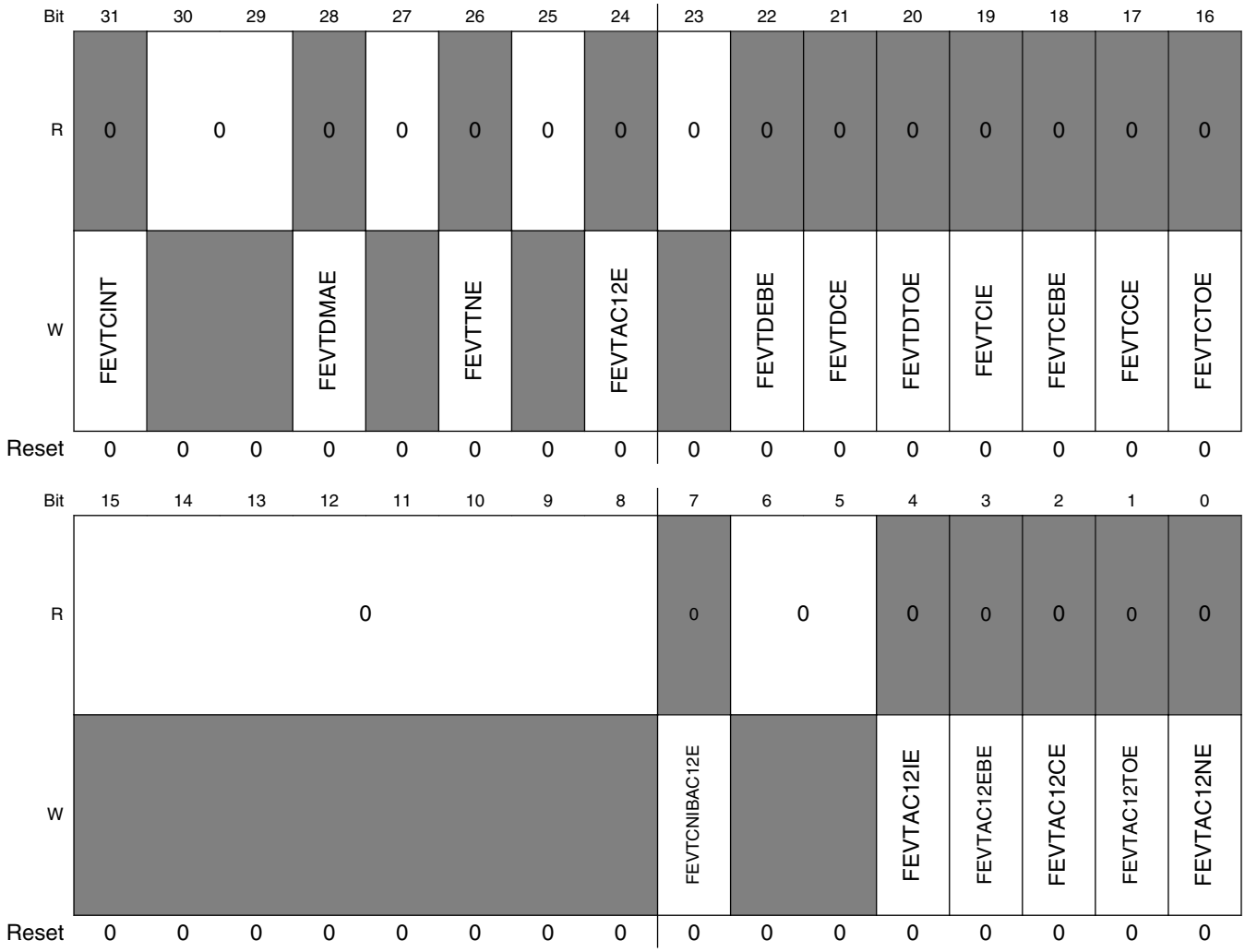
Field	Description
	1 Read (Card to Host) 0 Write (Host to Card)
3 DDR_EN	Dual Data Rate mode selection
2 AC12EN	Auto CMD12 Enable Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the uSDHC will issue a CMD12 automatically when the last block transfer has completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the uSDHC will ignore this bit no matter it is set or not. 1 Enable 0 Disable
1 BCEN	Block Count Enable This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 1 Enable 0 Disable
0 DMAEN	DMA Enable This bit enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the Host Driver sets the DPSEL bit of this register. Whether the Simple DMA or the Advanced DMA is active depends on the DMA Select field of the Protocol Control register. 1 Enable 0 Disable

56.8.20 Force Event (uSDHCx_FORCE_EVENT)

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status Register can be written if the corresponding bit of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding bit of Interrupt Status Register. A read from this register always results in 0's. In order to change the corresponding status bits in the Interrupt Status Register, make sure to set IPGEN bit in System Control Register so that IPG_CLK is always active.

Forcing a card interrupt will generate a short pulse on the DATA1 line, and the driver may treat this interrupt as a normal interrupt. The interrupt service routine may skip polling the card interrupt factor as the interrupt is self cleared.

Address: Base address + 50h offset



uSDHC*x*_FORCE_EVENT field descriptions

Field	Description
31 FEVTCINT	Force Event Card Interrupt Writing 1 to this bit generates a short low-level pulse on the internal DATA1 line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine may treat this interrupt as a normal interrupt from the external card.
30–29 Reserved	This read-only field is reserved and always has the value 0.
28 FEVTDMAE	Force Event DMA Error Forces the DMAE bit of Interrupt Status Register to be set.
27 Reserved	This read-only field is reserved and always has the value 0.
26 FEVTTNE	Force Tuning Error Forces the TNE bit of Interrupt Status Register to be set.
25 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_FORCE_EVENT field descriptions (continued)

Field	Description
24 FEVTAC12E	Force Event Auto Command 12 Error Forces the AC12E bit of Interrupt Status Register to be set.
23 Reserved	This read-only field is reserved and always has the value 0.
22 FEVTDEBE	Force Event Data End Bit Error Forces the DEBE bit of Interrupt Status Register to be set.
21 FEVTDCE	Force Event Data CRC Error Forces the DCE bit of Interrupt Status Register to be set.
20 FEVTDTOE	Force Event Data Time Out Error Force the DTOE bit of Interrupt Status Register to be set.
19 FEVTCIE	Force Event Command Index Error Forces the CCE bit of Interrupt Status Register to be set.
18 FEVTCEBE	Force Event Command End Bit Error Forces the CEBE bit of Interrupt Status Register to be set.
17 FEVTCCE	Force Event Command CRC Error Forces the CCE bit of Interrupt Status Register to be set.
16 FEVTCTOE	Force Event Command Time Out Error Forces the CTOE bit of Interrupt Status Register to be set.
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 FEVTCNIBAC12E	Force Event Command Not Executed By Auto Command 12 Error Forces the CNIBAC12E bit in the Auto Command12 Error Status Register to be set.
6–5 Reserved	This read-only field is reserved and always has the value 0.
4 FEVTAC12IE	Force Event Auto Command 12 Index Error Forces the AC12IE bit in the Auto Command12 Error Status Register to be set.
3 FEVTAC12EBE	Force Event Auto Command 12 End Bit Error Forces the AC12EBE bit in the Auto Command12 Error Status Register to be set.
2 FEVTAC12CE	Force Event Auto Command 12 CRC Error Forces the AC12CE bit in the Auto Command12 Error Status Register to be set.
1 FEVTAC12TOE	Force Event Auto Command 12 Time Out Error Forces the AC12TOE bit in the Auto Command12 Error Status Register to be set.
0 FEVTAC12NE	Force Event Auto Command 12 Not Executed Forces the AC12NE bit in the Auto Command12 Error Status Register to be set.

56.8.21 ADMA Error Status Register (uSDHCx_ADMA_ERR_STATUS)

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

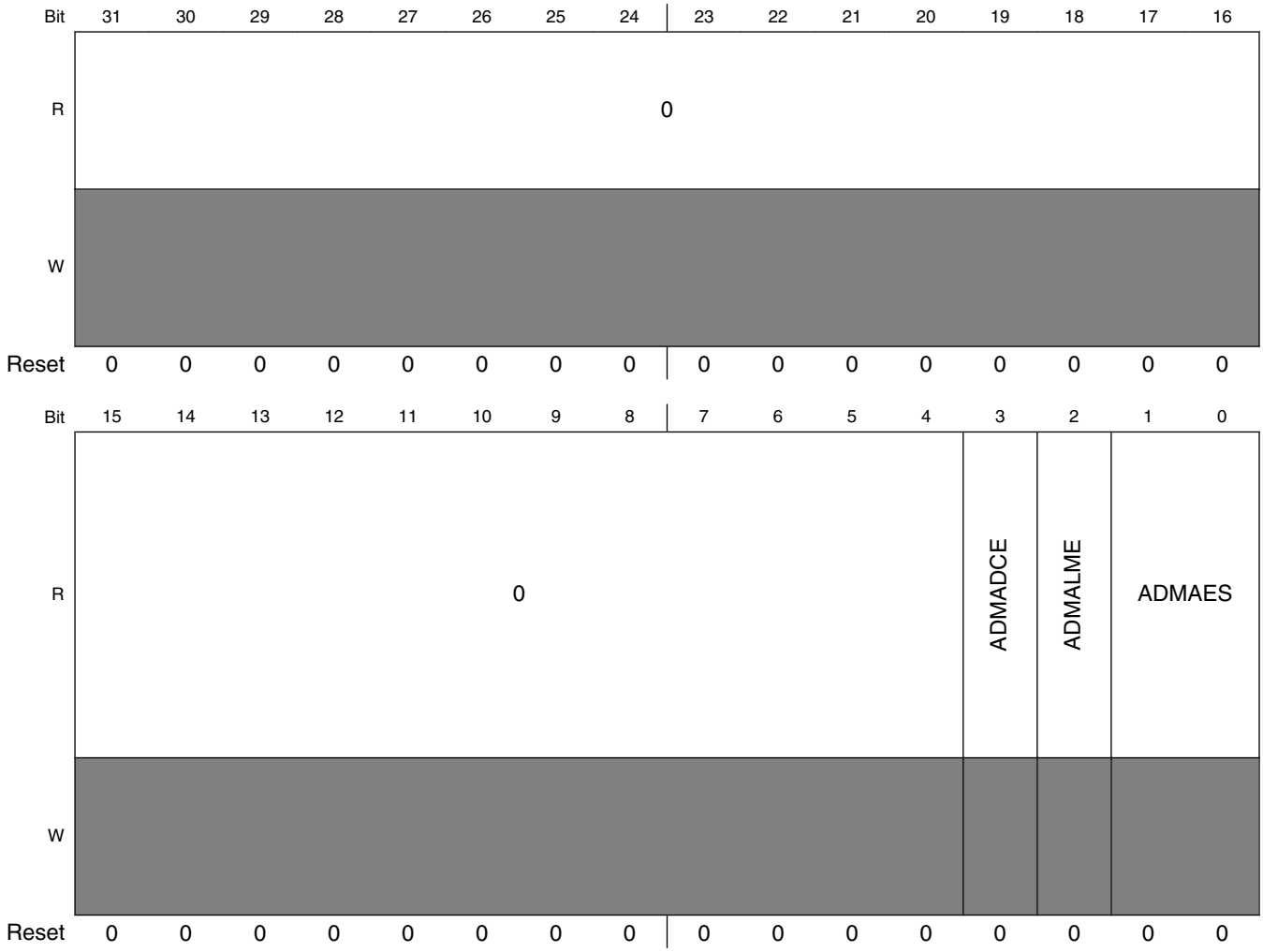
In case of a write operation, the Host Driver should use the ACMD22 to get the number of the written block, rather than using this information, since unwritten data may exist in the Host Controller.

The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The Host Driver can distinguish this error by reading the Valid bit of the error descriptor.

Table 56-13. ADMA Error State Coding

D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
00	ST_STOP (Stop DMA)	Holds the address of the next executable Descriptor command
01	ST_FDS (Fetch Descriptor)	Holds the valid Descriptor address
10	ST_CADR (Change Address)	No ADMA Error is generated
11	ST_TFR (Transfer Data)	Holds the address of the next executable Descriptor command

Address: Base address + 54h offset



uSDHC*x*_ADMA_ERR_STATUS field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value 0.
3 ADMADCE	ADMA Descritor Error This error occurs when invalid descriptor fetched by ADMA. 1 Error 0 No Error
2 ADMALME	ADMA Length Mismatch Error This error occurs in the following 2 cases: • While the Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length. • Total data length cannot be divided by the block length. 1 Error 0 No Error
ADMAES	ADMA Error State (when ADMA Error is occurred)

Table continues on the next page...

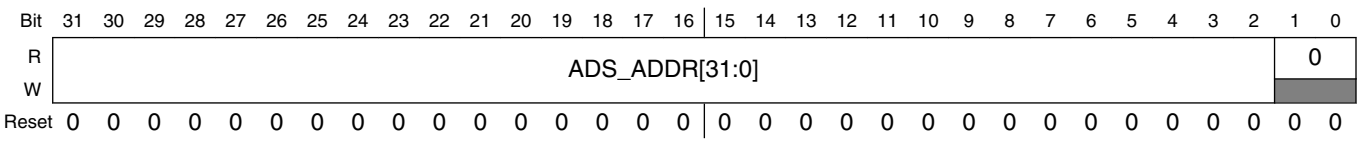
uSDHCx_ADMA_ERR_STATUS field descriptions (continued)

Field	Description
	This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. Refer to ADMA Error Status Register (uSDHC_ADMA_ERR_STATUS) for more details.

56.8.22 ADMA System Address (uSDHCx_ADMA_SYS_ADDR)

This register contains the physical system memory address used for ADMA transfers.

Address: Base address + 58h offset



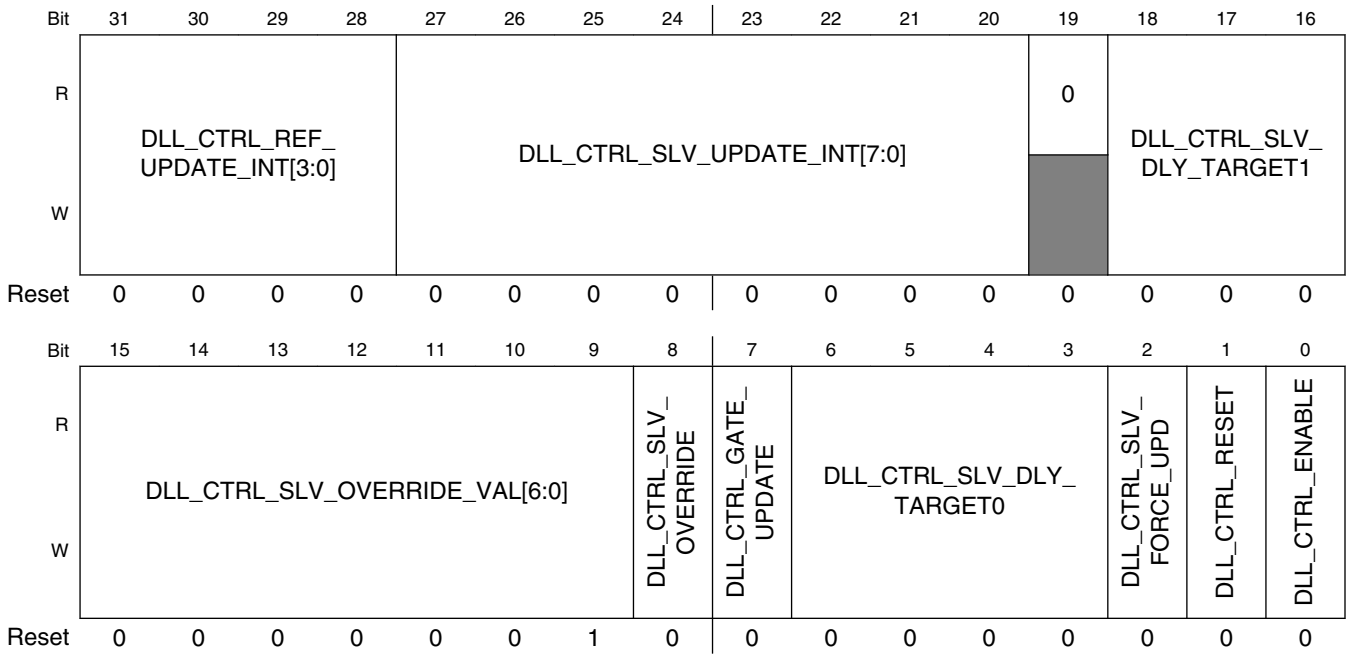
uSDHCx_ADMA_SYS_ADDR field descriptions

Field	Description
31–2 ADS_ ADDR[31:0]	ADMA System Address This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the Host Driver shall set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register shall hold the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned. Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions .
Reserved	This read-only field is reserved and always has the value 0.

56.8.23 DLL (Delay Line) Control (uSDHCx_DLL_CTRL)

This register contains control bits for DLL.

Address: Base address + 60h offset



uSDHCx_DLL_CTRL field descriptions

Field	Description
31–28 DLL_CTRL_REF_UPDATE_INT[3:0]	DLL control loop update interval. The interval cycle is (2 + REF_UPDATE_INT) * REF_CLOCK. By default, the DLL control loop shall update every two REF_CLOCK cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 DLL_CTRL_SLV_UPDATE_INT[7:0]	Slave delay line update interval. If default 0 is used, it means 256 cycles of REF_CLOCK. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 DLL_CTRL_SLV_DLY_TARGET1	Refer to DLL_CTRL_SLV_DLY_TARGET0 below.
15–9 DLL_CTRL_SLV_OVERRIDE_VAL[6:0]	When SLV_OVERRIDE = 1 This field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.

Table continues on the next page...

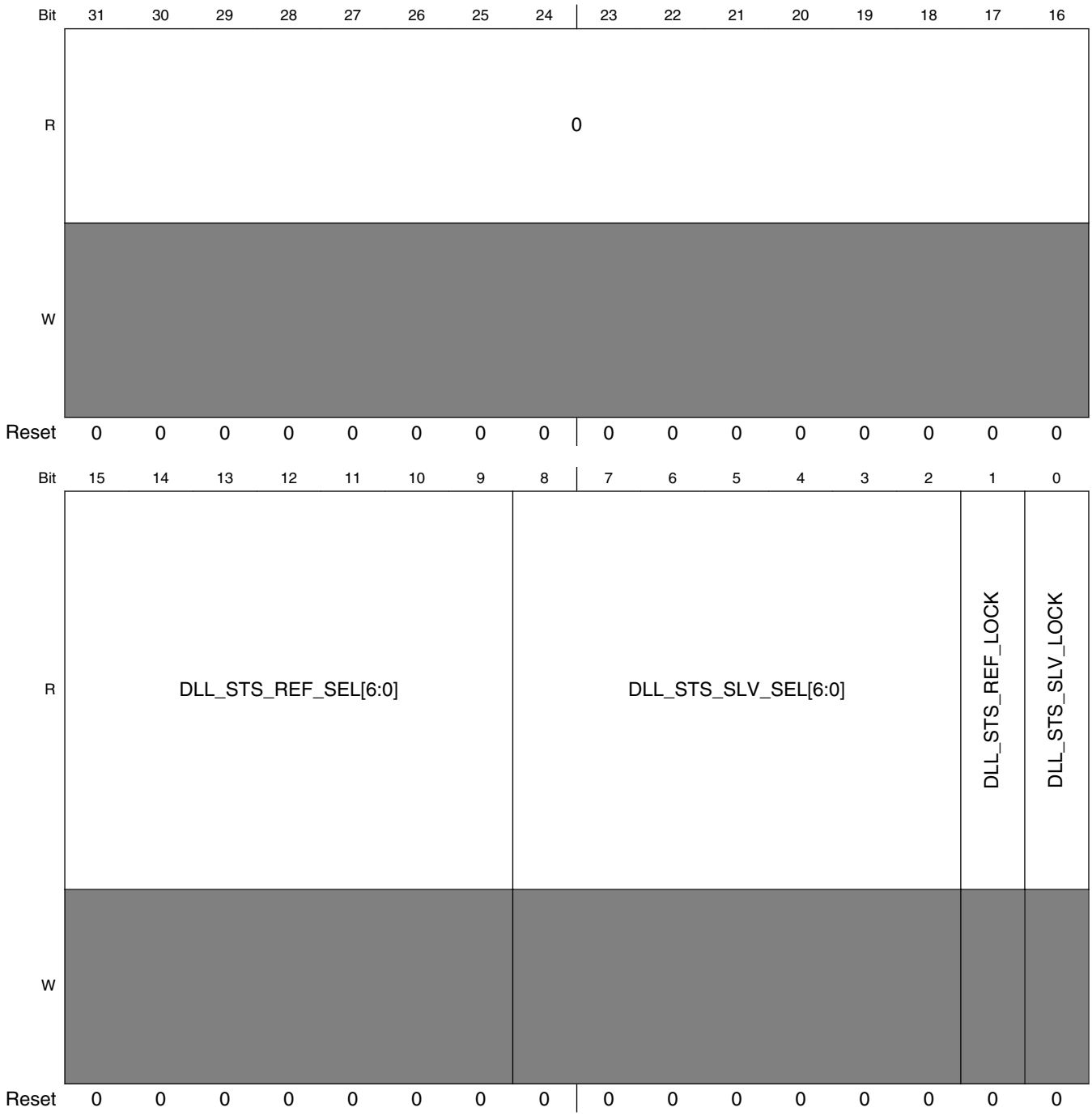
uSDHC*x*_DLL_CTRL field descriptions (continued)

Field	Description
8 DLL_CTRL_ SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0
7 DLL_CTRL_ GATE_UPDATE	Set this bit to 1 to prevent the DLL from updating (since when clock_in exists, glitches may appear during DLL updates). This bit may be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6–3 DLL_CTRL_ SLV_DLY_ TARGET0	The delay target for the uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is ((DLL_CTRL_SLV_DLY_TARGET1 +1)* REF_CLOCK / 2) / 16 So the input read-clock can be delayed relative input data from (REF_CLOCK / 2) / 16 to REF_CLOCK * 4.
2 DLL_CTRL_ SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function may not work when uSDHC is working on data / cmd / response.
1 DLL_CTRL_ RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 DLL_CTRL_ ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

56.8.24 DLL Status (uSDHC*x*_DLL_STATUS)

This register contains the DLL status information. All bits are read only and will read the same as the power-reset value.

Address: Base address + 64h offset



uSDHCx_DLL_STATUS field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–9 DLL_STS_REF_SEL[6:0]	Reference delay line select taps. This is encoded by 7 bits for 127 taps.
8–2 DLL_STS_SLV_SEL[6:0]	Slave delay line select status. This is the instant value generated from reference chain. Since the reference chain can only be updated when REF_CLOCK is detected, this value should be the right value to be updated when the reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

56.8.25 CLK Tuning Control and Status (uSDHCx_CLK_TUNE_CTRL_STATUS)

This register contains the Clock Tuning Control status information. All bits are read only and will read the same as the power-reset value. This register is added to support SD3.0 UHS-I SDR104 mode.

Address: Base address + 68h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRE_ERR	TAP_SEL_PRE[6:0]							TAP_SEL_OUT[3:0]				TAP_SEL_POST[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NXT_ERR	DLY_CELL_SET_PRE[6:0]							DLY_CELL_SET_OUT[6:0]				DLY_CELL_SET_POST[6:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CLK_TUNE_CTRL_STATUS field descriptions

Field	Description
31 PRE_ERR	PRE error which means the number of delay cells added on the feedback clock is too small. It is valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.

Table continues on the next page...

uSDHCx_CLK_TUNE_CTRL_STATUS field descriptions (continued)

Field	Description
30–24 TAP_SEL_ PRE[6:0]	Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is enabled, TAP_SEL_PRE will be updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.
23–20 TAP_SEL_ OUT[3:0]	Reflect the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.
19–16 TAP_SEL_ POST[3:0]	Reflect the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.
15 NXT_ERR	NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
14–8 DLY_CELL_ SET_PRE[6:0]	Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.
7–4 DLY_CELL_ SET_OUT[6:0]	Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.
DLY_CELL_ SET_POST[6:0]	Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.

56.8.26 Vendor Specific Register (uSDHC*x*_VEND_SPEC)

This register contains the vendor specific control / status register.

Address: Base address + C0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMD_BYTE_EN	-	-	-	-				INT_ST_VAL[7:0]							
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CRC_CHK_DIS	CARD_CLK_SOFT_EN	IPG_PERCLK_SOFT_EN	HCLK_SOFT_EN	IPG_CLK_SOFT_EN	-	-	FRC_SDCLK_ON	CLKONJ_IN_ABORT	WP_POL	CD_POL	DAT3_CD_POL	AC12_WR_CHKBUSY_EN	CONFLICT_CHK_EN	VSELECT	EXT_DMA_EN
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	1

uSDHC*x*_VEND_SPEC field descriptions

Field	Description
31 CMD_BYTE_EN	Byte access 0 Disable 1 Enable
30 -	Reserved. Always write as 0.

Table continues on the next page...

uSDHCx_VEND_SPEC field descriptions (continued)

Field	Description
29 -	Reserved. Always write as 1.
28 -	Reserved. Always write as 0.
27–24 -	Reserved. Always write as 4'b0000.
23–16 INT_ST_VAL[7:0]	Internal State Value Internal state value, reflecting the corresponding state value selected by Debug Select field. This field is read-only and write to this field does not have effect.
15 CRC_CHK_DIS	CRC Check Disable 0 Check CRC16 for every read data packet and check CRC bits for every write data packet 1 Ignore CRC16 check for every read data packet and ignore CRC bits check for every write data packet
14 CARD_CLK_ SOFT_EN	Card Clock Software Enable 0 Gate off the sd_clk 1 Enable the sd_clk
13 IPG_PERCLK_ SOFT_EN	IPG_PERCLK Software Enable 0 Gate off the IPG_PERCLK 1 Enable the IPG_PERCLK
12 HCLK_SOFT_EN	AHB Clock Software Enable **NOTE:** Hardware auto-enables the AHB clock when the internal DMA is enabled even if HCLK_SOFT_EN is 0. 0 Gate off the AHB clock. 1 Enable the AHB clock.
11 IPG_CLK_ SOFT_EN	IPG_CLK Software Enable 0 Gate off the IPG_CLK 1 Enable the IPG_CLK
10 -	Reserved. Always write as 0.
9 -	Reserved. Always write as 0.
8 FRC_SDCLK_ ON	Force CLK output active 0 CLK active or inactive is fully controlled by the hardware. 1 Force CLK active.
7 CLKONJ_IN_ ABORT	Only for debug. Force CLK output active when sending Abort command:

Table continues on the next page...

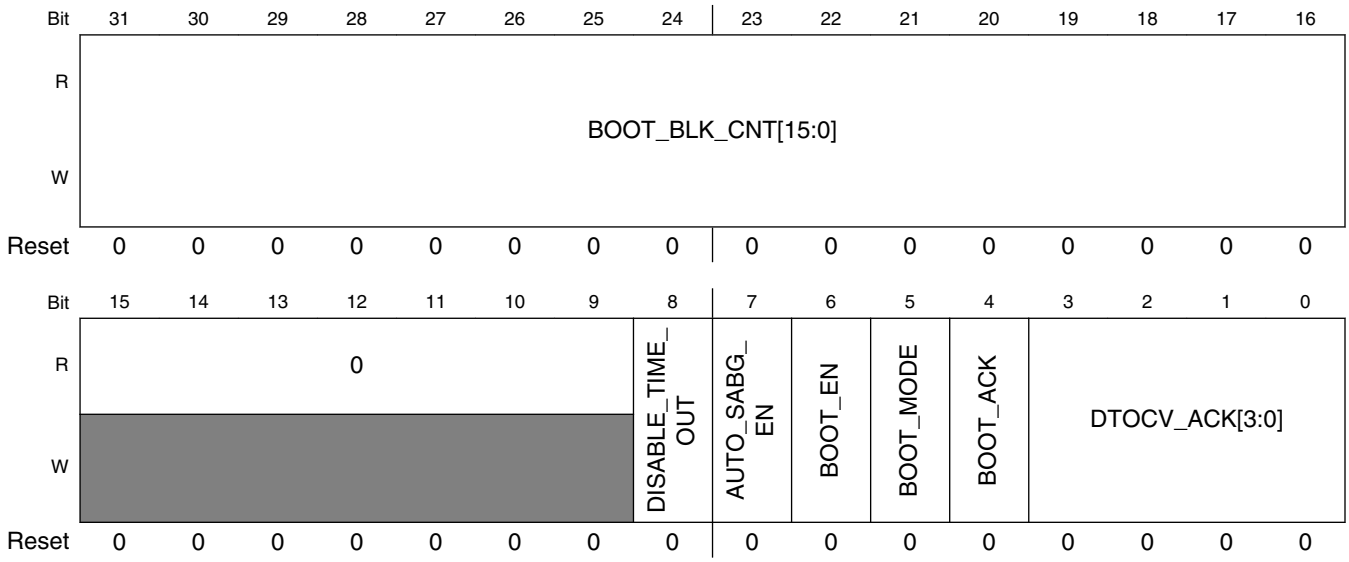
uSDHC*x*_VEND_SPEC field descriptions (continued)

Field	Description
	0 The CLK output is active when sending abort command while data is transmitting even if the internal FIFO is full (for read) or empty (for write). 1 The CLK output is inactive when sending abort command while data is transmitting if the internal FIFO is full (for read) or empty (for write).
6 WP_POL	Only for debug. Polarity of the WP pin: 0 WP pin is high active. 1 WP pin is low active.
5 CD_POL	Only for debug. Polarity of the CD_B pin: 0 CD_B pin is low active. 1 CD_B pin is high active.
4 DAT3_CD_POL	Only for debug. Polarity of DATA3 pin when it is used as card detection. 0 Card detected when DATA3 is high. 1 Card detected when DATA3 is low.
3 AC12_WR_ CHKBUSY_EN	Check busy enable after auto CMD12 for write data packet 0 Do not check busy after auto CMD12 for write data packet 1 Check busy after auto CMD12 for write data packet
2 CONFLICT_ CHK_EN	Conflict check enable. It is not implemented in uSDHC IP. 0 Conflict check disable 1 Conflict check enable
1 VSELECT	Voltage Selection Change the value of output signal VSELECT, to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads. 1 Change the voltage to low voltage range, around 1.8 V 0 Change the voltage to high voltage range, around 3.0 V
0 EXT_DMA_EN	External DMA Request Enable Enable the request to external DMA. When the internal DMA (either Simple DMA or Advanced DMA) is not in use, and this bit is set, uSDHC will send out DMA request when the internal buffer is ready. This bit is particularly useful when transferring data by ARM platform polling mode, and it is not allowed to send out the external DMA request. By default, this bit is set. 0 In any scenario, uSDHC does not send out external DMA request. 1 When internal DMA is not active, the external DMA request will be sent out.

56.8.27 MMC Boot Register (uSDHC*x*_MMC_BOOT)

This register contains the MMC Fast Boot control register.

Address: Base address + C4h offset



uSDHC*x*_MMC_BOOT field descriptions

Field	Description
31–16 BOOT_BLK_CNT[15:0]	The value defines the Stop At Block Gap value of automatic mode. When received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Atrributes Register, bit31 - 16 of 0x04.
15–9 Reserved	This read-only field is reserved and always has the value 0.
8 DISABLE_TIME_OUT	Disable Time Out **NOTE:** When this bit is set, there is no timeout check no matter whether BOOT_EN is set or not. 0 Enable time out 1 Disable time out
7 AUTO_SABG_EN	During boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).
6 BOOT_EN	Boot mode enable 0 Fast boot disable 1 Fast boot enable
5 BOOT_MODE	Boot mode select

Table continues on the next page...

uSDHCx_MMC_BOOT field descriptions (continued)

Field	Description
	0 Normal boot 1 Alternative boot
4 BOOT_ACK	Boot ACK mode select 0 No ack 1 Ack
DTOCV_ ACK[3:0]	Boot ACK time out counter value. 0000 SDCLK x 2^13 0001 SDCLK x 2^14 0010 SDCLK x 2^15 0011 SDCLK x 2^16 0100 SDCLK x 2^17 0101 SDCLK x 2^18 0110 SDCLK x 2^19 0111 SDCLK x 2^20 1110 SDCLK x 2^27 1111 SDCLK x 2^28

56.8.28 Vendor Specific 2 Register (uSDHCx_VEND_SPEC2)

This register contains the vendor specific control 2 register.

Address: Base address + C8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ACMD23_ARGU2_EN	Reserved						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Reserved		0	0	CARD_INT_AUTO_CLR_DIS	TUNING_CMD_EN	TUNING_1bit_EN	TUNING_8bit_EN	CARD_INT_D3_TEST	SDR104_NSD_DIS	SDR104_OE_DIS	SDR104_TIMING_DIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

uSDHCx_VEND_SPEC2 field descriptions

Field	Description
31–24 -	This field is reserved.
23 ACMD23_ARGU2_EN	Default1 1 Argument2 register enable for ACMD23 sharing with SDMA system address register. Default is enable. 0 Disable
22–12 -	This field is reserved.
11–10 -	This field is reserved. Reserved
9 Reserved	This read-only field is reserved and always has the value 0.
8 Reserved	This read-only field is reserved and always has the value 0.
7 CARD_INT_AUTO_CLR_DIS	Disable the feature to clear the Card interrupt status bit when Card Interrupt status enable bit is cleared. Only for debug. 0 Card interrupt status bit (CINT) can be cleared when Card Interrupt status enable bit is 0. 1 Card interrupt status bit (CINT) can only be cleared by writting a 1 to CINT bit.
6 TUNING_CMD_EN	Enable the auto tuning circuit to check the CMD line. 0 Auto tuning circuit does not check the CMD line. 1 Auto tuning circuit checks the CMD line.
5 TUNING_1bit_EN	Enable the auto tuning circuit to check the DATA0 only. It is used with the TUNING_8bit_EN together.
4 TUNING_8bit_EN	Enable the auto tuning circuit to check the DATA[7:0]. It is used with the TUNING_1bit_EN together. **NOTE:** The format of these two bits are [TUNNING_8bit_EN:TUNNING_1bit_EN]. 00 Tuning circuit only checks the DATA[3:0]. 01 Tuning circuit only checks the DATA0. 10 Tuning circuit checks the whole DATA[7:0]. 11 Invalid.
3 CARD_INT_D3_TEST	Card Interrupt Detection Test This bit only uses for debugging. 0 Check the card interrupt only when DATA3 is high. 1 Check the card interrupt by ignoring the status of DATA3.
2 SDR104_NSD_DIS	Interrupt window after abort command is sent. This bit only uses for debugging. 0 Enable the interrupt window 9 cycles later after the end of the I/O abort command (or CMD12) is sent. 1 Enable the interrupt window 5 cycles later after the end of the I/O abort command (or CMD12) is sent.
1 SDR104_OE_DIS	CMD_OE / DATA_OE logic generation test. This bit only uses for debugging.

Table continues on the next page...

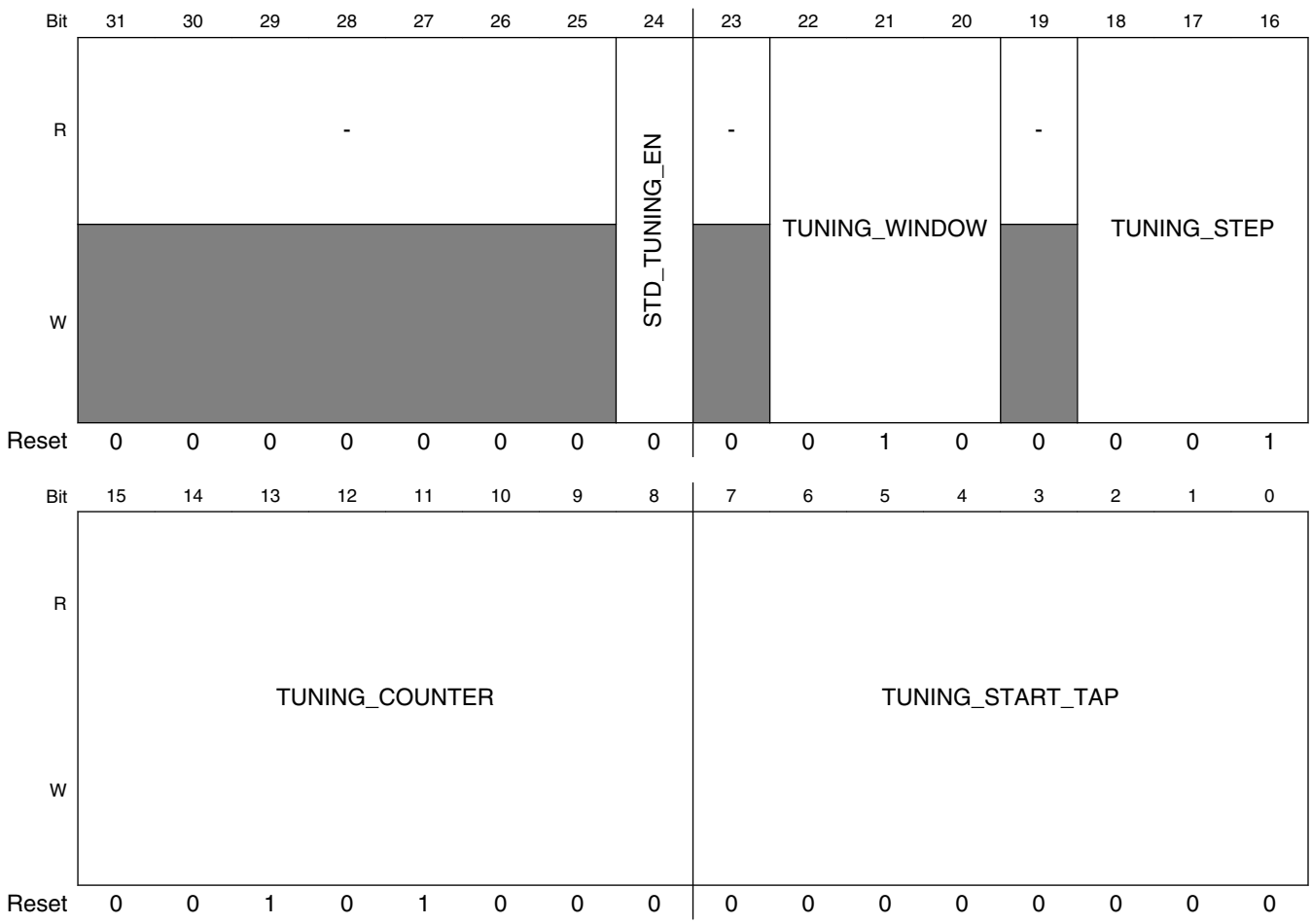
uSDHCx_VEND_SPEC2 field descriptions (continued)

Field	Description
	0 Drive the CMD_OE / DATA_OE for one more clock cycle after the end bit. 1 Stop to drive the CMD_OE / DATA_OE at once after driving the end bit.
0 SDR104_ TIMING_DIS	Timeout counter test. This bit only uses for debugging. 0 The timeout counter for Ncr changes to 80, Ncrc changes to 21. 1 The timeout counter for Ncr changes to 72, Ncrc changes to 15.

56.8.29 Tuning Control Register (uSDHCx_TUNING_CTRL)

The register contains configuration of tuning circuit.

Address: Base address + CCh offset



uSDHCx_TUNING_CTRL field descriptions

Field	Description
31–25 -	Reserved
24 STD_TUNING_EN	Standard tuning circuit and procedure enable: This bit is used to enable standard tuning circuit and procedure.
23 -	Reserved
22–20 TUNING_WINDOW	Select data window value for auto tuning
19 -	Reserved
18–16 TUNING_STEP	The increasing delay cell steps in tuning procedure.
15–8 TUNING_COUNTER	The MAX repeat CMD19 times in tuning procedure.
TUNING_START_TAP	The start dealy cell point when send first CMD19 in tuning procedure.

Chapter 57
Watchdog Timer (WDOG)

57.1 Overview

The Watchdog Timer (WDOG) protects against system failures by providing a method by which to escape from unexpected events or programming errors.

Once the WDOG is activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon timeout, the WDOG asserts the internal system reset signal, WDOG_RESET_B_DEB to the System Reset Controller (SRC).

There is also a provision for WDOG signal assertion by timeout counter expiration. There is an option of programmable interrupt generation before the counter actually times out. The time at which the interrupt needs to be generated prior to counter timeout is programmable. There is a power down counter which is enabled out of any reset (POR, Warm/Cold). This counter has a fixed timeout period of 16 seconds, upon which it asserts the WDOG signal.

Flow diagrams for the timeout counter, power down counter and interrupt operations are are shown in Flow Diagrams.

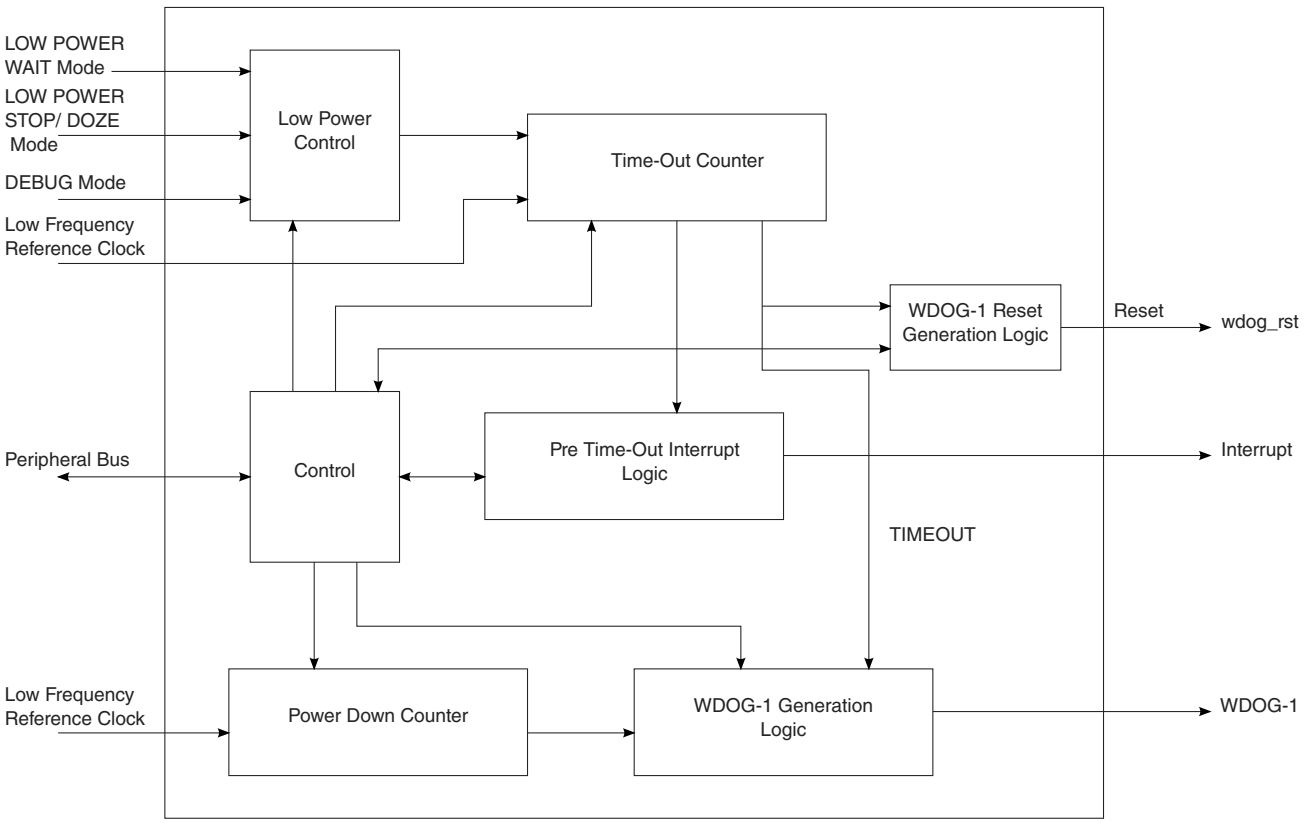


Figure 57-1. WDOG Diagram

57.1.1 Features

The WDOG features are listed below:

- Configurable timeout counter with timeout periods from 0.5 to 128 seconds which, after timeout expiration, result in the assertion of WDOG_RESET_B_DEB reset signal .
- Time resolution of 0.5 seconds
- Configurable timeout counter that can be programmed to run or stop during low-power modes
- Configurable timeout counter that can be programmed to run or stop during DEBUG mode
- Programmable interrupt generation prior to timeout
- The duration between interrupt and timeout events can be programmed from 0 to 127.5 seconds in steps of 0.5 seconds.
- Power down counter with fixed timeout period of 16 seconds, which if not disabled after reset will assert WDOG_B signal low

- Power down counter will be enabled out of any reset (POR, Warm / Cold reset) by default.

57.2 External signals

Table 57-1. WDOG External Signals

Signal	Description	Pad	Mode	Direction
WDOG1_ANY	Global WDOG signal	ENET2_RX_ER	ALT8	IO
		GPIO1_IO09	ALT1	
		LCD_DATA19	ALT2	
		LCD_RESET	ALT4	
WDOG1_B	This signal will power down the chip.	GPIO1_IO01	ALT8	IO
		GPIO1_IO08	ALT1	
		UART3_RTS_B	ALT8	
WDOG1_RST_B_DEB	This signal is a reset source for the chip.	ENET1_TX_DATA1	ALT8	O
		LCD_CLK	ALT8	
WDOG2_B	This signal will power down the chip.	LCD_VSYNC	ALT4	IO
WDOG2_RST_B_DEB	This signal is a reset source for the chip.	ENET1_TX_EN	ALT8	O
WDOG3_B	This signal will power down the chip.	GPIO1_IO00	ALT8	IO
WDOG3_RST_B_DEB	This signal is a reset source for the chip.	LCD_HSYNC	ALT4	O

57.3 Clocks

This section describes clocks and special clocking requirements of the block.

The WDOG uses the low frequency reference clock for its counter and control operations. The peripheral bus clock is used for register read/write operations.

The following table describes the clock sources for WDOG. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 57-2. WDOG Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	IP Global functional clock. All functionality inside the WDOG module is synchronized to this clock.
ipg_clk_s	ipg_clk_root	IP slave bus clock. This clock is synchronized to ipg_clk and is only used for register read/write operations.

Table continues on the next page...

Table 57-2. WDOG Clocks (continued)

Clock name	Clock Root	Description
ipg_clk_32k	ckil_sync_clk_root	Low frequency (32.768 kHz) clock that continues to run in low-power mode. It is assumed that the Clock Controller will provide this clock signal synchronized to ipg_clk in the normal mode, and switch to a non-synchronized signal in low-power mode when the ipg_clk is off.

57.4 Watchdog mechanism and system integration

The modules are disabled by default (after reset). WDOG1 will be configured during boot while WDOG2 is dedicated for secure world purposes and will be activated by TZ software if required. The TZ watchdog (TZ watchdog) module protects against TZ starvation by providing a method of escaping normal mode and forcing a switch to the TZ mode.TZ starvation is a situation where the normal OS prevents switching to the TZ mode. Such a situation is undesirable as it can compromise the system's security.

Once the TZ WDOG module is activated, it must be serviced by TZ on a periodic basis. If servicing does not take place, the timer times out. Upon a timeout, the TZ WDOG asserts a TZ-mapped interrupt that forces switching to the TZ mode. If it is still not serviced, the TZ WDOG asserts a security violation signal to the CSU. The TZ WDOG module cannot be programmed or de-activated by normal mode software.

The WDOG modules operate as follows:

- If servicing does not take place, the timer times out and the wdog_rst_b signal is activated (low)
- Interrupt can be generated before the counter actually times out
- The wdog_rst_b signal can be activated by software
- There is a power-down counter which gets enabled out of any reset. This counter has a fixed timeout period of 16 seconds upon which it will assert the ipp_wdog_b signal.

57.5 Functional description

This section provides a complete functional description of the block.

57.5.1 Timeout event

The WDOG provides timeout periods from 0.5 to 128 seconds with a time resolution of 0.5 seconds.

The user can determine the timeout period by writing to the WDOG timeout field (WT[7:0]) in the Watchdog Control Register (WDOG_WCR). The WDOG must be enabled by setting the WDE bit of Watchdog Control Register (WDOG_WCR) for the timeout counter to start running. After the WDOG is enabled, the counter is activated, loads the timeout value and begins to count down from this programmed value. The timer will time out when the counter reaches zero and the WDOG outputs a system reset signal, WDOG_RESET_B_DEB and asserts WDOG_B (WDT bit should be set in Watchdog Control Register (WDOG_WCR)).

However, the timeout condition can be prevented by reloading the counter with the new timeout value (WT[7:0] of WDOG_WCR) if a service routine (see Servicing WDOG to reload the counter) is performed before the counter reaches zero. If any system errors occur which prevent the software from servicing the Watchdog Service Register (WDOG_WSR), the timeout condition occurs. By performing the service routine, the WDOG reloads its counter to the timeout value indicated by bits WT[7:0] of the Watchdog Control Register (WDOG_WCR) and it restarts the countdown.

A system reset will reset the counter and place it in the idle state at any time during the countdown. The counter flow diagram is shown in Flow Diagrams.

NOTE

The timeout value is reloaded to the counter either at the time WDOG is enabled or after the service routine has been performed.

57.5.1.1 Servicing WDOG to reload the counter

To reload a timeout value to the counter the proper service sequence begins by writing 0x_5555 followed by 0x_AAAA to the Watchdog Service Register (WDOG_WSR). Any number of instructions can be executed between the two writes. If the WDOG_WSR is not loaded with 0x_5555 prior to writing 0x_AAAA to the WDOG_WSR, the counter is not reloaded. If any value other than 0x_AAAA is written to the WDOG_WSR after 0x_5555, the counter is not reloaded. This service sequence will reload the counter with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR). The timeout value can be changed at any point; it is reloaded when WDOG is serviced by the core.

57.5.2 Interrupt event

Prior to timeout, the WDOG can generate an interrupt which can be considered a warning that timeout will occur shortly.

The duration between interrupt event and timeout event can be controlled by writing to the WICT field of Watchdog Interrupt Control Register (WDOG_WICR). It can vary between 0 and 127.5 seconds. If the WDOG is serviced (Servicing WDOG to reload the counter) before the interrupt generation, the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR) and the interrupt will not be triggered.

57.5.3 Power-down counter event

The power-down counter inside WDOG will be enabled out of reset. This counter has a fixed timeout value of 16 seconds, after which it will drive the WDOG_B signal low.

To prevent this, the software must disable this counter by clearing the PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) within 16 seconds of reset deassertion. Once disabled, this counter can't be enabled again until the next system reset occurs. This feature is intended to prevent the hanging up of cores after reset, as WDOG is not enabled out of reset.

57.5.4 Low power modes

57.5.4.1 STOP and DOZE mode

If the WDOG timer disable bit for low power STOP and DOZE mode (WDZST) bit in the Watchdog Control Register (WDOG_WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock. If the low power enable (WDZST) bit is set, the WDOG timer operation will be suspended in low power STOP or DOZE mode. Upon exiting low power STOP or DOZE mode, the WDOG operation returns to what it was prior to entering the STOP or DOZE mode.

57.5.4.2 WAIT mode

If the WDOG timer disable bit for low power WAIT mode (WDW) bit in the Watchdog Control Register (WDOG_WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock . If the low power WAIT enable (WDW) bit is set, the WDOG timer operation will be suspended. Upon exiting low power WAIT mode, the WDOG operation returns to what it was prior to entering the WAIT mode.

NOTE

The WDOG timer won't be able to detect events that happen for periods shorter than one low frequency reference clock cycle. For example, in repeated WAIT mode entry or exit, if the RUN mode time is less than one low frequency reference clock cycle and if the WDW bit is set, the WDOG timer may never time out, even though the system is in RUN mode for a finite duration; WDOG may not see a low frequency reference clock edge during its wake time.

57.5.5 Debug mode

The WDOG timer can be configured for continual operation, or for suspension during debug mode. If the WDOG debug enable (WDBG) bit is set in the Watchdog Control Register (WDOG_WCR), the WDOG timer operation is suspended in debug mode. If the WDBG bit is set and the debug mode is entered, WDOG timer operation is suspended after two low frequency reference clocks. Similarly, WDOG timer operation continues after two low frequency reference clocks of debug mode exit. Register read and write accesses in debug mode continue to function normally. Also, while in debug mode, the WDE bit of Watchdog Control Register (WDOG_WCR) can be enabled/disabled directly. If the WDOG debug enable (WDBG) bit is cleared then WDOG timer operation is not suspended. The power-down counter is not affected by debug mode entry/exit.

NOTE

If the WDE bit of Watchdog Control Register (WDOG_WCR) is set/cleared while in debug mode, it remains set/cleared even after exiting debug mode.

57.5.6 Operations

57.5.6.1 Watchdog reset generation

The WDOG generated reset signal WDOG_RESET_B_DEB is asserted by the following operations:

- A software write to the Software Reset Signal (SRS) bit of the Watchdog Control Register (WDOG_WCR).
- WDOG timeout. See Timeout event.

The $\overline{\text{wdog_rst}}$ will be asserted for one clock cycle of low frequency reference clock for both a timeout condition and a software write occurrence. It remains asserted for 1 clock cycle of low frequency reference clock even if a system reset is asserted in between. Figure 57-3 shows the timing diagram of this signal due to a timeout condition.

57.5.6.2 WDOG_B generation

The WDOG asserts WDOG_B in the following scenarios:

- Software write to WDA bit of Watchdog Control Register (WDOG_WCR). WDOG_B signal remains asserted as long as the WDA bit is "0".
- WDOG timeout condition, WDT bit of Watchdog Control Register (WDOG_WCR) must be set for this scenario. A description of the timeout condition can be found in the Timeout event. WDOG_B signal remains asserted until a power-on reset (POR) occurs. It gets cleared after the POR occurs (not due to any other system reset). Figure 57-4 shows the timing diagram of WDOG_B due to timeout condition.
- WDOG power-down counter timeout, PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should not be cleared for this scenario. A description of this counter can be found in the Power-down counter event. WDOG_B signal remains asserted for one clock cycle of low frequency reference clock.

Figure 57-2 shows the scenarios under which WDOG_B gets asserted.

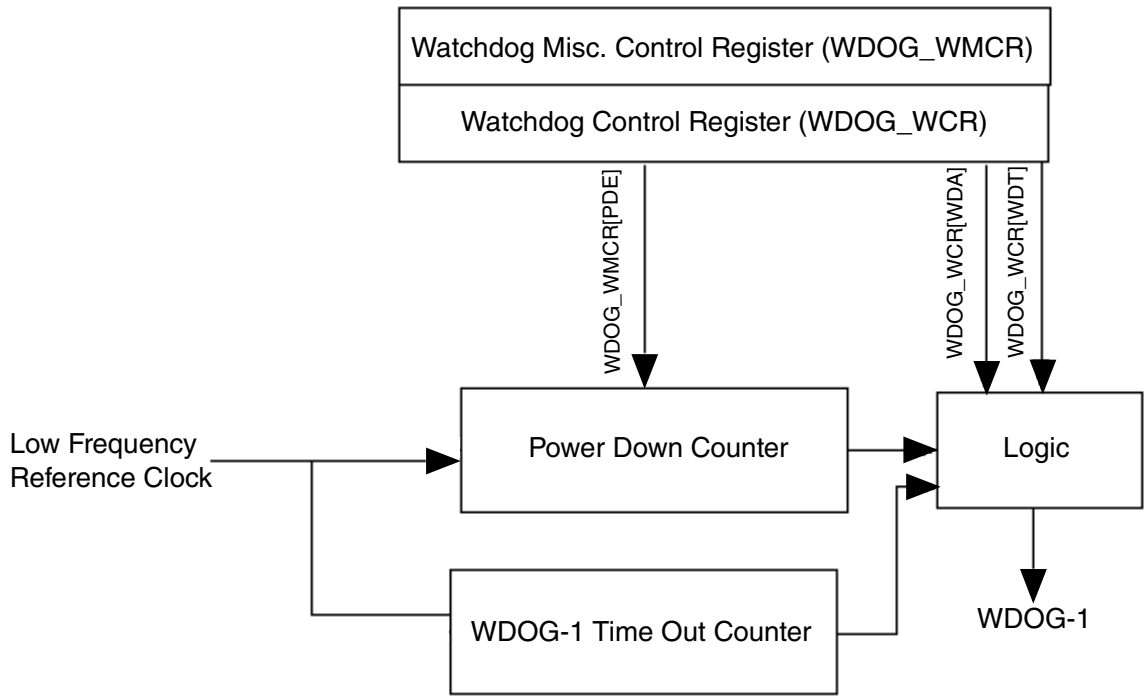


Figure 57-2. WDOG_B generation

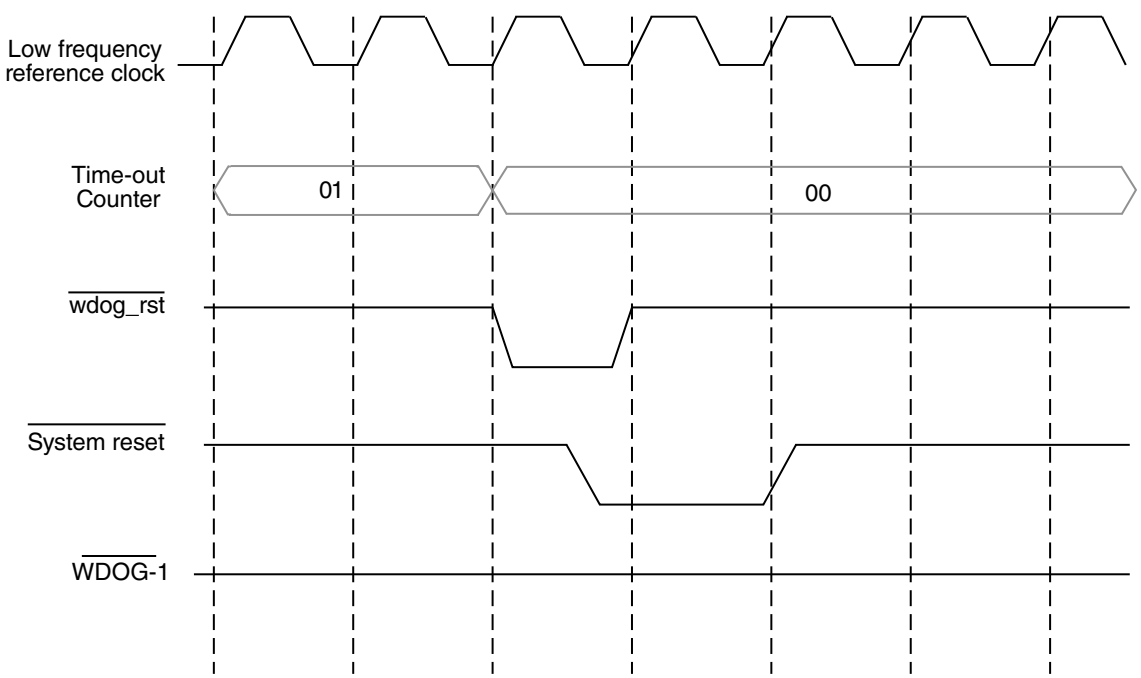


Figure 57-3. WDOG timeout vondition/WDT bit is not set

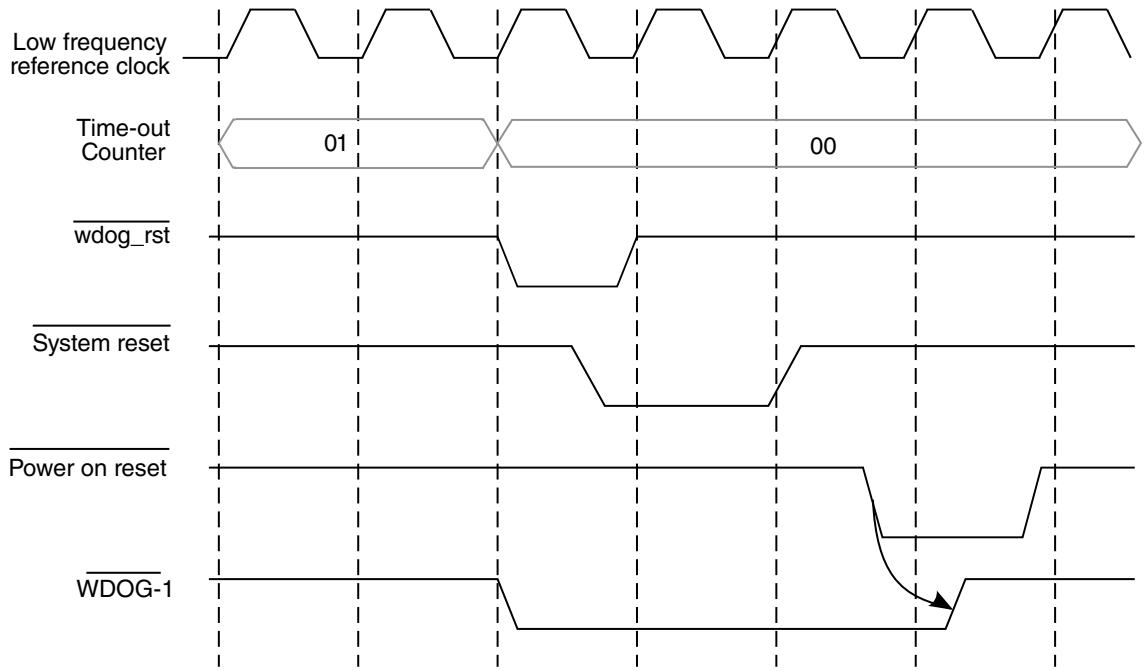


Figure 57-4. WDOG timeout condition/WDT bit is set

57.5.7 Reset

The block is reset by a system reset and the WDOG counter will be disabled. The power-down counter is enabled and starts counting.

57.5.8 Interrupt

The WDOG has the feature of Interrupt generation before timeout.

The interrupt will be generated only if the WIE bit in Watchdog Interrupt Control Register (WDOG_WICR) is set. The exact time at which the interrupt should occur (prior to timeout) depends on the value of WICT field of Watchdog Interrupt Control Register (WDOG_WICR). For example, if the WICT field has a value 0x04, then the interrupt will be generated two seconds prior to timeout. Once the interrupt is triggered the WTIS bit in Watchdog Interrupt Control Register (WDOG_WICR) will be set. The software needs to clear this bit to deassert the interrupt. If the WDOG is serviced before the interrupt generation then the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR) and interrupt would not be triggered.

57.5.9 Flow Diagrams

A flow diagram of WDOG operation is shown below.

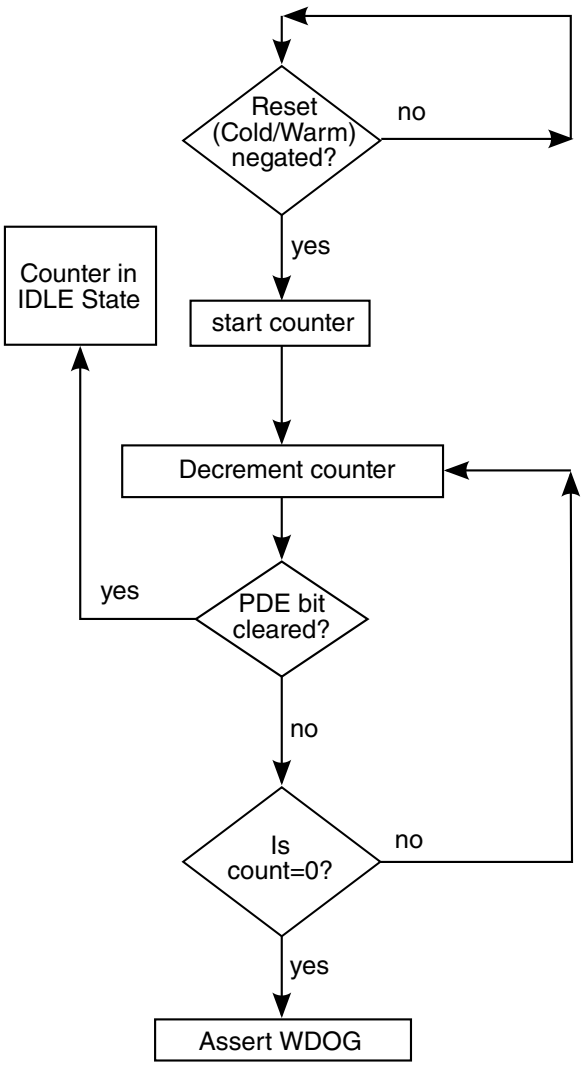


Figure 57-5. Power-Down Counter Flow Diagram

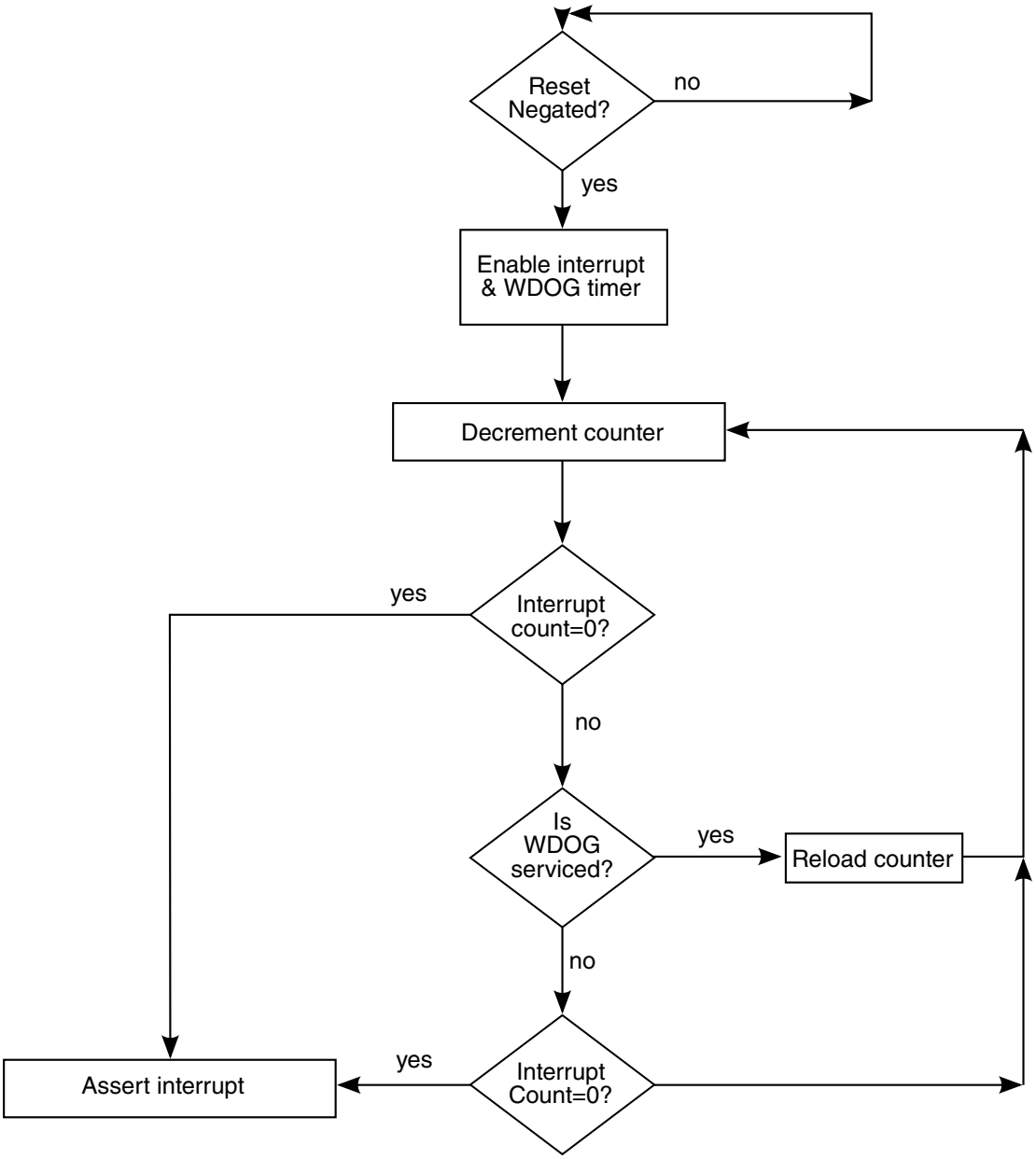


Figure 57-6. Interrupt Generation Flow Diagram

57.6 Initialization

The following sequence should be performed for WDOG initialization.

- PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should be cleared to disable the power down counter.

- WT field of Watchdog Control Register (WDOG_WCR) should be programmed for sufficient timeout value.
- WDOG should be enabled by setting WDE bit of Watchdog Control Register (WDOG_WCR) so that the timeout counter loads the WT field value of Watchdog Control Register (WDOG_WCR) and starts counting.

57.7 WDOG Memory Map/Register Definition

The WDOG has user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Byte operations can be performed on these registers. If a 32-bit access is performed,the WDOG will not generate a peripheral bus error but will behave normally, like a 16-Bit access, making read/write possible. A 32-Bit access should be avoided, as the system may go to an unknown state.

WDOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_C000	Watchdog Control Register (WDOG1_WCR)	16	R/W	0030h	57.7.1/3644
20B_C002	Watchdog Service Register (WDOG1_WSR)	16	R/W	0000h	57.7.2/3645
20B_C004	Watchdog Reset Status Register (WDOG1_WRSR)	16	R	0000h	57.7.3/3646
20B_C006	Watchdog Interrupt Control Register (WDOG1_WICR)	16	R/W	0004h	57.7.4/3647
20B_C008	Watchdog Miscellaneous Control Register (WDOG1_WMCR)	16	R/W	0001h	57.7.5/3648
20C_0000	Watchdog Control Register (WDOG2_WCR)	16	R/W	0030h	57.7.1/3644
20C_0002	Watchdog Service Register (WDOG2_WSR)	16	R/W	0000h	57.7.2/3645
20C_0004	Watchdog Reset Status Register (WDOG2_WRSR)	16	R	0000h	57.7.3/3646
20C_0006	Watchdog Interrupt Control Register (WDOG2_WICR)	16	R/W	0004h	57.7.4/3647
20C_0008	Watchdog Miscellaneous Control Register (WDOG2_WMCR)	16	R/W	0001h	57.7.5/3648
21E_4000	Watchdog Control Register (WDOG3_WCR)	16	R/W	0030h	57.7.1/3644
21E_4002	Watchdog Service Register (WDOG3_WSR)	16	R/W	0000h	57.7.2/3645
21E_4004	Watchdog Reset Status Register (WDOG3_WRSR)	16	R	0000h	57.7.3/3646
21E_4006	Watchdog Interrupt Control Register (WDOG3_WICR)	16	R/W	0004h	57.7.4/3647
21E_4008	Watchdog Miscellaneous Control Register (WDOG3_WMCR)	16	R/W	0001h	57.7.5/3648

57.7.1 Watchdog Control Register (WDOGx_WCR)

The Watchdog Control Register (WDOG_WCR) controls the WDOG operation.

- WDZST, WDBG and WDW are write-once only bits. Once the software does a write access to these bits, they will be locked and cannot be reprogrammed until the next system reset assertion.
- WDE is a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next system reset.
- WDT is also a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next POR. This bit does not get reset/cleared due to any system reset.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8
Read / Write	WT							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read / Write	WDW	SRE	WDA	SRS	WDT	WDE	WDBG	WDZST
Reset	0	0	1	1	0	0	0	0

WDOGx_WCR field descriptions

Field	Description
15–8 WT	Watchdog Time-out Field. This 8-bit field contains the time-out value that is loaded into the Watchdog counter after the service routine has been performed or after the Watchdog is enabled. After reset, WT[7:0] must have a value written to it before enabling the Watchdog otherwise count value of zero which is 0.5 seconds is loaded into the counter. **NOTE:** The time-out value can be written at any point of time but it is loaded to the counter at the time when WDOG is enabled or after the service routine has been performed. For more information see Timeout event . 0x00 - 0.5 Seconds (Default). 0x01 - 1.0 Seconds. 0x02 - 1.5 Seconds. 0x03 - 2.0 Seconds. 0xff - 128 Seconds.
7 WDW	Watchdog Disable for Wait. This bit determines the operation of WDOG during Low Power WAIT mode. This is a write once only bit. 0 Continue WDOG timer operation (Default). 1 Suspend WDOG timer operation.
6 SRE	software reset extension, an option way to generate software reset

Table continues on the next page...

WDOGx_WCR field descriptions (continued)

Field	Description
	adopt a new way to generate a more robust software reset. This bit can be set/clear with IP bus and will be reset with power-on reset . 0 using original way to generate software reset (default) 1 using new way to generate software reset.
5 WDA	WDOG_B assertion. Controls the software assertion of the WDOG_B signal. 0 Assert WDOG_B output. 1 No effect on system (Default).
4 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal WDOG_RESET_B_DEB . This bit automatically resets to "1" after it has been asserted to "0". **NOTE:** This bit does not generate the software reset to the block. 0 Assert system reset signal. 1 No effect on the system (Default).
3 WDT	WDOG_B Time-out assertion. Determines if the WDOG_B gets asserted upon a Watchdog Time-out Event. This is a write-one once only bit. **NOTE:** There is no effect on WDOG_RESET_B_DEB (WDOG Reset) upon writing on this bit. WDOG_B gets asserted along with WDOG_RESET_B_DEB if this bit is set. 0 No effect on WDOG_B (Default). 1 Assert WDOG_B upon a Watchdog Time-out event.
2 WDE	Watchdog Enable. Enables or disables the WDOG block. This is a write one once only bit. It is not possible to clear this bit by a software write, once the bit is set. **NOTE:** This bit can be set/reset in debug mode (exception). 0 Disable the Watchdog (Default). 1 Enable the Watchdog.
1 WDBG	Watchdog DEBUG Enable. Determines the operation of the WDOG during DEBUG mode. This bit is write once only. 0 Continue WDOG timer operation (Default). 1 Suspend the watchdog timer.
0 WDZST	Watchdog Low Power. Determines the operation of the WDOG during low-power modes. This bit is write once-only. **NOTE:** The WDOG can continue/suspend the timer operation in the low-power modes (STOP and DOZE mode). 0 Continue timer operation (Default). 1 Suspend the watchdog timer.

57.7.2 Watchdog Service Register (WDOGx_WSR)

When enabled, the WDOG requires that a service sequence be written to the Watchdog Service Register (WSR) to prevent the timeout condition.

NOTE

Executing the service sequence will reload the WDOG timeout counter.

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	WSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WDOGx_WSR field descriptions

Field	Description
WSR	Watchdog Service Register. This 16-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: 0x5555 Write to the Watchdog Service Register (WDOG_WSR). 0xAAAA Write to the Watchdog Service Register (WDOG_WSR).

57.7.3 Watchdog Reset Status Register (WDOGx_WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. Therefore, only one bit in the WRSR will always be asserted high. The register will always indicate the source of the last reset generated due to WDOG. Read access to this register is with one wait state. Any write performed on this register will generate a Peripheral Bus Error .

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Watchdog Time-out
- Software Reset

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			POR	0		TOUT	SFTW
Write								
Reset	0	0	0	0	0	0	0	0

WDOGx_WRSR field descriptions

Field	Description
15–5 Reserved	This read-only field is reserved and always has the value 0.
4 POR	Power On Reset. Indicates whether the reset is the result of a power on reset. 0 Reset is not the result of a power on reset. 1 Reset is the result of a power on reset.
3–2 Reserved	This read-only field is reserved and always has the value 0.
1 TOUT	Timeout. Indicates whether the reset is the result of a WDOG timeout. 0 Reset is not the result of a WDOG timeout. 1 Reset is the result of a WDOG timeout.
0 SFTW	Software Reset. Indicates whether the reset is the result of a WDOG software reset by asserting SRS bit 0 Reset is not the result of a software reset. 1 Reset is the result of a software reset.

57.7.4 Watchdog Interrupt Control Register (WDOGx_WICR)

The WDOG_WICR controls the WDOG interrupt generation.

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WIE	WTIS	0						WICT							
Write		w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

WDOGx_WICR field descriptions

Field	Description
15 WIE	Watchdog Timer Interrupt enable bit. Reset value is 0. **NOTE:** This bit is a write once only bit. Once the software does a write access to this bit, it will get locked and cannot be reprogrammed until the next system reset assertion 0 Disable Interrupt (Default). 1 Enable Interrupt.
14 WTIS	Watchdog TImer Interrupt Status bit will reflect the timer interrupt status, whether interrupt has occurred or not. Once the interrupt has been triggered software must clear this bit by writing 1 to it. 0 No interrupt has occurred (Default). 1 Interrupt has occurred
13–8 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

WDOGx_WICR field descriptions (continued)

Field	Description
WICT	Watchdog Interrupt Count Time-out (WICT) field determines, how long before the counter time-out must the interrupt occur. The reset value is 0x04 implies interrupt will occur 2 seconds before time-out. The maximum value that can be programmed to WICT field is 127.5 seconds with a resolution of 0.5 seconds. **NOTE:** This field is write once only. Once the software does a write access to this field, it will get locked and cannot be reprogrammed until the next system reset assertion. 0x00 WICT[7:0] = Time duration between interrupt and time-out is 0 seconds. 0x01 WICT[7:0] = Time duration between interrupt and time-out is 0.5 seconds. 0x04 WICT[7:0] = Time duration between interrupt and time-out is 2 seconds (Default). 0xff WICT[7:0] = Time duration between interrupt and time-out is 127.5 seconds.

57.7.5 Watchdog Miscellaneous Control Register (WDOGx_WMCR)

WDOG_WMCR Controls the Power Down counter operation.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0															PDE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

WDOGx_WMCR field descriptions

Field	Description
15–1 Reserved	This read-only field is reserved and always has the value 0.
0 PDE	Power Down Enable bit. Reset value of this bit is 1, which means the power down counter inside the WDOG is enabled after reset. The software must write 0 to this bit to disable the counter within 16 seconds of reset de-assertion. Once disabled this counter cannot be enabled again. See Power-down counter event for operation of this counter. **NOTE:** This bit is write-one once only bit. Once software sets this bit it cannot be reset until the next system reset. 0 Power Down Counter of WDOG is disabled. 1 Power Down Counter of WDOG is enabled (Default).

Chapter 58
Crystal Oscillator (XTALOSC)

58.1 Overview

This block comprises both the 24 MHz and 32 kHz implementation of a biased amplifier that when combined with a suitable external quartz crystal and external load capacitors, implements an oscillator.

The block includes means to:

- Accept an external clock source.
- Detect if the crystal frequency is close to 24 MHz or 32 kHz.
- Reduce the operating current via software after the oscillator has started (24 MHz specific feature)
- Supply another ~32 kHz clock source based off an independent internal oscillator if there is no oscillation sensed on the RTC_XTAL bumps(contacts) (32 kHz specific feature). The internal oscillator will provide clocks to the same on-chip modules as the external 32 kHz oscillator.
- Automatically switch to the external oscillation source when sensed on the RTC_XTAL bumps(contacts) (32 kHz specific feature).

58.2 External Signals

The table found here describes the external signals of XTALOSC:

Table 58-1. XTALOSC External Signals

Signal	Description	Pad	Mode	Direction
XTALOSC_REF_CLK_32K	32 kHz reference clock	ENET1_RX_EN	ALT2	O
		GPIO1_IO03	ALT3	
		JTAG_TCK	ALT6	

Table continues on the next page...

Table 58-1. XTALOSC External Signals (continued)

Signal	Description	Pad	Mode	Direction
XTALOSC_REF_CLK_24M	24 MHz reference clock	ENET1_TX_DATA0	ALT2	O
		ENET2_TX_DATA0	ALT8	
		GPIO1_IO04	ALT3	
		JTAG_TRST_B	ALT6	
XTALOSC_REF_CLK_25M	25 MHz reference clock	ENET2_RX_EN	ALT8	O
		GPIO1_IO02	ALT3	
		JTAG_MOD	ALT3	
XTALOSC_REF_CLK1	XTALOSC reference clock 1	ENET1_TX_CLK	ALT4	O
		GPIO1_IO00	ALT3	
		GPIO1_IO04	ALT0	
XTALOSC_REF_CLK2	XTALOSC reference clock 2	ENET2_TX_CLK	ALT4	O
		GPIO1_IO01	ALT3	
		GPIO1_IO05	ALT0	

58.3 Crystal Oscillator 24 MHz

58.3.1 Oscillator Configuration (24 MHz)

The basic block diagram of the 24 MHz module configured as a crystal oscillator is shown below.

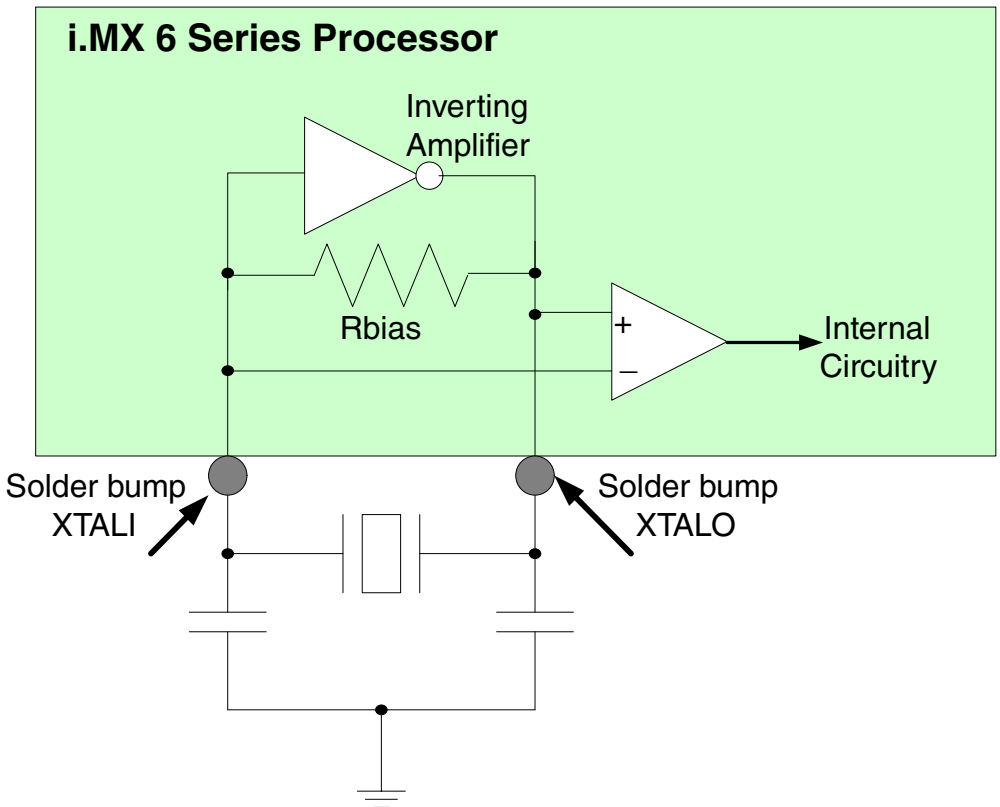


Figure 58-1. Oscillator Configuration (24 MHz)

This integrated biased amplifier can be used to create different frequency oscillators with different external component selection. However, care should be taken as many of the serial IO modules depend on the fixed frequency of 24 MHz. Please consult the sections of the document pertaining to the USB, ENET interfaces, for example. Once a healthy oscillation is established, then the bias current of the oscillator can generally be reduced to save power. This is accomplished through the XTALOSC24M_MISC0[OSC_I] bits, defined in the MISC0 register later in this chapter. Restore the XTALOSC24M_MISC0[OSC_I] bits before going into a power mode where the XTALOSC24 is powered down or oscillator startup may become an issue. The power down of the XTALOSC24 module is controlled by the CCM. See this section of the manual for more details.

58.3.2 RC Oscillator (24 MHz)

A lower-power RC oscillator module is available on-chip as a possible alternative to the 24 MHz crystal oscillator after a successful power-up sequence.

The 24 MHz RC oscillator is a self-tuning circuit that will output the programmed frequency value by using the RTC clock as it's reference. This oscillator is intended for normal operation and not fast boot.

While the power consumption of this RC oscillator is much lower than the 24 MHz crystal oscillator, one limitation of this RC oscillator module is that its clock frequency is not as accurate. Therefore, care should be observed when using this oscillator as the reference for the on-chip PLLs as their output clock frequency will be lower/higher than when using the 24 MHz crystal oscillator clock.

For more details on the possible usage of this module please contact a Freescale FAE for pertinent application-notes.

58.3.3 Crystal Frequency Detection(24 MHz)

A submodule exists that gives a fairly crude (relative to the accuracy of a crystal) estimation of whether the clock frequency is correct.

This function may be enabled by setting the XTALOSC24M_MISC0[OSC_XTALOK_EN] bit. It is disabled at system reset. When the oscillator is stable and the correct frequency is detected, the XTALOSC24M_MISC0[OSC_XTALOK] bit will be set. Note that the correct frequency will be observed before the oscillator fully blooms(the oscillation waveform build-up is completed).

58.4 Crystal Oscillator 32 kHz

58.4.1 Oscillator Configuration (32 kHz)

The basic block diagram of the 32 kHz module configured as a crystal oscillator is shown below.

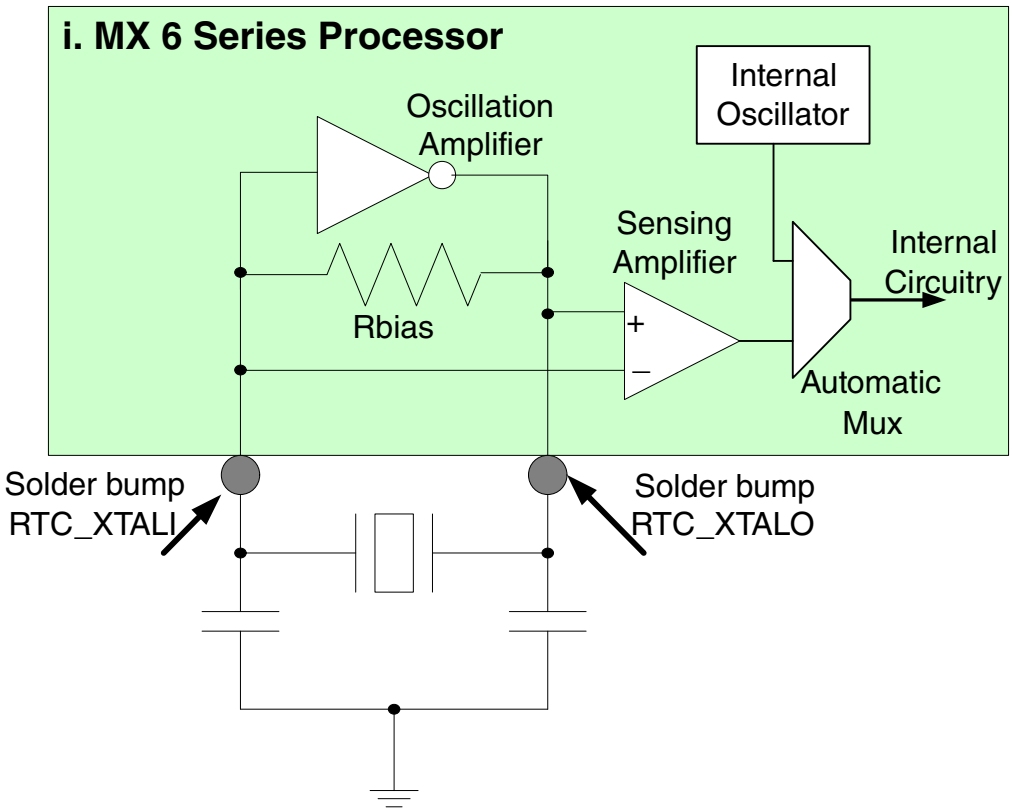


Figure 58-2. Oscillator Configuration (32 kHz)

This integrated biased amplifier can be used to create different frequency oscillators with different external component selection. Generally, RTC oscillators are either implemented with 32 kHz or 32.768 kHz crystals. Please consult the Security Reference Manual for appropriate frequency selection and configuration. Care must be taken to limit external leakage as this may debias the amplifier and degrade the gain.

The internal oscillator is automatically multiplexed in the clocking system when the system detects a loss of clock. The internal oscillator will provide clocks to the same on-chip modules as the external 32 kHz oscillator. The internal oscillator is not precise relative to a crystal. While it will provide a clock to the system, it generally will not be precise enough for long term time keeping. The internal oscillator is anticipated to be useful for quicker startup times, tampering prevention, and for cost savings in applications not needing the precision of an external crystal.

58.4.2 Bypass Configuration (32 kHz)

If it is desired to drive the chip with an external clock source, then the 32 kHz oscillator could be driven in one of three configurations using a nominal 1.1V source.

1. A single ended external clock source can be used to overdrive the output of the amplifier (RTC_XTALO). Since the oscillation sensing amplifier is differential, the RTC_XTALI pin should be externally floating and capacitively loaded. The combination of the internal biasing resistor and the external capacitor will filter the signal applied to the RTC_XTALO pin and develop a rough reference for the sensing amplifier to compare to.
2. A single ended external clock source can be used to drive RTC_XTALI. In this configuration, RTC_XTALO should be left externally floating.
3. A differential external clock source can be used to drive both RTC_XTALI and RTC_XTALO.

Generally, configuration 2 is anticipated to be the most used configuration, but all three configurations may be utilized.

58.5 XTALOSC 24MHz Memory Map/Register Definition

NOTE

The register content is mixed with analog functions not related to the oscillator function. These bits are noted.

XTALOSC24M memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8150	Miscellaneous Register 0 (XTALOSC24M_MISC0)	32	R/W	0400_0000h	58.5.1/3656
20C_8270	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL)	32	R/W	0000_4009h	58.5.2/3659
20C_8274	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_SET)	32	R/W	0000_4009h	58.5.2/3659
20C_8278	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_CLR)	32	R/W	0000_4009h	58.5.2/3659
20C_827C	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_TOG)	32	R/W	0000_4009h	58.5.2/3659
20C_82A0	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0)	32	R/W	0000_1020h	58.5.3/3662
20C_82A4	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_SET)	32	R/W	0000_1020h	58.5.3/3662
20C_82A8	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_CLR)	32	R/W	0000_1020h	58.5.3/3662
20C_82AC	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_TOG)	32	R/W	0000_1020h	58.5.3/3662

Table continues on the next page...

XTALOSC24M memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_82B0	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1)	32	R/W	0000_02EEh	58.5.4/3663
20C_82B4	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_SET)	32	R/W	0000_02EEh	58.5.4/3663
20C_82B8	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_CLR)	32	R/W	0000_02EEh	58.5.4/3663
20C_82BC	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_TOG)	32	R/W	0000_02EEh	58.5.4/3663
20C_82C0	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2)	32	R/W	0001_02E2h	58.5.5/3664
20C_82C4	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_SET)	32	R/W	0001_02E2h	58.5.5/3664
20C_82C8	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_CLR)	32	R/W	0001_02E2h	58.5.5/3664
20C_82CC	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_TOG)	32	R/W	0001_02E2h	58.5.5/3664

58.5.1 Miscellaneous Register 0 (XTALOSC24M_MISC0)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 20C_8000h base + 150h offset = 20C_8150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	VID_PLL_PREDIV	XTAL_24M_PWD	RTC_XTAL_SOURCE	CLKGATE_DELAY			CLKGATE_CTRL	Reserved								OSC_XTALOK_EN
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OSC_XTALOK	OSC_I		DISCON_HIGH_SNVS	STOP_ MODE_ CONFIG		Reserved		REFTOP_VBGUP	REFTOP_VBGADJ			REFTOP_SELFBIASOFF	Reserved		REFTOP_PWD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XTALOSC24M_MISC0 field descriptions

Field	Description
31 VID_PLL_ PREDIV	Predivider for the source clock of the PLL's. **NOTE:** Not related to oscillator. 0 Divide by 1 1 Divide by 2
30 XTAL_24M_PWD	This field powers down the 24M crystal oscillator if set true.
29 RTC_XTAL_ SOURCE	This field indicates which chip source is being used for the rtc clock. 0 Internal ring oscillator 1 RTC_XTAL
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. 000 0.5ms 001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms 101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–17 -	This field is reserved. Always set to zero.
16 OSC_XTALOK_ EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable.
15 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency.
14–13 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable. 00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%

Table continues on the next page...

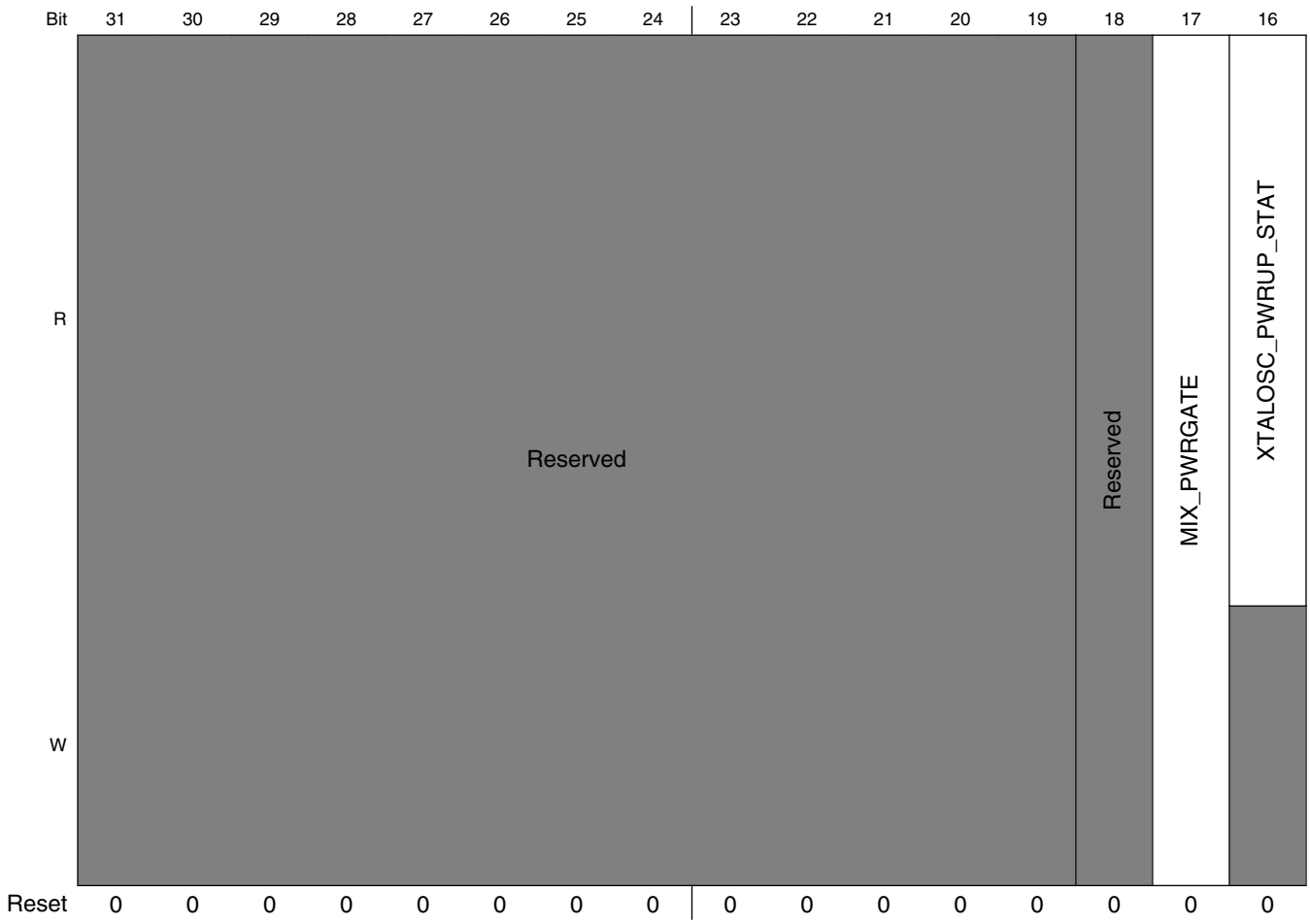
XTALOSC24M_MISC0 field descriptions (continued)

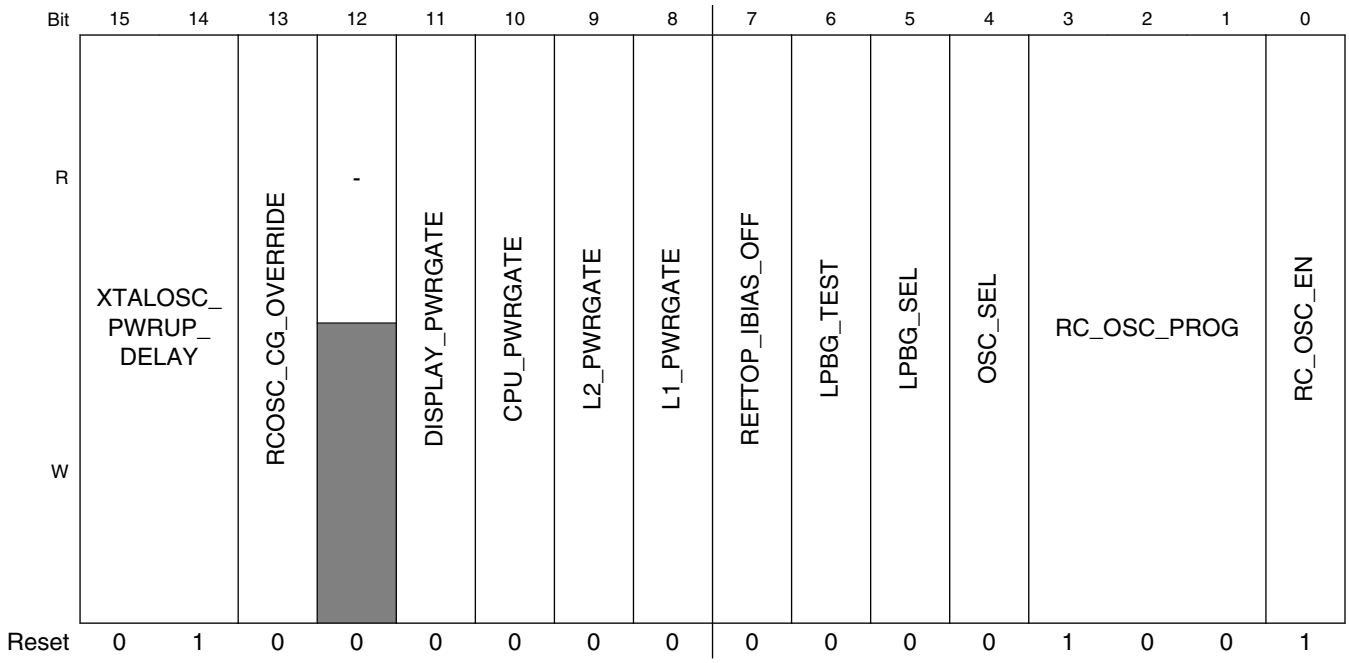
Field	Description
12 DISCON_HIGH_SNVS	This bit controls a switch from VDD_HIGH_IN to VDD_SNVS_IN. 0 Turn on the switch 1 Turn off the switch
11–10 STOP_MODE_CONFIG	Configure the analog behavior in stop mode. **NOTE:** Not related to oscillator. 00 All analog except rtc powered down on stop mode assertion. XtalOsc=on, RCOsc=off; 01 Certain analog functions such as certain regulators left up. XtalOsc=on, RCOsc=off; 10 XtalOsc=off, RCOsc=on, Old BG=on, New BG=off. 11 XtalOsc=off, RCOsc=on, Old BG=off, New BG=on.
9–8 -	This field is reserved. Reserved
7 REFTOP_VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable. **NOTE:** Not related to oscillator.
6–4 REFTOP_VBGADJ	**NOTE:** Not related to oscillator. 000 Nominal VBG 001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. **NOTE:** Not related to oscillator. 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2–1 -	This field is reserved.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, pll, and other analog functions on the die. **NOTE:** Not related to oscillator.

58.5.2 XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL*n*)

This register defines xtal osc and low power configuration.

Address: 20C_8000h base + 270h offset + (4d × i), where i=0d to 3d





XTALOSC24M_LOWPWR_CTRL*n* field descriptions

Field	Description
31–19 -	This field is reserved.
18 -	This field is reserved.
17 MIX_PWRGATE	Display power gate control. Used as software mask. Set to zero to force ungated.
16 XTALOSC_ PWRUP_STAT	Status of the 24MHz xtal oscillator. 0 Not stable 1 Stable and ready to use
15–14 XTALOSC_ PWRUP_DELAY	Specifies the time delay between when the 24MHz xtal is powered up until it is stable and ready to use. 00 0.25ms 01 0.5ms 10 1ms 11 2ms
13 RCOSC_CG_ OVERRIDE	For debug purposes only. This bit effects clock gating of certain digital logic clocked by the 24MHz clk.
12 -	Reserved
11 DISPLAY_ PWRGATE	Display logic power gate control. Used as software override. **NOTE:** Not related to oscillator.
10 CPU_PWRGATE	CPU power gate control. Used as software override.

Table continues on the next page...

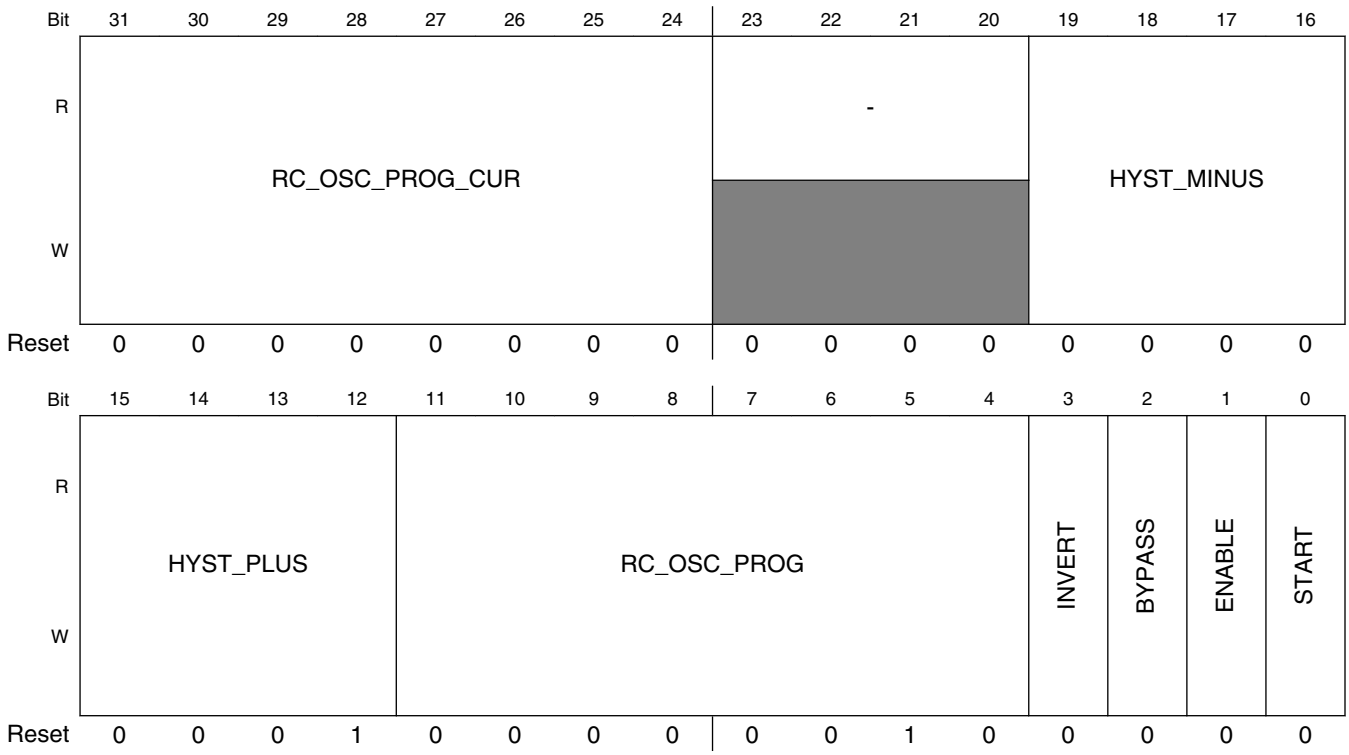
XTALOSC24M_LOWPWR_CTRL*n* field descriptions (continued)

Field	Description
	Attention: Test purpose only **NOTE:** Not related to oscillator.
9 L2_PWRGATE	L2 power gate control. Used as software override. **NOTE:** Not related to oscillator.
8 L1_PWRGATE	L1 power gate control. Used as software override. **NOTE:** Not related to oscillator.
7 REFTOP_IBIAS_OFF	Low power reftop ibias disable. **NOTE:** Not related to oscillator.
6 LPBG_TEST	Low power bandgap test bit. **NOTE:** Not related to oscillator.
5 LPBG_SEL	Bandgap select. **NOTE:** Not related to oscillator. 0 Normal power bandgap 1 Low power bandgap
4 OSC_SEL	Select the source for the 24MHz clock. 0 XTAL OSC 1 RC OSC
3–1 RC_OSC_PROG	RC osc. tuning values.
0 RC_OSC_EN	RC Osc. enable control. 0 Use XTAL OSC to source the 24MHz clock 1 Use RC OSC

58.5.3 XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 20C_8000h base + 2A0h offset + (4d × i), where i=0d to 3d



XTALOSC24M_OSC_CONFIG0*n* field descriptions

Field	Description
31–24 RC_OSC_ PROG_CUR	The current tuning value in use.
23–20 -	Reserved.
19–16 HYST_MINUS	Negative hysteresis value. Subtracted from target value before comparison. This, along with HYST_PLUS creates a range.
15–12 HYST_PLUS	Positive hysteresis value. Added to target value before comparison. This, along with HYST_MINUS creates a range.
11–4 RC_OSC_PROG	RC osc. tuning values.
3 INVERT	Invert the stepping of the calculated RC tuning value.

Table continues on the next page...

XTALOSC24M_OSC_CONFIG0*n* field descriptions (continued)

Field	Description
2 BYPASS	Bypasses any calculated RC tuning value and uses the programmed register value.
1 ENABLE	Enables the tuning logic to calculate new RC tuning values. Disabling essentially freezes the state of the calculation.
0 START	Start/stop bit for the RC tuning calculation logic. If stopped the tuning logic is reset.

58.5.4 XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 20C_8000h base + 2B0h offset + (4d × i), where i=0d to 3d

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

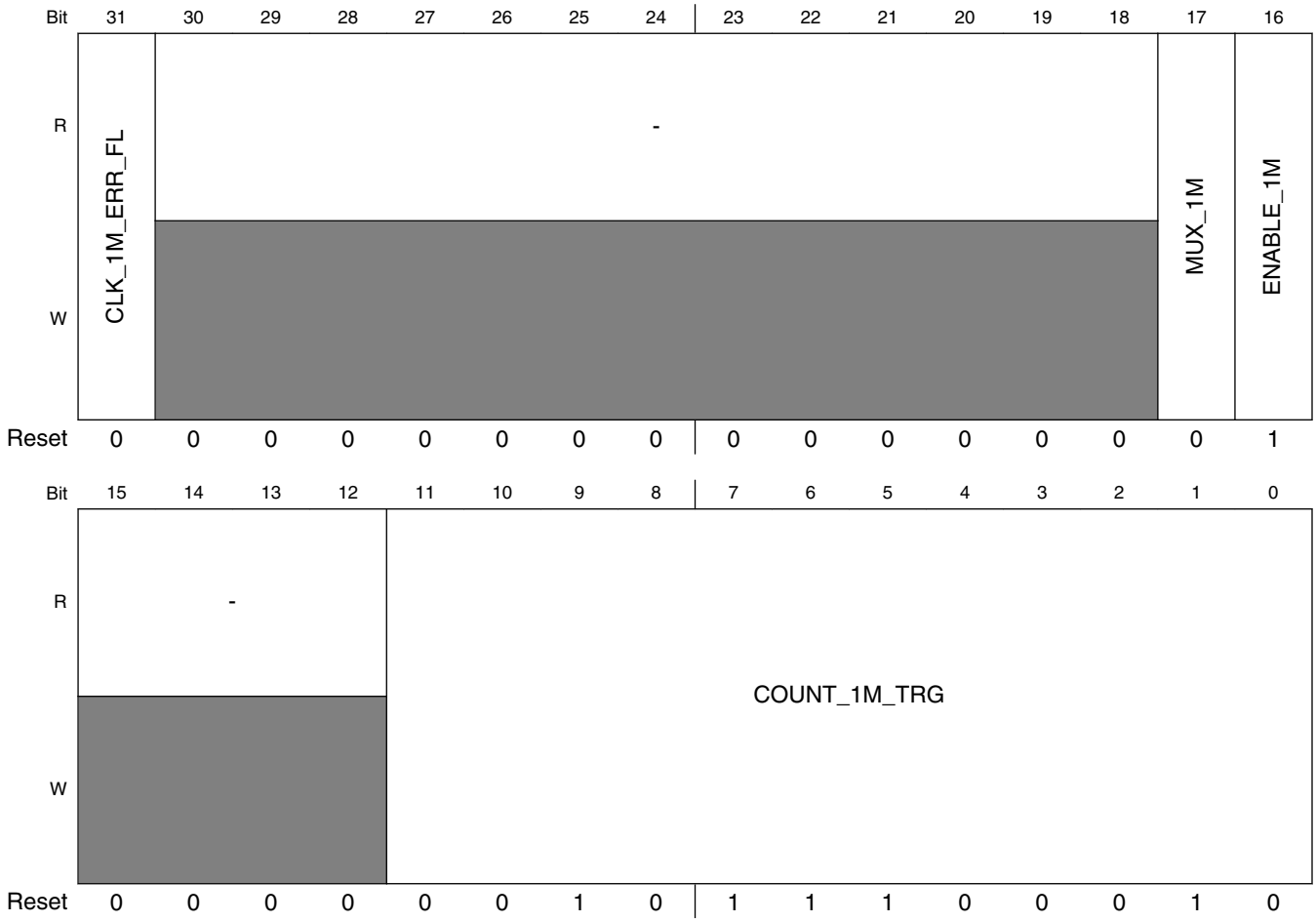
XTALOSC24M_OSC_CONFIG1*n* field descriptions

Field	Description
31–20 COUNT_RC_ CUR	The current tuning value in use.
19–12 -	Reserved
COUNT_RC_ TRG	The target count used to tune the RC OSC frequency. Essentially the number of desired RC Osc clock cycles within 1 32KHz clock cycle.

58.5.5 XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 20C_8000h base + 2C0h offset + (4d × i), where i=0d to 3d



XTALOSC24M_OSC_CONFIG2*n* field descriptions

Field	Description
31 CLK_1M_ERR_FL	Flag indicates that the count_1m count wasn't reached within 1 32KHz period. This is intended as feedback to software that the HW_ANADIG_OSC_CONFIG2_COUNT_1M_TRG value is too high for the RC Osc frequency.
30–18 -	Reserved.
17 MUX_1M	Mux the corrected or uncorrected 1MHz clock to the output.

Table continues on the next page...

XTALOSC24M_OSC_CONFIG2*n* field descriptions (continued)

Field	Description
16 ENABLE_1M	Enable the 1MHz clock output. 0 - disabled; 1 - enabled.
15–12 -	Reserved
COUNT_1M_ TRG	The target count used to tune the RC OSC frequency. Essentially the number of desired RC Osc clock cycles within 1 32KHz clock cycles.

Appendix A
i.MX 6UltraLite Revision History

A.1 Substantive changes from revision 0 to revision 1

Substantive changes from revision 0 to revision 1 are as follows:

A.1.1 Reference Manual Revision History

No substantive changes

A.1.2 Architecture Overview Revision History

Reference	Description
Features	Edited QuadSPI FLASH description and peak frequency of camera sensors.

A.1.3 Memory Maps Revision History

No substantive changes

A.1.4 Interrupts Revision History

No substantive changes

A.1.5 External Signals Revision History

No substantive changes

A.1.6 Fusemap Revision History

Reference	Description
Fusemap Descriptions Table	Added description for 460[16] - [18]

A.1.7 External Memory Controllers Revision History

No substantive changes

A.1.8 System Debug Revision History

Reference	Description
JTAG ID	Added new JTAG ID for i.MX 6UltraLite.

A.1.9 System Boot Revision History

Reference	Description
Persistent Bits	Updated "Persistent Bits" table.

A.1.10 Multimedia Revision History

Reference	Description
LCD Interface	Removed 32 bit LCD data bus.

A.1.11 Power Management Revision History

No substantive changes

A.1.12 System Security Revision History

No substantive changes

A.1.13 ARM A7 Revision History

No substantive changes

A.1.14 ADC Revision History

No substantive changes

A.1.15 AIPSTZ Revision History

No substantive changes

A.1.16 APBH Revision History

No substantive changes

A.1.17 ASRC Revision History

Reference	Description
Overview	Updated Figure "General System Overview."

A.1.18 BCH Revision History

No substantive changes

A.1.19 CCM Revision History

No substantive changes

A.1.20 CSI Revision History

No substantive changes

A.1.21 ECSPI Revision History

No substantive changes

A.1.22 EIM Revision History

Reference	Description
Features	Added 32 bit EIM and remvoed RDY_INT signal function.
Multiplexed Address/Data Mode	Updated EIM Multiplexing table.

A.1.23 ENET Revision History

No substantive changes

A.1.24 EPIT Revision History

No substantive changes

A.1.25 FLEXCAN3 Revision History

No substantive changes

A.1.26 GPC Revision History

Reference	Description
GPC	Removed the GPU/VPU and PCIE PHY bit from CNTR register.

A.1.27 GPIO Revision History

No substantive changes

A.1.28 GPMI Revision History

Reference	Description
Overview	Added ONFI 2.2 for 6UL.

A.1.29 GPT Revision History

No substantive changes

A.1.30 I2C Revision History

No substantive changes

A.1.31 IOMUXC Revision History

Reference	Description
Daisy chain - multi pads driving same module input pin	Added a note for the diagram.
IOMUXC_GPR	Updated bit field [0:3] of GPR3 register.

A.1.32 KPP Revision History

No substantive changes

A.1.33 eLCDIF Revision History

Reference	Description
Overview	Added 32 bits LCD data bus.

A.1.34 MMDC Revision History

Reference	Description
External Signals	Added DRAM_SDCLK0_P and DRAM_SDCLK0_N.
Address mirroring	Updated the note.
MMDC	Updated the MPZQLP2CTL field description.

A.1.35 MQS Revision History

No substantive changes

A.1.36 OCOTP Revision History

No substantive changes

A.1.37 OCRAM Revision History

No substantive changes

A.1.38 PMU Revision History

Reference	Description
PMU	Updated the Bit 18 of PMU_LOWPWR_CTRLn to reserved.

A.1.39 PWM Revision History

No substantive changes

A.1.40 PXP Revision History

Reference	Description
Top-level architecture	Updated description.

A.1.41 QSPI Revision History

No substantive changes

A.1.42 ROMCP Revision History

No substantive changes

A.1.43 SAI Revision History

No substantive changes

A.1.44 SDMA Revision History

No substantive changes

A.1.45 SIM Revision History

No substantive changes

A.1.46 SJC Revision History

No substantive changes

A.1.47 SNVS Revision History

No substantive changes

A.1.48 SPBA Revision History

No substantive changes

A.1.49 SPDIF Revision History

No substantive changes

A.1.50 SRC Revision History

No substantive changes

A.1.51 TEMPMON Revision History

No substantive changes

A.1.52 TZASC Revision History

No substantive changes

A.1.53 UART Revision History

Reference	Description
Features	Removed UART 8-wire function.

A.1.54 USB Revision History

No substantive changes

A.1.55 USB PHY Revision History

No substantive changes

A.1.56 USDHC Revision History

Reference	Description
Features	Added HS200 for 6UltraLite.

A.1.57 WDOG Revision History

No substantive changes

A.1.58 XTALOSC Revision History

No substantive changes

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. ARM, ARM Powered, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere.

© 2016 Freescale Semiconductor, Inc.

Document Number IMX6**UL**RM
Revision 1, 04/2016