MPC5606S Microcontroller Reference Manual

Supports MPC5602S, MPC5604S and MPC5606S

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: MPC5606SRM
Rev. 7
10/2012

Chapter 1
Overview

1.1 Introduction 41
1.2 MPC5606S family comparison 43
1.3 Block diagram 45
1.4 Chip-level features 45
1.5 Feature details 48
1.5.1 Low-power operation 48
1.5.2 e200z0h core processor 51
1.5.3 Crossbar switch (XBAR) 53
1.5.4 Enhanced Direct Memory Access (eDMA) 54
1.5.5 Inter-IC communications module (I2C) 54
1.5.6 Interrupt Controller (INTC) 55
1.5.7 QuadSPI serial flash controller 55
1.5.8 System Integration Unit (SIU) 56
1.5.9 Flash memory 56
1.5.10 SRAM 57
1.5.11 On-chip graphics SRAM 57
1.5.12 Memory Protection Unit (MPU) 57
1.5.13 Boot Assist Module (BAM) 57
1.5.14 Enhanced Modular Input/Output System (eMIOS) 58
1.5.15 Analog-to-Digital Converter (ADC) 59
1.5.16 Deserial Serial Peripheral Interface (DSPI) 60
1.5.17 FlexCAN 61
1.5.18 Serial communication interface module (LINFlex) 62
1.5.19 System clocks and clock generation modules 62
1.5.20 Periodic Interrupt Timer module (PIT) 63
1.5.21 Real Time Counter (RTC) 64
1.5.22 System Timer Module (STM) 64
1.5.23 Software Watchdog Timer (SWT) 64
1.5.24 Display Control Unit (DCU) 64
1.5.25 Parallel Data Interface (PDI) 65
1.5.26 Liquid Crystal Display (LCD) driver 66
1.5.27 Stepper Motor Controller (SMC) 67
1.5.28 Stepper Stall Detect (SSD) 67
1.5.29 Sound Generation Logic (SGL) 67
1.5.30 IEEE 1149.1 JTAG Controller (JTAGC) 68
1.5.31 Nexus Development Interface (NDI) 68
1.6 Developer environment 68
1.7 How to use the MPC5606S documents 69
1.7.1 The MPC5606S document set 69
1.7.2 Reference manual content 69
1.8 Using the MPC5606S 71
1.8.1 Hardware design 71
1.8.2 Input/output pins 72

1.8.3 Software design 72
1.8.4 Other features 73

Chapter 2
Memory Map

Chapter 3
Signal Description

3.1 Introduction 81
3.2 Package pinouts 82
3.3 Pad configuration during reset phases 85
3.4 Voltage supply pins 85
3.5 Pad types 86
3.6 System pins 87
3.7 Debug pins 87
3.8 Functional ports 88
3.9 Signal details 107

Chapter 4
Safety

4.1 Register protection 111
4.1.1 Introduction 111
4.1.1.1 Overview 111
4.1.1.2 Features 111
4.1.1.3 Modes of operation 112
4.1.2 External signal description 112
4.1.3 Memory map and register description 112
4.1.3.1 Memory map 113
4.1.3.2 Register description 114
4.1.4 Functional description 116
4.1.4.1 General 116
4.1.4.2 Change lock settings 116
4.1.4.3 Access errors 119
4.1.5 Reset 119
4.2 Software Watchdog Timer (SWT) 119
4.2.1 Overview 119
4.2.2 Features 120
4.2.3 Modes of operation 120
4.2.4 External signal description 121
4.2.5 Memory map and register description 121
4.2.5.1 Memory map 122
4.2.5.2 SWT Control Register (SWT_CR) 122
4.2.5.3 SWT Interrupt Register (SWT_IR) 123
4.2.5.4 SWT Timeout Register (SWT_TO) 124

4.2.5.5 SWT Window Register (SWT_WN) 125
4.2.5.6 SWT Service Register (SWT_SR) 125
4.2.5.7 SWT Counter Output Register (SWT_CO) 126
4.2.5.8 SWT Service Key Register (SWT_SK) 126
4.2.6 Functional description 127

Chapter 5
Analog-to-Digital Converter (ADC)

5.1 Overview 129
5.1.1 Device-specific features 129
5.1.2 Device-specific implementation 130
5.2 Introduction 130
5.3 Functional description 131
5.3.1 Analog channel conversion 131
5.3.1.1 Normal conversion 131
5.3.1.2 Start of normal conversion 131
5.3.1.3 Normal conversion operating modes 132
5.3.1.4 Injected channel conversion 133
5.3.1.5 Abort conversion 134
5.3.2 Analog clock generator and conversion timings 135
5.3.3 ADC sampling and conversion timing 135
5.3.4 Programmable analog watchdog 137
5.3.4.1 Introduction 137
5.3.5 DMA functionality 138
5.3.6 Interrupts 138
5.3.7 External decode signals delay 139
5.3.8 Power-down mode 139
5.3.9 Auto-clock-off mode 139
5.4 Register descriptions 140
5.4.1 Introduction 140
5.4.2 Control logic registers 143
5.4.2.1 Main Configuration Register (MCR) 143
5.4.2.2 Main Status Register (MSR) 145
5.4.3 Interrupt registers 147
5.4.3.1 Interrupt Status Register (ISR) 147
5.4.3.2 Channel Pending Registers (CEOCFR[1..2]) 148
5.4.3.3 Interrupt Mask Register (IMR) 149
5.4.3.4 Channel Interrupt Mask Register (CIMR[1..2]) 150
5.4.3.5 Watchdog Threshold Interrupt Status Register (WTISR) 151
5.4.3.6 Watchdog Threshold Interrupt Mask Register (WTIMR) 152
5.4.4 DMA registers 153
5.4.4.1 DMA Enable Register (DMAE) 153
5.4.4.2 DMA Channel Select Register (DMAR[1..2]) 154
5.4.5 Threshold registers 155
5.4.5.1 Introduction 155

5.4.5.2 Threshold Control Register (TRC*x*, *x* = [0..3]) 155
5.4.5.3 Threshold Register (THRHLR[0:3]) 156
5.4.6 Conversion timing registers CTR[1..2] 157
5.4.7 Mask registers 158
5.4.7.1 Introduction 158
5.4.7.2 Normal Conversion Mask Registers (NCMR[1..2]) 158
5.4.7.3 Injected Conversion Mask Registers (JCMR[1..2]) 159
5.4.8 Delay registers 160
5.4.8.1 Decode Signals Delay Register (DSDR) 160
5.4.8.2 Power-down Exit Delay Register (PDEDR) 160
5.4.9 Data registers 162
5.4.9.1 Introduction 162
5.4.9.2 Channel Data Register (CDR[0..95]) 162

Chapter 6
Boot Assist Module (BAM)

6.1 Overview 165
6.2 Features 165
6.3 Boot modes 165
6.4 Memory map 166
6.5 Functional description 166
6.5.1 Entering boot modes 166
6.5.2 Reset Configuration Half Word Source (RCHW) 168
6.5.3 Single-chip boot mode 169
6.5.3.1 Boot and alternate boot 170
6.5.4 Boot through BAM 170
6.5.4.1 Executing BAM 170
6.5.4.2 BAM software flow 171
6.5.4.3 BAM resources 172
6.5.4.4 Download and execute the new code 173
6.5.4.5 Download 64-bit password and password check 173
6.5.4.6 Download start address, VLE bit and code size 174
6.5.4.7 Download data 175
6.5.4.8 Execute code 175
6.5.5 Boot from UART 176
6.5.5.1 Configuration 176
6.5.5.2 Protocol 176
6.5.6 Bootstrap with CAN 177
6.5.6.1 Configuration 177
6.6 Protocol 178
6.6.1 Flash memory password swapping 178
6.6.2 Interrupts 179

Chapter 7
CAN Sampler

7.1 Introduction 181
7.2 Main features 182
7.3 Register description 182
7.3.1 CAN Sampler Control Register 182
7.3.2 CAN Sampler Sample Registers 0–11 183
7.4 Functional description 184
7.4.1 Enabling/disabling the CAN Sampler 185
7.4.2 Selecting the Rx port 185
7.4.3 Baud rate generation 186

Chapter 8
Clock Description

8.1 Clock architecture 187
8.2 Auxiliary clocks 188
8.3 Clock gating 189
8.4 Clock Generation Module (MC_CGM) 190
8.4.1 Introduction 190
8.4.1.1 Overview 190
8.4.1.2 Features 191
8.4.1.3 Modes of operation 192
8.4.2 External signal description 192
8.4.3 Memory map and register definition 192
8.4.3.1 Register descriptions 196
8.4.4 Functional description 204
8.4.4.1 System clock generation 204
8.4.4.2 Auxiliary clock generation 204
8.4.4.3 Output clock multiplexing 208
8.4.4.4 Output Clock Division Selection 208
8.5 FXOSC external oscillator 209
8.5.1 Main features 209
8.5.2 Functional description 209
8.5.3 Register description 210
8.6 32 KHz OSC digital interface 211
8.6.1 Introduction 211
8.6.2 Main features 211
8.6.3 Functional description 211
8.6.4 Register description 212
8.7 SIRC digital interface 213
8.7.1 Introduction 213
8.7.2 Low-Power RC Oscillator (128 kHz) 213
8.7.3 Register description 214
8.8 FIRC digital interface 214

8.8.1 Introduction 214
8.8.2 Functional description (16 MHz) 214
8.8.3 Register description 215
8.9 Frequency-modulated phase locked loops and system clocks (FMPLL0 and FMPLL1) 215
8.9.1 Introduction 215
8.9.2 Overview 216
8.9.3 Features 216
8.9.4 Memory map 217
8.9.5 Register description 217
8.9.5.1 Control Register (CR) 217
8.9.5.2 Modulation Register (MR) 220
8.9.6 Functional description 221
8.9.6.1 Normal mode 221
8.9.6.2 Progressive clock switching 221
8.9.6.3 Normal mode with frequency modulation 222
8.9.6.4 Powerdown mode 223
8.9.6.5 1:1 mode (FMPLL0 only) 223
8.9.7 Recommendations 223
8.10 Clock Monitor Unit (CMU) 224
8.10.1 Introduction 224
8.10.2 Main features 224
8.10.3 Block diagram 225
8.10.4 Functional description 225
8.10.4.1 Crystal clock monitor 226
8.10.4.2 PLL clock monitor 226
8.10.4.3 Frequency meter 226
8.10.5 Memory map and register description 227
8.10.5.1 Control Status Register (CMU_CSR) 228
8.10.5.2 Frequency Display Register (CMU_FDR) 229
8.10.5.3 High Frequency Reference Register FMPLL0 (CMU_HFREFR) 229
8.10.5.4 Low Frequency Reference Register FMPLL0 (CMU_LFREFR) 230
8.10.5.5 Interrupt Status Register (CMU_ISR) 230
8.10.5.6 Measurement Duration Register (CMU_MDR) 231

Chapter 9
Configurable Enhanced Modular IO Subsystem (eMIOS200)

9.1 Device-specific information 233
9.1.1 Unsupported features 233
9.1.2 Device-specific configuration 233
9.1.3 eMIOS clocking configuration 234
9.1.4 MPC5606S family comparison 234
9.1.5 Channel Types 234
9.1.6 Unified Channel block 236
9.1.6.1 Channel mode selection 236
9.2 Introduction 237

9.2.1 Overview 239
9.2.2 Features 239
9.2.3 Modes of operation 239
9.3 External signal description 240
9.3.1 Overview 240
9.3.2 Detailed signal descriptions 240
9.3.2.1 emiosi[*n*] — eMIOS200 Channel Input Signal 240
9.3.2.2 emioso[*n*] — eMIOS200 Channel Output Signal 240
9.3.2.3 emios_flag_out[*n*] — eMIOS200 Channel Flag Signal 240
9.4 Memory map and register description 240
9.4.1 Memory map 240
9.4.1.1 Unified Channel memory map 241
9.4.2 Register description 242
9.4.2.1 eMIOS200 Module Configuration Register (EMIOSMCR) 242
9.4.2.2 eMIOS200 Global FLAG Register (EMIOSGFLAG) 243
9.4.2.3 eMIOS200 Output Update Disable (EMIOSOUDIS) 244
9.4.2.4 eMIOS200 Disable Channel (EMIOSUCDIS) 245
9.4.2.5 eMIOS200 UC A Register (EMIOSA[*n*]) 246
9.4.2.6 eMIOS200 UC B Register (EMIOSB[*n*]) 247
9.4.2.7 eMIOS200 UC Counter Register (EMIOSCNT[*n*]) 248
9.4.2.8 eMIOS200 UC Control Register (EMIOSC[*n*]) 248
9.4.2.9 eMIOS200 UC Status Register (EMIOSS[*n*]) 253
9.4.2.10 eMIOS200 UC Alternate A Register (EMIOSALTA[*n*]) 253
9.5 Functional description 254
9.5.1 Unified Channel (UC) 256
9.5.1.1 UC modes of operation 257
9.5.1.2 Input Programmable Filter (IPF) 271
9.5.1.3 Clock Prescaler (CP) 272
9.5.1.4 Effect of Freeze on the Unified Channel 273
9.5.2 IP Bus Interface Unit (BIU) 273
9.5.2.1 Effect of Freeze on the BIU 273
9.5.3 Global Clock Prescaler Submodule (GCP) 273
9.5.3.1 Effect of Freeze on the GCP 274
9.6 Initialization/application information 274
9.6.1 Considerations 274
9.6.2 Application information 274
9.6.2.1 Time Base Generation 274
9.6.2.2 Coherent accesses 276
9.6.2.3 Channel/modes initialization 276

Chapter 10
Crossbar Switch (XBAR)

10.1 Introduction 279
10.2 Block diagram 279
10.3 Overview 279

10.4 Features 280
10.5 Modes of operation 280
10.5.1 Normal mode 280
10.5.2 Debug mode 280
10.6 Functional description 280
10.6.1 Overview 280
10.6.2 General operation 281
10.6.3 Master ports 281
10.6.4 Slave ports 282
10.6.5 Priority assignment 282
10.6.6 Arbitration 282
10.6.6.1 Fixed priority operation 282

Chapter 11
Deserial Serial Peripheral Interface (DSPI)

11.1 Introduction 285
11.2 Block diagram 285
11.3 Overview 286
11.4 Features 286
11.5 Modes of operation 287
11.5.1 Master mode 287
11.5.2 Slave mode 288
11.5.3 Module Disable mode 288
11.5.4 External Stop mode 288
11.5.5 Debug mode 288
11.6 External signal description 288
11.6.1 Signal overview 288
11.6.2 Signal names and descriptions 289
11.6.2.1 Peripheral Chip Select / Slave Select (CS_0) 289
11.6.2.2 Peripheral Chip Selects 1–2 (CS1:2) 289
11.6.2.3 Serial Input (SIN_x) 289
11.6.2.4 Serial Output (SOUT_x) 289
11.6.2.5 Serial Clock (SCK_*x*) 289
11.7 Memory map and register description 290
11.7.1 Memory map 290
11.7.2 Register description 291
11.7.2.1 DSPI Module Configuration Register (DSPIx_MCR) 291
11.7.2.2 DSPI Transfer Count Register (DSPIx_TCR) 293
11.7.2.3 DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn) . 293
11.7.2.4 DSPI Status Register (DSPIx_SR) 299
11.7.2.5 DSPI DMA / Interrupt Request Select and Enable Register (DSPIx_RSER) 301
11.7.2.6 DSPI PUSH TX FIFO Register (DSPIx_PUSHR) 303
11.7.2.7 DSPI POP RX FIFO Register (DSPIx_POPR) 305
11.7.2.8 DSPI Transmit FIFO Registers 0–4 (DSPIx_TXFRn) 305

11.7.2.9 DSPI Receive FIFO Registers 0–4 (DSPIx_RXFRn) 306
11.8 Functional description 307
11.8.1 Modes of operation 308
11.8.1.1 Master mode 308
11.8.1.2 Slave mode 309
11.8.1.3 Module Disable mode 309
11.8.1.4 External Stop mode 309
11.8.1.5 Debug mode 309
11.8.2 Start and stop of DSPI transfers 309
11.8.3 Serial Peripheral Interface (SPI) configuration 310
11.8.3.1 SPI Master mode 311
11.8.3.2 SPI Slave mode 311
11.8.3.3 FIFO disable operation 311
11.8.3.4 transmit First In First Out (TX FIFO) buffering mechanism 311
11.8.3.5 Receive First In First Out (RX FIFO) buffering mechanism 312
11.8.4 DSPI baud rate and clock delay generation 313
11.8.4.1 Baud rate generator 314
11.8.4.2 CS to SCK delay (tCSC) 314
11.8.4.3 After SCK delay (tASC) 314
11.8.4.4 Delay after transfer (tDT) 315
11.8.5 Transfer formats 316
11.8.5.1 Classic SPI transfer format (CPHA = 0) 318
11.8.5.2 Classic SPI transfer format (CPHA = 1) 319
11.8.5.3 Modified SPI transfer format (MTFE = 1, CPHA = 0) 320
11.8.5.4 Modified SPI transfer format (MTFE = 1, CPHA = 1) 321
11.8.5.5 Continuous selection format 322
11.8.5.6 Clock polarity switching between DSPI transfers 323
11.8.6 Continuous serial communications clock 324
11.8.7 Interrupts/DMA requests 326
11.8.7.1 End of queue interrupt request (EOQF) 326
11.8.7.2 Transmit FIFO fill interrupt or DMA request (TFFF) 326
11.8.7.3 Transfer complete interrupt request (TCF) 326
11.8.7.4 Transmit FIFO underflow interrupt request (TFUF) 327
11.8.7.5 Receive FIFO drain interrupt or DMA request (RFDF) 327
11.8.7.6 Receive FIFO overflow interrupt request (RFOF) 327
11.8.7.7 FIFO overrun request (TFUF) or (RFOF) 327
11.8.8 Power-saving features 327
11.8.8.1 External Stop mode 327
11.8.8.2 Module Disable mode 328
11.8.8.3 Slave interface signal gating 328
11.9 Initialization and application information 328
11.9.1 How to change queues 328
11.9.2 Baud rate settings 329
11.9.3 Delay settings 330
11.9.4 Calculation of FIFO pointer addresses 330

11.9.4.1 Address calculation for the first-in entry and last-in entry in the TX FIFO 331
11.9.4.2 Address calculation for the first-in entry and last-in entry in the RX FIFO 331

Chapter 12
Display Control Unit (DCU)

12.1 Introduction 333
12.1.1 Overview 334
12.1.2 Features 335
12.1.3 Modes of operation 336
12.2 External signal description 336
12.2.1 Overview 336
12.2.2 Detailed signal descriptions 337
12.3 Memory map and register definition 338
12.3.1 Memory map 338
12.3.2 Register map 338
12.3.3 Register summary 344
12.3.4 Register descriptions 353
12.3.4.1 Control Descriptor L0_1 Register (CtrlDescL0_1) 353
12.3.4.2 Control Descriptor L0_2 Register 354
12.3.4.3 Control Descriptor L0_3 Register 355
12.3.4.4 Control Descriptor L0_4 Register 356
12.3.4.5 Control Descriptor L0_5 Register 358
12.3.4.6 Control Descriptor L0_6 Register 359
12.3.4.7 Control Descriptor L0_7 Register 361
12.3.4.8 Control Descriptor Cursor 1 Register (CtrlDescCursor_1) 361
12.3.4.9 Control Descriptor Cursor 2 Register (CtrlDescCursor_2) 362
12.3.4.10 Control Descriptor Cursor 3 Register (CtrlDescCursor_3) 363
12.3.4.11 Control Descriptor Cursor 4 Register (CtrlDescCursor_4) 363
12.3.4.12 DCU Mode Register (DCU_MODE) 364
12.3.4.13 BGND Register 366
12.3.4.14 DISP_SIZE Register 367
12.3.4.15 HSYN_PARA Register 368
12.3.4.16 VSYN_PARA Register 368
12.3.4.17 SYN_POL Register 369
12.3.4.18 Threshold Register 370
12.3.4.19 Interrupt Status Register (INT_STATUS) 371
12.3.4.20 Interrupt Mask Register (INT_MASK) 373
12.3.4.21 COLBAR Registers 375
12.3.4.22 Divide Ratio (DIV_RATIO) register 379
12.3.4.23 SIGN_CALC_1 Register 380
12.3.4.24 SIGN_CALC_2 Register 381
12.3.4.25 CRC_VAL Register 381
12.3.4.26 PDI Status Register 382

12.3.4.27 PDI Status Mask Register 383
12.3.4.28 Parameter Error Status (PARR_ERR) register 384
12.3.4.29 Mask PARR_ERR Status register 387
12.3.4.30 THRESHOLD_INP_BUF_1 Register 389
12.3.4.31 THRESHOLD_INP_BUF_2 Register 389
12.3.4.32 LUMA Component Register 390
12.3.4.33 Red Chroma Components 391
12.3.4.34 Green Chroma Component Register 391
12.3.4.35 Blue Chroma Component Register 392
12.3.4.36 CRC_POS Register 393
12.3.4.37 FG0_FCOLOR Register 393
12.3.4.38 FG0_bcolor 394
12.3.4.39 Global Protection Register 395
12.3.4.40 Soft Lock Bit Register L0 396
12.3.4.41 Soft Lock Bit Register L1 397
12.3.4.42 Soft Lock DISP_SIZE Register 399
12.3.4.43 Soft Lock HSYNC/VSYNC PARA Register 400
12.3.4.44 Soft Lock POL Register 401
12.3.4.45 Soft Lock L0_TRANSP Register 401
12.3.4.46 Soft Lock L1_TRANSP Register 402
12.4 Functional description 403
12.4.1 Graphic sources 403
12.4.2 TFT LCD panel configuration 404
12.4.3 DCU mode selection and background color 406
12.4.4 Proper sequence for enabling and disabling the DCU 406
12.4.5 Layer configuration and blending 407
12.4.5.1 Blending priority of layers 407
12.4.5.2 Control Descriptors 410
12.4.5.3 Layer size and positioning 410
12.4.5.4 Graphics and data format 411
12.4.5.5 Alpha and Chroma-key blending 413
12.4.5.6 Transparency mode and blending 420
12.4.5.7 Luminance mode 423
12.4.5.8 Tile mode 423
12.4.6 Hardware cursor 424
12.4.7 CLUT/Tile RAM 426
12.4.8 Gamma correction 427
12.5 Timing, error and interrupt management 427
12.5.1 Synchronizing to panel frame rate 428
12.5.2 Managing the DCU FIFOs and DMA activity 428
12.5.3 Error detection 430
12.5.4 Interrupt generation 430
12.6 Register protection 431
12.6.1 Operation of scheme 431
12.6.2 List of protected registers 432

12.7 Safety mode 432
12.7.1 CRC area description 433
12.7.1.1 Relationship between various input signals 433
12.7.2 Features 435
12.7.3 Programming for Debug mode 436
12.7.4 Programming of Tag mode 436
12.8 Parallel Data Interface (Camera Interface) 436
12.8.1 ITU-R BT.656 sync information extraction 437
12.8.2 PDI interface description 438
12.8.2.1 Introduction 438
12.8.2.2 PDI interaction with other modules 439
12.8.2.3 Features 440
12.8.2.4 Normal and Narrow mode 440
12.8.2.5 Modes of operation based on sync extraction 442
12.8.2.6 Mode of operation depending on PDI_datain 446
12.8.2.7 PDI-related interrupts 447
12.9 DCU initialization 447
12.10Glossary 448

Chapter 13
DMA Channel Mux (DMACHMUX)

13.1 Introduction 449
13.1.1 Overview 449
13.1.2 Features 449
13.1.3 Modes of operation 450
13.2 External signal description 450
13.2.1 Overview 450
13.3 Memory map and register definition 450
13.3.1 Register descriptions 451
13.3.1.1 Channel configuration registers 451
13.4 Functional description 454
13.4.1 DMA channels with periodic triggering capability 454
13.4.2 DMA channels with no triggering capability 456
13.4.3 Always-enabled DMA sources 456
13.5 Initialization/application information 457
13.5.1 Reset 457
13.5.2 Enabling and configuring sources 457
13.5.2.1 Enabling a source with periodic triggering 457
13.5.2.2 Enabling a source without periodic triggering 458
13.5.2.3 Disabling a source 459
13.5.2.4 Switching the source of a DMA channel 459

Chapter 14
e200z0h Core

14.1 Overview 461
14.2 Features 461
14.2.1 Microarchitecture summary 462
14.2.1.1 Block diagram 463
14.2.1.2 Instruction unit features 463
14.2.1.3 Integer unit features 464
14.2.1.4 Load/store unit features 464
14.2.1.5 e200z0h system bus features 464
14.3 Core registers and programmer's model 464
14.3.1 Unimplemented SPRs and Read-only SPRs 467
14.4 Instruction summary 467

Chapter 15
Enhanced Direct Memory Access (eDMA)

15.1 Information specific to this device 469
15.1.1 Device-specific features 469
15.2 Introduction 469
15.2.1 Overview 470
15.2.2 Features 471
15.3 Memory map/register definition 476
15.3.1 Register descriptions 478
15.3.1.1 DMA Control Register (DMACR) register 478
15.3.1.2 DMA Error Status (DMAES) register 481
15.3.1.3 DMA Enable Request (DMAERQH, DMAERQL) registers 483
15.3.1.4 DMA Enable Error Interrupt (DMAEEIH, DMAEEIL) registers 484
15.3.1.5 DMA Set Enable Request (DMASERQ) register 485
15.3.1.6 DMA Clear Enable Request (DMACERQ) register 486
15.3.1.7 DMA Set Enable Error Interrupt (DMASEEI) register 486
15.3.1.8 DMA Clear Enable Error Interrupt (DMACEEI) register 487
15.3.1.9 DMA Clear Interrupt Request (DMACINT) register 488
15.3.1.10 DMA Clear Error (DMACERR) register 488
15.3.1.11 DMA Set START Bit (DMASSRT) register 489
15.3.1.12 DMA Clear DONE Status (DMACDNE) register 489
15.3.1.13 DMA Interrupt Request (DMAINTH, DMAINTL) registers 490
15.3.1.14 DMA Error (DMAERRH, DMAERRL) registers 491
15.3.1.15 DMA Hardware Request Status (DMAHRSH, DMAHRSL) registers 492
15.3.1.16 DMA General Purpose Output Register (DMAGPOR) register 493
15.3.1.17 DMA Channel n Priority (DCHPRIn), n = 0,..., {15,31,63} registers . 494
15.3.1.18 Transfer Control Descriptor (TCD) 495
15.4 Functional description 505
15.4.1 DMA microarchitecture 505
15.4.2 DMA basic data flow 506

15.4.3 DMA performance 509
15.5 Initialization/application information 512
15.5.1 DMA initialization 512
15.5.2 DMA programming errors 513
15.5.3 DMA arbitration mode considerations 514
15.5.3.1 Fixed group arbitration, fixed channel arbitration 514
15.5.3.2 Round-robin group arbitration, fixed channel arbitration 514
15.5.3.3 Round-robin group arbitration, round-robin channel arbitration 514
15.5.3.4 Fixed group arbitration, round-robin channel arbitration 515
15.5.4 DMA transfer 515
15.5.4.1 Single request 515
15.5.4.2 Multiple requests 516
15.5.5 TCD status 518
15.5.5.1 Minor loop complete 518
15.5.5.2 Active channel TCD reads 518
15.5.5.3 Preemption status 519
15.5.6 Channel linking 519
15.5.7 Dynamic programming 520
15.5.7.1 Dynamic priority changing 520
15.5.7.2 Dynamic channel linking and dynamic scatter/gather 520
15.5.8 Hardware request release timing 521

Chapter 16
Error Correction Status Module (ECSM)

16.1 Introduction 523
16.2 Overview 523
16.3 Features 523
16.4 Memory map and register description 523
16.4.1 Memory map 523
16.4.2 Register description 524
16.4.2.1 Processor Core Type (PCT) register 525
16.4.2.2 Revision (REV) register 525
16.4.2.3 Miscellaneous Reset Status Register (MRSR) 525
16.4.2.4 Miscellaneous Wakeup Control Register (MWCR) 526
16.4.2.5 Miscellaneous Interrupt Register (MIR) 527
16.4.2.6 Miscellaneous User-Defined Control Register (MUDCR) 528
16.4.2.7 ECC registers 529
16.4.2.8 ECC Configuration Register (ECR) 529
16.4.2.9 ECC Status Register (ESR) 530
16.4.2.10 ECC Error Generation Register (EEGR) 532
16.4.2.11 Flash ECC Address Register (FEAR) 535
16.4.2.12 Flash ECC Master Number Register (FEMR) 536
16.4.2.13 Flash ECC Attributes (FEAT) register 536
16.4.2.14 Flash ECC Data Register (FEDR) 537
16.4.2.15 RAM ECC Address Register (REAR) 538

16.4.2.16 RAM ECC Syndrome Register (RESR) 538
16.4.2.17 RAM ECC Master Number Register (REMR) 540
16.4.2.18 RAM ECC Attributes (REAT) register 540
16.4.2.19 RAM ECC Data Register (REDR) 541
16.4.3 High-priority enables 542
16.4.4 Spp_ips_reg_protection 542

Chapter 17
Flash Memory

17.1 Introduction 545
17.2 Program flash memory (code flash 0 and code flash 1) 545
17.2.1 Introduction 545
17.2.2 Main features 546
17.2.3 Block diagram 546
17.2.4 Functional description 547
17.2.4.1 Macrocell structure 547
17.2.4.2 Flash module sectorization 548
17.2.5 User mode operation 551
17.2.5.1 Reset 552
17.2.5.2 Power-Down mode 553
17.2.5.3 Low-Power mode 553
17.2.6 Register description 554
17.2.6.1 Module Configuration Register (MCR) 555
17.2.6.2 Low/Mid Address Space Block Locking Register (LML) 560
17.2.6.3 Non-Volatile Low/Mid Address Space Block Locking Register (NVLML) . 561
17.2.6.4 High Address Space Block Locking Register (HBL) 563
17.2.6.5 Non-Volatile High Address Space Block Locking Register (NVHBL) . 563
17.2.6.6 Secondary Low/Mid Address Space Block Locking Register (SLL) . . 564
17.2.6.7 Non-volatile Secondary Low/Mid Address Space Block Locking Register (NVSLL) 565
17.2.6.8 Low/Mid aDdress Space Block Select Register (LMS) 567
17.2.6.9 High Address Space Block Select Register (HBS) 568
17.2.6.10 Address Register (ADR) 569
17.2.6.11 Bus Interface Unit 0 register (BIU0) 571
17.2.6.12 Bus Interface Unit 1 register (BIU1) 571
17.2.6.13 Bus Interface Unit 2 register (BIU2) 572
17.2.6.14 Non-volatile Bus Interface Unit 2 register (NVBIU2) 572
17.2.6.15 User Test 0 register (UT0) 573
17.2.6.16 User Test 1 register (UT1) 575
17.2.6.17 User Test 2 register (UT2) 576
17.2.6.18 User Multiple Input Signature Register 0 (UMISR0) 576
17.2.6.19 User Multiple Input Signature Register 1 (UMISR1) 577
17.2.6.20 User Multiple Input Signature Register 2 (UMISR2) 578
17.2.6.21 User Multiple Input Signature Register 3 (UMISR3) 578

17.2.6.22 User Multiple Input Signature Register 4 (UMISR4) 579
17.2.6.23 Non-volatile private censorship PassWord 0 register (NVPWD0) . .580
17.2.6.24 Non-Volatile Private Censorship Password 1 Register (NVPWD1) .581
17.2.6.25 Non-volatile System Censoring Information 0 register (NVSCI0) . . .581
17.2.6.26 Non-Volatile System Censoring Information 1 register (NVSCI1) . .582
17.2.6.27 Non-Volatile User Options register (NVUSRO) 583
17.2.7 Programming considerations 584
17.2.7.1 Modify operation 584
17.2.7.2 Error Correction Code (ECC) 592
17.2.7.3 Protection strategy 592
17.3 Data flash memory 594
17.3.1 Introduction 594
17.3.2 Main features 594
17.3.3 Block diagram 595
17.3.4 Functional description 596
17.3.4.1 Macrocell structure 596
17.3.4.2 Flash module sectorization 596
17.3.5 User mode operation 598
17.3.5.1 Reset 598
17.3.5.2 Power-Down mode 599
17.3.5.3 Low-Power mode 599
17.3.6 Register description 600
17.3.6.1 Module Configuration Register (MCR) 601
17.3.6.2 Low/Mid Address Space Block Locking Register (LML) 605
17.3.6.3 Non-Volatile Low/Mid Address Space Block Locking Register (NVLML) . 606
17.3.6.4 High Address Space Block Locking Register (HBL) 608
17.3.6.5 Non-Volatile High Address Space Block Locking Register (NVHBL) .608
17.3.6.6 Secondary Low/Mid Address Space Block Locking Register (SLL) . .609
17.3.6.7 Non-Volatile Secondary Low/Mid Address Space Block Locking Register (NVSLL) 610
17.3.6.8 Low/Mid Address Space Block Select Register (LMS) 612
17.3.6.9 High Address Space Block Select Register (HBS) 613
17.3.6.10 Address Register (ADR) 614
17.3.6.11 User Test 0 register (UT0) 616
17.3.6.12 User Test 1 register (UT1) 618
17.3.6.13 User Test 2 register (UT2) 618
17.3.6.14 User Multiple Input Signature Register 0 (UMISR0) 619
17.3.6.15 User Multiple Input Signature Register 1 (UMISR1) 620
17.3.6.16 User Multiple Input Signature Register 2 (UMISR2) 620
17.3.6.17 User Multiple Input Signature Register 3 (UMISR3) 621
17.3.6.18 User Multiple Input Signature Register 4 (UMISR4) 622
17.3.7 Programming considerations 622
17.3.7.1 Modify operation 622
17.3.7.2 Double Word program 623

17.3.7.3 Sector erase 625
17.3.7.4 User Test mode 627
17.3.8 Error Correction Code (ECC) 631
17.3.8.1 ECC algorithm 631
17.3.8.2 Bit manipulation 631
17.3.8.3 EEPROM emulation 632
17.3.9 Protection strategy 632
17.3.9.1 Modify protection 632
17.3.9.2 Censored mode 633
17.4 Platform flash controller (PFLASH2P_LCA) 633
17.4.1 Introduction 633
17.4.1.1 Overview 636
17.4.1.2 Features 636
17.4.1.3 Modes of operation 639
17.4.2 External signal descriptions 639
17.4.3 Memory map and register definition 639
17.4.3.1 Memory map 639
17.4.3.2 Register descriptions 641
17.4.4 Functional description 648
17.4.4.1 Access protections 649
17.4.4.2 Read cycles—buffer miss 649
17.4.4.3 Read cycles—buffer hit 650
17.4.4.4 Write cycles 650
17.4.4.5 Error termination 650
17.4.4.6 Access pipelining 651
17.4.4.7 Flash error response operation 651
17.4.4.8 Bank 0 and 2 page read buffers and prefetch operation 651
17.4.4.9 Bank1 temporary holding registers 654
17.4.4.10 Input port arbitration 654
17.4.4.11 Read-While-Write functionality 655
17.4.4.12 Wait-State emulation 656
17.4.4.13 Timing diagrams 657
17.5 Initialization / application information 663
17.5.1 Background 663
17.5.2 Flash memory setting recommendations 664

Chapter 18
FlexCAN

18.1 Introduction 669
18.1.1 Overview 669
18.1.2 FlexCAN module features 670
18.1.3 Modes of operation 671
18.2 External signal description 671
18.2.1 Overview 671
18.2.2 Signal descriptions 672

18.2.2.1 CAN Rx 672
18.2.2.2 CAN Tx 672
18.3 Memory map and register description 672
18.3.1 FlexCAN memory mapping 672
18.3.2 Message Buffer structure 674
18.3.3 Rx FIFO Structure 677
18.3.4 Register descriptions 679
18.3.4.1 Module Configuration Register (MCR) 679
18.3.4.2 Control Register (CTRL) 683
18.3.4.3 Free Running Timer (TIMER) 686
18.3.4.4 Rx Global Mask (RXGMASK) 687
18.3.4.5 Rx 14 Mask (RX14MASK) 688
18.3.4.6 Rx 15 Mask (RX15MASK) 688
18.3.4.7 Error Counter Register (ECR) 688
18.3.4.8 Error and Status Register (ESR) 690
18.3.4.9 Interrupt Mask Register High (IMRH) 692
18.3.4.10 Interrupt Mask Register Low (IMRL) 693
18.3.4.11 Interrupt Flag Register High (IFRH) 694
18.3.4.12 Interrupt Flag Register Low (IFRL) 694
18.3.4.13 Rx Individual Mask Registers (RXIMR0–RXIMR63) 696
18.4 Functional description 697
18.4.1 Overview 697
18.4.2 Transmit process 698
18.4.3 Arbitration process 698
18.4.4 Receive process 699
18.4.5 Matching process 700
18.4.6 Data coherence 702
18.4.6.1 Transmission abort mechanism 702
18.4.6.2 Message Buffer deactivation 703
18.4.6.3 Message Buffer lock mechanism 703
18.4.7 Rx FIFO 704
18.4.8 CAN protocol related features 705
18.4.8.1 Remote frames 705
18.4.8.2 Overload frames 706
18.4.8.3 Time stamp 706
18.4.8.4 Protocol timing 706
18.4.8.5 Arbitration and matching timing 709
18.4.9 Modes of operation: details 710
18.4.9.1 Freeze mode 710
18.4.9.2 Module Disable mode 710
18.4.10Interrupts 711
18.4.11Bus interface 711
18.5 Initialization/application information 712
18.5.1 FlexCAN initialization sequence 712
18.5.2 FlexCAN Addressing and RAM size configurations 713

Chapter 19
IEEE 1149.1 Test Access Port Controller (JTAGC)

19.1 Introduction . . . 715
19.2 Block diagram . . . 715
19.3 Overview . . . 715
19.4 Features . . . 716
19.5 Modes of operation . . . 716
19.5.1 Reset . . . 716
19.5.2 IEEE 1149.1-2001 defined test modes . . . 716
19.5.2.1 Bypass mode . . . 717
19.5.2.2 TAP sharing mode . . . 717
19.6 External signal description . . . 718
19.7 Memory map and register description . . . 718
19.7.1 Instruction register . . . 719
19.7.2 Bypass register . . . 719
19.7.3 Device Identification register . . . 719
19.7.4 Boundary Scan register . . . 720
19.8 Functional description . . . 720
19.8.1 JTAGC reset configuration . . . 720
19.8.2 IEEE 1149.1-2001 (JTAG) Test Access Port . . . 720
19.8.3 TAP Controller state machine . . . 721
19.8.3.1 Selecting an IEEE 1149.1-2001 register . . . 723
19.8.4 JTAGC instructions . . . 723
19.8.4.1 BYPASS instruction . . . 724
19.8.4.2 ACCESS_AUX_TAP_*x* instructions . . . 724
19.8.4.3 EXTEST — External Test instruction . . . 724
19.8.4.4 IDCODE instruction . . . 725
19.8.4.5 SAMPLE instruction . . . 725
19.8.4.6 SAMPLE/PRELOAD instruction . . . 725
19.8.5 Boundary scan . . . 725
19.9 e200z0 OnCE controller . . . 726
19.9.1 e200z0 OnCE controller block diagram . . . 726
19.9.2 e200z0 OnCE controller functional description . . . 726
19.9.2.1 Enabling the TAP controller . . . 726
19.9.3 e200z0 OnCE controller register description . . . 727
19.9.3.1 OnCE Command Register (OCMD) . . . 727
19.10 Initialization/application information . . . 728

Chapter 20
Inter-Integrated Circuit Bus Controller Module (I^2C)

20.1 Introduction . . . 729
20.1.1 Overview . . . 729
20.1.2 Features . . . 729
20.1.3 Block diagram . . . 729

20.2 Modes of operation 730
20.3 External signal description 730
20.3.1 Overview 730
20.3.2 Detailed signal descriptions 730
20.3.2.1 SCL 730
20.3.2.2 SDA 730
20.4 Memory map and register description 731
20.4.1 Overview 731
20.4.2 Module memory map 731
20.4.3 Register description 731
20.4.3.1 I2C Bus Address Register 732
20.4.3.2 I2C Bus Frequency Divider Register 732
20.4.3.3 I2C Bus Control Register 738
20.4.3.4 I2C Bus Status Register 739
20.4.3.5 I2C Bus Data I/O Register 740
20.4.3.6 I2C Bus Interrupt Configuration Register 741
20.5 Functional description 741
20.5.1 General 741
20.5.2 I-Bus Protocol 741
20.5.2.1 START Signal 742
20.5.2.2 Slave Address Transmission 743
20.5.2.3 Data Transfer 743
20.5.2.4 Stop Signal 743
20.5.2.5 Repeated START Signal 744
20.5.2.6 Arbitration Procedure 744
20.5.2.7 Clock Synchronization 744
20.5.2.8 Handshaking 745
20.5.2.9 Clock Stretching 745
20.5.3 Interrupts 745
20.5.3.1 General 745
20.5.3.2 Interrupt Description 745
20.6 Initialization/application information 746
20.6.1 I2C Programming Examples 746
20.6.1.1 Initialization Sequence 746
20.6.1.2 Generation of START 746
20.6.1.3 Post-Transfer Software Response 746
20.6.1.4 Generation of stop 747
20.6.1.5 Generation of repeated START 748
20.6.1.6 Slave mode 748
20.6.1.7 Arbitration Lost 748
20.6.2 DMA application information 750

Chapter 21
Interrupt Controller (INTC)

21.1 Introduction 751

21.2 Features 751
21.3 Block diagram 753
21.4 Modes of operation 753
21.4.1 Normal mode 753
21.4.1.1 Software vector mode 754
21.4.1.2 Hardware vector mode 754
21.4.1.3 Debug mode 754
21.4.1.4 Stop mode 755
21.5 Memory map and register description 755
21.5.1 Module memory map 755
21.5.2 Register description 755
21.5.2.1 INTC Module Configuration Register (INTC_MCR) 756
21.5.2.2 INTC Current Priority Register for Processor (INTC_CPR) 756
21.5.2.3 INTC Interrupt Acknowledge Register (INTC_IACKR) 758
21.5.2.4 INTC End-of-Interrupt Register (INTC_EOIR) 759
21.5.2.5 INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR4_7) 759
21.5.2.6 INTC Priority Select Registers (INTC_PSR0_3–INTC_PSR204_207) . . . 761
21.6 Functional description 762
21.6.1 Interrupt Request Sources 772
21.6.1.1 Peripheral Interrupt Requests 773
21.6.1.2 Software configurable Interrupt Requests 773
21.6.1.3 Unique Vector for Each Interrupt Request Source 773
21.6.2 Priority management 773
21.6.2.1 Current Priority and Preemption 773
21.6.2.2 Last-In First-Out (LIFO) 774
21.6.3 Handshaking with processor 775
21.6.3.1 Software vector mode handshaking 775
21.6.3.2 Hardware vector mode handshaking 776
21.7 Initialization/application information 777
21.7.1 Initialization flow 777
21.7.2 Interrupt Exception Handler 777
21.7.2.1 Software vector mode 778
21.7.2.2 Hardware vector mode 778
21.7.3 ISR, RTOS, and task hierarchy 779
21.7.4 Order of execution 780
21.7.5 Priority Ceiling Protocol 781
21.7.5.1 Elevating Priority 781
21.7.5.2 Ensuring Coherency 781
21.7.6 Selecting Priorities According to Request Rates and Deadlines 781
21.7.7 Software configurable interrupt requests 782
21.7.7.1 Scheduling a lower priority portion of an ISR 782
21.7.7.2 Scheduling an ISR on another processor 783
21.7.8 Lowering priority within an ISR 783

21.7.9 Negating an Interrupt Request Outside of its ISR 783
21.7.9.1 Negating an Interrupt Request as a Side Effect of an ISR 783
21.7.9.2 Negating multiple interrupt requests in one ISR 783
21.7.9.3 Proper setting of interrupt request priority 784
21.7.10 Examining LIFO contents 784

Chapter 22
LCD Driver (LCD64F6B)

22.1 Information specific to this device 785
22.1.1 Number of front and back planes 785
22.1.2 LCD clock selection 785
22.1.3 Settings during Standby mode 785
22.2 Introduction 786
22.2.1 Overview 786
22.2.2 Features 787
22.2.3 Modes of operation 788
22.3 External signal description 788
22.3.1 Detailed signal descriptions 788
22.4 Memory map and register definition 789
22.4.1 Memory map 789
22.4.2 Register descriptions 790
22.4.2.1 LCD Control Register (LCDCR) 790
22.4.2.2 LCD Prescaler Control Register (LCDPCR) 793
22.4.2.3 LCD Contrast Control Register (LCDCCR) 794
22.4.2.4 LCD Frontplane Enable Register 0 (FPENR0) 795
22.4.2.5 LCD Frontplane Enable Register 1 (FPENR1) 796
22.4.2.6 LCDRAM (Location 0) 797
22.4.2.7 LCDRAM (Location 1) 798
22.4.2.8 LCDRAM (Location 2) 799
22.4.2.9 LCDRAM (Location 3) 800
22.4.2.10 LCDRAM (Location 4) 801
22.4.2.11 LCDRAM (Location 5) 802
22.4.2.12 LCDRAM (Location 6) 803
22.4.2.13 LCDRAM (Location 7) 804
22.4.2.14 LCDRAM (Location 8) 805
22.4.2.15 LCDRAM (Location 9) 806
22.4.2.16 LCDRAM (Location 10) 807
22.4.2.17 LCDRAM (Location 11) 808
22.4.2.18 LCDRAM (Location 12) 809
22.4.2.19 LCDRAM (Location 13) 810
22.4.2.20 LCDRAM (Location 14) 811
22.4.2.21 LCDRAM (Location 15) 812
22.5 Functional description 812
22.5.1 Frontplane, backplane, and LCD system during reset 812
22.5.2 LCD clock and frame frequency 813

22.5.3 Contrast adjustment 814
22.5.3.1 Adjusting the supply voltage (VLCD) 814
22.5.3.2 Adding contrast adjustment phases 814
22.5.4 LCD RAM 815
22.5.5 LCD driver system enable and frontplane enable sequencing 815
22.5.6 LCD driver backplane remapping 815
22.5.7 LCD bias and modes of operation 816
22.5.8 Operation in power saving modes 818
22.5.8.1 Operation in Stop mode 818
22.5.8.2 Operation in Standby mode 818
22.5.9 Other power saving 818
22.5.9.1 LCD reference clock select 818
22.5.9.2 Boost at switching 819
22.5.9.3 Standard drive selection 819
22.5.9.4 Usage recommendation 819
22.5.10 Interrupts 820
22.5.10.1 EOF interrupt 820
22.6 LCD waveform examples 821
22.6.1 1/1 Duty multiplexed with 1/1 Bias mode 821
22.6.2 1/2 duty multiplexed with 1/2 Bias mode 822
22.6.3 1/2 duty multiplexed with 1/3 Bias mode 822
22.6.4 1/3 Duty multiplexed with 1/3 Bias mode 823
22.6.5 1/4 Duty multiplexed with 1/3 Bias mode 824
22.6.6 1/5 Duty multiplexed with 1/3 Bias 825
22.6.7 1/6 Duty multiplexed with 1/3 Bias mode 826
22.7 Initialization information 827

Chapter 23
LIN Controller (LINFlex)

23.1 Introduction 829
23.2 Main features 829
23.2.1 LIN mode features 829
23.2.2 UART mode features 829
23.2.3 Features common to LIN and UART 830
23.3 General description 830
23.4 Fractional baud rate generation 831
23.5 Operating modes 833
23.5.1 Initialization mode 833
23.5.2 Normal mode 833
23.5.3 Low-power mode (Sleep) 834
23.6 Test modes 834
23.6.1 Loopback mode 834
23.6.2 Self-test mode 834
23.7 Memory map and registers description 835
23.7.1 Memory map 835

23.7.2 Register description 836
23.7.2.1 LIN control register 1 (LINCR1) 837
23.7.2.2 LIN interrupt enable register (LINIER) 840
23.7.2.3 LIN status register (LINSR) 842
23.7.2.4 LIN error status register (LINESR) 845
23.7.2.5 UART mode control register (UARTCR) 846
23.7.2.6 UART mode status register (UARTSR) 848
23.7.2.7 LIN timeout control status register (LINTCSR) 850
23.7.2.8 LIN output compare register (LINOCR) 851
23.7.2.9 LIN timeout control register (LINTOCR) 851
23.7.2.10 LIN fractional baud rate register (LINFBRR) 852
23.7.2.11 LIN integer baud rate register (LINIBRR) 853
23.7.2.12 LIN checksum field register (LINCFR) 854
23.7.2.13 LIN control register 2 (LINCR2) 854
23.7.2.14 Buffer identifier register (BIDR) 856
23.7.2.15 Buffer data register LSB (BDRL) 857
23.7.2.16 Buffer data register MSB (BDRM) 857
23.7.2.17 Identifier filter enable register (IFER) 858
23.7.2.18 Identifier filter match index (IFMI) 859
23.7.2.19 Identifier filter mode register (IFMR) 860
23.7.2.20 Identifier filter control register (IFCR2*n*) 861
23.7.2.21 Identifier filter control register (IFCR2*n* + 1) 862
23.7.3 Register map and reset values 863
23.8 Functional description 867
23.8.1 UART mode 867
23.8.1.1 Buffer in UART mode 867
23.8.1.2 UART transmitter 868
23.8.1.3 UART receiver 868
23.8.1.4 Clock gating 869
23.8.2 LIN mode 869
23.8.2.1 Master mode 869
23.8.2.2 Slave mode 871
23.8.2.3 Slave mode with identifier filtering 873
23.8.2.4 Slave mode with automatic resynchronization 875
23.8.2.5 Clock gating 877
23.8.3 8-bit timeout counter 877
23.8.3.1 LIN timeout mode 877
23.8.3.2 Output compare mode 879
23.8.4 Interrupts 879

Chapter 24
Memory Protection Unit (MPU)

24.1 Introduction 881
24.1.1 Overview 881
24.1.2 Features 883

24.1.3 Modes of operation 883
24.1.4 External signal description 884
24.2 Memory map and register description 884
24.2.1 Memory map 884
24.2.2 Register description 885
24.2.2.1 MPU Control/Error Status Register (MPU_CESR) 885
24.2.2.2 MPU Error Address Register, Slave Port *n* (MPU_EAR*n*) 886
24.2.2.3 MPU Error Detail Register, Slave Port *n* (MPU_EDR*n*) 887
24.2.2.4 MPU Region Descriptor *n* (MPU_RGD*n*) 888
24.2.2.5 MPU Region Descriptor Alternate Access Control *n* (MPU_RGDAAC*n*) . 893
24.3 Functional description 895
24.3.1 Access evaluation macro 895
24.3.1.1 Access evaluation—hit determination 896
24.3.1.2 Access evaluation—privilege violation determination 896
24.3.2 Putting It All Together and AHB Error Terminations 897
24.4 Initialization Information 898
24.5 Application information 898

Chapter 25
Mode Entry Module (MC_ME)

25.1 Introduction 901
25.1.1 Overview 901
25.1.2 Features 903
25.1.3 Modes of operation 903
25.2 External signal description 904
25.3 Memory map and register definition 904
25.3.1 Memory map 905
25.3.2 Register description 912
25.3.2.1 Global Status Register (ME_GS) 912
25.3.2.2 Mode Control Register (ME_MCTL) 914
25.3.2.3 Mode Enable Register (ME_ME) 915
25.3.2.4 Interrupt Status Register (ME_IS) 917
25.3.2.5 Interrupt Mask Register (ME_IM) 918
25.3.2.6 Invalid Mode Transition Status Register (ME_IMTS) 919
25.3.2.7 Debug Mode Transition Status Register (ME_DMTS) 920
25.3.2.8 Reset Mode Configuration Register (ME_RESET_MC) 922
25.3.2.9 Test Mode Configuration Register (ME_TEST_MC) 923
25.3.2.10 Safe Mode Configuration Register (ME_SAFE_MC) 923
25.3.2.11 DRUN Mode Configuration Register (ME_DRUN_MC) 924
25.3.2.12 Run0…3 Mode Configuration Registers (ME_RUN0…3_MC) 925
25.3.2.13 Halt Mode Configuration Register (ME_HALT_MC) 925
25.3.2.14 Stop Mode Configuration Register (ME_STOP_MC) 926
25.3.2.15 Standby Mode Configuration Register (ME_STANDBY_MC) 926
25.3.2.16 Peripheral Status Register 0 (ME_PS0) 928

25.3.2.17 Peripheral Status Register 1 (ME_PS1) 929
25.3.2.18 Peripheral Status Register 2 (ME_PS2) 930
25.3.2.19 Peripheral Status Register 3 (ME_PS3) 930
25.3.2.20 Run Peripheral Configuration Registers (ME_RUN_PC0...7) 931
25.3.2.21 Low-Power Peripheral Configuration Registers (ME_LP_PC0...7) . 932
25.3.2.22 Peripheral Control Registers (ME_PCTL0...143) 933
25.4 Functional description 933
25.4.1 Mode Transition Request 933
25.4.2 Mode details 935
25.4.2.1 Reset mode 935
25.4.2.2 DRUN mode 935
25.4.2.3 Safe mode 935
25.4.2.4 Test mode 936
25.4.2.5 Run0...3 modes 937
25.4.2.6 Halt mode 937
25.4.2.7 Stop mode 938
25.4.2.8 Standby mode 939
25.4.3 Mode transition process 939
25.4.3.1 Target mode request 940
25.4.3.2 Target mode configuration loading 940
25.4.3.3 Peripheral Clocks Disable 941
25.4.3.4 Processor Low-Power mode entry 942
25.4.3.5 Processor and system memory clock disable 942
25.4.3.6 Clock sources switch-on 942
25.4.3.7 Main Voltage Regulator Switch-On 942
25.4.3.8 Flash Modules Switch-On 943
25.4.3.9 FMPLL0 Switch-On 943
25.4.3.10 Power Domain #2 Switch-On 943
25.4.3.11 Pad Outputs-On 943
25.4.3.12 Peripheral Clocks Enable 944
25.4.3.13 Processor and memory clock enable 944
25.4.3.14 Processor Low-Power mode exit 944
25.4.3.15 System clock switching 944
25.4.3.16 Power Domain #2 Switch-Off 945
25.4.3.17 Pad Switch-Off 945
25.4.3.18 FMPLL0 Switch-Off 946
25.4.3.19 Clock Sources Switch-Off 946
25.4.3.20 Flash Switch-Off 946
25.4.3.21 Main Voltage Regulator Switch-Off 946
25.4.3.22 Current mode update 947
25.4.4 Protection of mode configuration registers 949
25.4.5 Mode transition interrupts 949
25.4.5.1 Invalid mode configuration interrupt 949
25.4.5.2 Invalid mode transition interrupt 949
25.4.5.3 Safe mode transition interrupt 951

25.4.5.4 Mode transition complete interrupt 951
25.4.6 Peripheral clock gating 951
25.4.7 Application Example 952

Chapter 26
Nexus Development Interface (NDI)

26.1 Introduction 955
26.2 Block diagram 955
26.3 Features 956
26.4 Modes of operation 957
26.4.1 Nexus Reset 957
26.4.2 Operating mode 958
26.4.2.1 Disabled-Port mode 958
26.4.2.2 Censored mode 958
26.4.2.3 Stop mode 958
26.5 External signal description 958
26.5.1 Nexus signal reset states 958
26.6 Memory map and register description 959
26.6.1 Nexus Debug Interface (NDI) registers 959
26.6.2 Register description 959
26.6.2.1 Nexus Device ID Register (DID) 959
26.6.2.2 Port Configuration Register (PCR) 960
26.6.2.3 Development Control Register 1, 2 (DC1, DC2) 962
26.6.2.4 Development Status Register (DS) 965
26.6.2.5 Read/Write Access Control/Status (RWCS) 966
26.6.2.6 Read/Write Access Address (RWA) 967
26.6.2.7 Read/Write Access Data (RWD) 967
26.6.2.8 Watchpoint Trigger Register (WT) 968
26.7 Functional description 969
26.7.1 NPC_HNDSHK module 969
26.7.2 Enabling Nexus clients for TAP access 970
26.7.3 Configuring the NDI for Nexus messaging 971
26.7.4 Programmable MCKO frequency 971
26.7.5 Nexus messaging 972
26.7.6 EVTO sharing 972
26.7.7 Debug mode control 972
26.7.7.1 EVTI generated break request 972
26.7.8 Nexus reset control 972

Chapter 27
Periodic Interrupt Timer (PIT)

27.1 Introduction 973
27.1.1 Overview 973
27.1.2 Features 973

27.2 Signal description 974
27.3 Memory map and register description 974
27.3.1 Memory map 974
27.3.2 Register descriptions 975
27.3.2.1 PIT Module Control Register (PITMCR) 975
27.3.2.2 Timer Load Value (LDVAL) register 975
27.3.2.3 Current Timer Value (CVAL) register 976
27.3.2.4 Timer Control (TCTRL) register 977
27.3.2.5 Timer Flag (TFLG) register 977
27.4 Functional description 978
27.4.1 General 978
27.4.1.1 Timers 978
27.4.1.2 Debug mode 979
27.4.2 Interrupts 979
27.5 Initialization and application information 979
27.5.1 Example configuration 979

Chapter 28
Peripheral Bridge (PBRIDGE)

28.1 Introduction 981
28.1.1 Overview 981
28.1.2 Features 981
28.1.3 Modes of operation 981
28.2 Functional description 981
28.2.1 Access support 981
28.2.1.1 Peripheral write buffering 981
28.2.1.2 Read Cycles 982
28.2.1.3 Write Cycles 982
28.2.2 General Operation 982

Chapter 29
Power Control Unit (MC_PCU)

29.1 Introduction 983
29.1.1 Overview 983
29.1.2 Features 984
29.1.3 Modes of operation 984
29.2 External signal description 985
29.3 Memory map and register definition 985
29.3.1 Memory map 985
29.3.2 Register descriptions 986
29.3.2.1 Power Domain #0 Configuration Register (PCU_PCONF0) 986
29.3.2.2 Power Domain #1 Configuration Register (PCU_PCONF1) 988
29.3.2.3 Power Domain #2 Configuration Register (PCU_PCONF2) 988
29.3.2.4 Power Domain Status Register (PCU_PSTAT) 989

29.4 Functional description 989
29.4.1 General 989
29.4.2 Reset / Power-On Reset 989
29.4.3 MC_PCU Configuration 989
29.4.4 Mode transitions 990
29.4.4.1 DRUN, Safe, Test, Run0…3, Halt, and Stop mode transition 990
29.4.4.2 Standby mode transition 991
29.4.4.3 Power saving for memories during Standby mode 992
29.5 Initialization information 992
29.6 Application information 992
29.6.1 Standby mode considerations 992

Chapter 30
Quad Serial Peripheral Interface (QuadSPI)

30.1 Preface 993
30.1.1 Conventions 993
30.1.2 Acronyms and Abbreviations 993
30.1.3 Glossary for QuadSPI module 994
30.2 Introduction 995
30.2.1 Overview 997
30.2.2 Features 997
30.2.3 QuadSPI modes of operation 998
30.2.3.1 SPI Master mode 998
30.2.3.2 SPI Slave mode 998
30.2.3.3 Serial Flash mode 998
30.2.3.4 Module Disable mode 998
30.2.3.5 Stop mode 998
30.2.3.6 Debug mode (SPI modes only) 999
30.3 External signal description 999
30.3.1 Overview 999
30.3.2 Detailed Signal Description 1000
30.3.2.1 PCS_C0 — Peripheral Chip Select/Slave Select 1000
30.3.2.2 PCS[3:1] — Peripheral Chip Selects 1–3 1000
30.3.2.3 PCS4 — Peripheral Chip Select 4 1000
30.3.2.4 PCS[7:5] — Peripheral Chip Selects 5–7 1000
30.3.2.5 SI_IO0 — Serial Input, QuadSPI Data IO_0 1000
30.3.2.6 SO_IO1 — Serial Output, QuadSPI Data IO_1 1000
30.3.2.7 QSPI_IO2—QuadSPI Data IO_2 1001
30.3.2.8 QSPI_IO3—QuadSPI Data IO_3 1001
30.3.2.9 SCK — Serial Clock 1001
30.4 Memory map and register definition 1001
30.4.1 IP bus register memory map 1001
30.4.2 AMBA bus register memory map 1002
30.4.3 IP bus register descriptions 1003
30.4.3.1 Register write access 1003

30.4.3.2 Module Configuration Register (QSPI_MCR) 1003
30.4.3.3 Transfer Count Register (QSPI_TCR) 1006
30.4.3.4 Clock and Transfer Attributes Registers 0 – 1 (QSPI_CTAR0 – QSPI_CTAR1) 1007
30.4.3.5 SPI Status Register (QSPI_SPISR) 1012
30.4.3.6 SPI Interrupt and DMA Request Select and Enable Register (QSPI_SPIRSER) 1013
30.4.3.7 PUSH TX FIFO Register (QSPI_PUSHR) 1015
30.4.3.8 POP RX FIFO Register (QSPI_POPR) 1016
30.4.3.9 Transmit FIFO Registers 0 – 14 (QSPI_TXFR0 – QSPI_TXFR14) . 1017
30.4.3.10 RX FIFO Registers 0 – 14 (QSPI_RXFR0 – QSPI_RXFR14) 1018
30.4.3.11 Serial Flash Address Register (QSPI_SFAR) 1018
30.4.3.12 Instruction Code Register (QSPI_ICR) 1019
30.4.3.13 Sampling Register (QSPI_SMPR) 1020
30.4.3.14 RX Buffer Status Register (QSPI_RBSR) 1021
30.4.3.15 RX Buffer Data Registers 0–14 (QSPI_RBDR0–QSPI_RBDR14) . 1022
30.4.3.16 TX Buffer Status Register (QSPI_TBSR) 1023
30.4.3.17 TX Buffer Data Register (QSPI_TBDR) 1024
30.4.3.18 AMBA Control Register (QSPI_ACR) 1024
30.4.3.19 Serial Flash Mode Status Register (QSPI_SFMSR) 1025
30.4.3.20 Serial Flash Mode Flag Register (QSPI_SFMFR) 1027
30.4.3.21 SFM Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER) 1029
30.4.4 AHB bus register memory map descriptions 1029
30.4.4.1 AHB bus access considerations 1030
30.4.4.2 Memory-mapped serial flash data (QSPI_SFD) 1030
30.4.4.3 AHB RX Data Buffer (QSPI_ARDB) 1030
30.5 Functional description 1031
30.5.1 Modes of operation 1031
30.5.2 SPI (Serial Peripheral Interface) modes 1032
30.5.2.1 Start and Stop of SPI Transfers 1033
30.5.2.2 Master mode 1034
30.5.2.3 Slave mode 1034
30.5.2.4 FIFO Disable Operation 1034
30.5.2.5 Transmit First In First Out (TX FIFO) Buffering Mechanism 1034
30.5.2.6 Receive First In First Out (RX FIFO) Buffering Mechanism 1035
30.5.2.7 Baud Rate and Clock Delay Generation 1036
30.5.2.8 SPI Transfer Formats 1038
30.5.2.9 Continuous Serial Communications Clock 1045
30.5.2.10 SPI mode interrupt and DMA requests 1046
30.5.3 SFM (Serial Flash) mode 1047
30.5.3.1 Issuing SFM Commands 1048
30.5.3.2 Flash Programming 1049
30.5.3.3 Flash Read 1049
30.5.3.4 Byte Ordering of Serial Flash Data 1051

30.5.3.5 Serial Flash mode interrupt and DMA requests 1052
30.5.3.6 TX Buffer Operation 1053
30.5.4 Power saving features 1054
30.5.4.1 Module Disable mode 1054
30.5.4.2 Leaving power-saving modes 1055
30.5.4.3 Slave bus signal gating 1055
30.6 Initialization/application information 1055
30.6.1 How to change queues—SPI modes only 1055
30.6.2 Baud rate settings—SPI modes only 1056
30.6.3 Delay settings—SPI modes only 1057
30.6.4 Oak family compatibility with the QuadSPI—SPI modes only 1058
30.6.5 Calculation of FIFO pointer addresses—SPI modes only 1059
30.6.5.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO 1060
30.6.5.2 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO 1060
30.6.6 Available status/flag information—SFM mode only 1061
30.6.6.1 IP bus related commands 1061
30.6.6.2 AHB bus related commands 1061
30.6.6.3 Overview of error flags 1062
30.6.6.4 IP bus and AHB Access command collisions 1062
30.6.7 Command arbitration—SFM mode only 1062
30.6.8 DMA usage 1063
30.6.8.1 DMA usage in SPI Slave mode 1063
30.6.8.2 DMA usage in SFM mode 1063
30.7 Serial Flash Devices 1064
30.7.1 Supported Instruction Codes in Winbond Devices 1065
30.7.2 Serial Flash Clock Frequency Limitations 1067
30.8 Internal Sampling of Serial Flash Input Data 1067

Chapter 31
Reset Generation Module (MC_RGM)

31.1 Introduction 1071
31.1.1 Overview 1071
31.1.2 Features 1072
31.1.3 Modes of operation 1073
31.2 External signal description 1074
31.3 Memory map and register definition 1074
31.3.1 Register descriptions 1076
31.3.1.1 Functional Event Status Register (RGM_FES) 1077
31.3.1.2 Destructive Event Status Register (RGM_DES) 1078
31.3.1.3 Functional Event Reset Disable Register (RGM_FERD) 1080
31.3.1.4 Destructive Event Reset Disable Register (RGM_DERD) 1081
31.3.1.5 Functional Event Alternate Request Register (RGM_FEAR) 1082
31.3.1.6 Destructive Event Alternate Request Register (RGM_DEAR) 1083

31.3.1.7 Functional Event Short Sequence Register (RGM_FESS) 1084
31.3.1.8 Standby Reset Sequence Register (RGM_STDBY) 1086
31.3.1.9 Functional Bidirectional Reset Enable Register (RGM_FBRE) 1086
31.4 Functional description 1087
31.4.1 Reset State Machine 1087
31.4.1.1 Phase0 Phase 1089
31.4.1.2 Phase1 Phase 1090
31.4.1.3 Phase2 Phase 1090
31.4.1.4 Phase3 Phase 1090
31.4.1.5 Idle Phase 1090
31.4.2 Destructive Resets 1091
31.4.3 External Reset 1091
31.4.4 Functional Resets 1092
31.4.5 Standby Entry Sequence 1092
31.4.6 Alternate Event Generation 1093
31.4.7 Boot mode capturing 1093

Chapter 32
Real-Time Clock (RTC/API)

32.1 Overview 1095
32.2 Features 1095
32.3 Device specific information 1097
32.4 Modes of operation 1097
32.5 Debug support 1098
32.6 Register descriptions 1098
32.6.1 RTC Supervisor Control Register (RTCSUPV) 1098
32.6.2 RTC Control Register (RTCC) 1099
32.6.3 RTC Status Register (RTCS) 1101
32.6.4 RTC Counter Register (RTCCNT) 1102
32.7 RTC functional description 1102
32.8 API functional description 1103

Chapter 33
Static RAM (SRAM)

33.1 Introduction 1105
33.2 General-purpose SRAM 1105
33.3 Graphics SRAM 1105
33.4 Low-power configuration 1105
33.5 Register memory map 1106
33.6 SRAM ECC mechanism 1106
33.6.1 Access timing 1106
33.6.2 Reset effects on SRAM accesses 1107
33.7 Functional description 1107
33.8 Initialization and application information 1108

Chapter 34
Sound Generation Logic (SGL)

34.1 Introduction 1109
34.1.1 Overview 1109
34.1.2 Features 1109
34.2 External signal description 1110
34.2.1 Detailed signal descriptions 1110
34.3 Memory map and register definition 1111
34.3.1 Memory map 1111
34.3.2 Register descriptions 1111
34.3.2.1 MODE_SEL register 1111
34.3.2.2 SOUND_DURATION register 1113
34.3.2.3 HIGH_PERIOD register 1113
34.3.2.4 LOW_PERIOD register 1113
34.3.2.5 SGL_STATUS register 1114
34.4 Functional description 1114
34.4.1 Interrupts 1118

Chapter 35
Stepper Motor Controller (SMC)

35.1 Introduction 1119
35.1.1 Features 1119
35.1.2 Modes of operation 1119
35.1.2.1 Functional modes 1119
35.1.2.2 PWM channel configuration modes 1119
35.1.2.3 PWM alignment modes 1120
35.1.2.4 Low-power mode 1120
35.1.3 Block diagram 1121
35.2 External signal description 1122
35.2.1 M0C0M/M0C0P/M0C1M/M0C1P — PWM output pins for Motor 0 1122
35.2.2 M1C0M/M1C0P/M1C1M/M1C1P — PWM output pins for Motor 1 1123
35.2.3 M2C0M/M2C0P/M2C1M/M2C1P — PWM output pins for Motor 2 1123
35.2.4 M3C0M/M3C0P/M3C1M/M3C1P — PWM output pins for Motor 3 1123
35.2.5 M4C0M/M4C0P/M4C1M/M4C1P — PWM output pins for Motor 4 1123
35.2.6 M5C0M/M5C0P/M5C1M/M5C1P — PWM output pins for Motor 5 1123
35.3 Memory map and register definition 1123
35.3.1 Module memory map 1123
35.3.2 Register description 1125
35.3.2.1 Motor Controller Control Register 0 (MCCTL0) 1126
35.3.2.2 Motor Controller Control Register 1 (MCCTL1) 1127
35.3.2.3 Motor Controller Period Register (MCPER) 1128
35.3.2.4 Motor Controller Channel Control Register (MCCC0..11) 1128
35.3.2.5 Motor Controller Duty Cycle Register (MCDC0..11) 1129
35.3.2.6 Short-circuit Detector Timeout Register (MCSDTO) 1131

35.3.2.7 Short-circuit Detector Enable Register 0 (MCSDE0) 1131
35.3.2.8 Short-circuit Detector Enable Register 1 (MCSDE1) 1132
35.3.2.9 Short-circuit Detector Enable Register 2 (MCSDE2) 1132
35.3.2.10 Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0) . . 1133
35.3.2.11 Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1) . . 1133
35.3.2.12 Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2) . . 1134
35.3.2.13 Short-circuit Detector Interrupt Register 0 (MCSDI0) 1134
35.3.2.14 Short-circuit Detector Interrupt Register 1 (MCSDI1) 1135
35.3.2.15 Short-circuit Detector Interrupt Register 2 (MCSDI2) 1135
35.4 Functional description 1135
35.4.1 Modes of operation 1135
35.4.1.1 PWM output modes 1135
35.4.1.2 Relationship between PWM mode and PWM channel enable 1139
35.4.1.3 Relationship between Sign, Duty, Dither, RECIRC, Period, and PWM mode functions 1139
35.4.2 PWM Duty Cycle 1149
35.4.3 Motor Controller Counter Clock Source 1149
35.4.4 Output switching delay 1150
35.4.5 Operation in SMC stop mode 1150
35.4.6 Short-circuit detection 1150
35.5 Reset 1155
35.6 Interrupts 1155

Chapter 36
Stepper Stall Detect (SSD)

36.1 Introduction 1157
36.1.1 Overview 1157
36.1.2 Features 1159
36.1.3 Modes of operation 1159
36.1.3.1 Disabled mode 1159
36.1.3.2 Normal mode 1159
36.1.3.3 Stop mode 1160
36.2 External signal description 1160
36.3 Memory map and register definition 1160
36.3.1 Memory map 1160
36.3.2 Register descriptions 1161
36.3.2.1 SSD Control and Status Register (CONTROL) 1161
36.3.2.2 Interrupt Enable and Flag Register (IRQ) 1162
36.3.2.3 Integration Accumulator Register (ITGACC) 1163
36.3.2.4 Down Counter Register (DCNT) 1163
36.3.2.5 Blanking Counter Load Register (BLNCNTLD) 1164
36.3.2.6 Integration Counter Load Register (ITGCNTLD) 1164
36.3.2.7 SSD Prescale and Divider Register (PRESCALE) 1165
36.4 Functional description 1166
36.4.1 Main building blocks of the SSD 1166

36.4.1.1 Analog block 1166
36.4.1.2 Analog Wrapper + Port Control 1168
36.4.1.3 Register Interface 1170
36.4.1.4 BIS control 1171
36.4.2 Stepper Stall Detection Measurement 1174
36.4.2.1 Overview of the SSD Measurement 1174
36.4.2.2 Details of the SSD Measurement 1175
36.4.3 Additional modes of operation 1177
36.4.3.1 Blanking with No Drive 1177
36.4.3.2 Integration with No Drive 1177
36.5 Initialization Information 1177
36.5.1 Analog Block Startup Time 1177
36.5.2 Analog Block Polarity Switching Time 1177
36.5.3 SSD startup 1178
36.6 Application information 1178
36.6.1 Current flow examples 1178
36.6.2 Setting of the PRESCALE Register 1180
36.6.2.1 Timing Resolution Considerations 1180
36.6.2.2 Offset Cancellation Considerations 1181
36.6.3 Watching Internal States of the SSD 1181
36.6.4 Stepper Motor Transition Considerations 1182
36.6.4.1 SSD Phase-In and Phase-Out 1182
36.6.4.2 Changing of SSD Internal States 1182
36.6.5 Legacy modes—separate blanking and integration phase 1183

Chapter 37
System Integration Unit Lite (SIUL)

37.1 Introduction 1185
37.2 Overview 1185
37.3 Features 1187
37.4 External signal description 1187
37.4.1 Detailed signal descriptions 1188
37.4.1.1 General-purpose I/O pins (GPIO[0:132]) 1188
37.4.1.2 External interrupt request input pins (EIRQ[0:13]) 1188
37.5 Memory map and register description 1188
37.5.1 SIUL memory map 1188
37.5.2 Register protection 1190
37.5.3 Register description 1190
37.5.3.1 MCU ID Register #1 (MIDR1) 1190
37.5.3.2 MCU ID Register #2 (MIDR2) 1191
37.5.3.3 Interrupt Status Flag Register (ISR) 1193
37.5.3.4 Interrupt Request Enable Register (IRER) 1193
37.5.3.5 Interrupt Rising-Edge Event Enable Register (IREER) 1194
37.5.3.6 Interrupt Falling-Edge Event Enable Register (IFEER) 1194
37.5.3.7 Interrupt Filter Enable Register (IFER) 1195

37.5.3.8 Pad Configuration Registers (PCR0–PCR132) 1195
37.5.3.9 Pad Selection for Multiplexed Inputs Registers (PSMI0_3–PSMI40_42) . 1198
37.5.3.10 GPIO Pad Data Output Registers (GPDO0_3–GPDO132_135) . . 1202
37.5.3.11 GPIO Pad Data Input Registers (GPDI0_3–GPDI132_135) 1202
37.5.3.12 Parallel GPIO Pad Data Out Registers (PGPDO0–PGPDO4) 1203
37.5.3.13 Parallel GPIO Pad Data In Register (PGPDI0–PGPDI4) 1205
37.5.3.14 Masked Parallel GPIO Pad Data Out Register (MPGPDO0–MPGPDO8) 1206
37.5.3.15 Interrupt Filter Maximum Counter Registers (IFMC0–IFMC15) . . . 1207
37.5.3.16 Interrupt Filter Clock Prescaler Register (IFCPR) 1207
37.6 Functional description 1208
37.6.1 General 1208
37.6.2 Pad control 1208
37.6.3 General purpose input and output pads (GPIO) 1208
37.6.4 External interrupts 1209
37.6.4.1 External interrupt management 1210
37.7 Pin muxing 1210

Chapter 38
System Status and Configuration Module (SSCM)

38.1 Introduction 1213
38.1.1 Overview 1213
38.1.2 Features 1213
38.1.3 Modes of operation 1214
38.2 Memory map and register description 1214
38.2.1 Memory map 1214
38.2.2 Register description 1214
38.2.2.1 System Status Register (STATUS) 1215
38.2.2.2 System Memory Configuration Register 1215
38.2.2.3 Error Configuration 1216
38.2.2.4 Debug Status Port Register 1217
38.2.2.5 Password Comparison Registers 1218
38.3 Functional description 1219
38.4 Initialization/application information 1220
38.4.1 Reset 1220

Chapter 39
System Timer Module (STM)

39.1 Introduction 1221
39.1.1 Overview 1221
39.1.2 Features 1221
39.1.3 Modes of operation 1221
39.2 External signal description 1221

39.3 Memory map and register definition 1221
39.3.1 Memory map 1221
39.3.2 Register descriptions 1222
39.3.2.1 STM Control Register (STM_CR) 1222
39.3.2.2 STM Count Register (STM_CNT) 1223
39.3.2.3 STM Channel Control Register (STM_CCRn) 1224
39.3.2.4 STM Channel Interrupt Register (STM_CIRn) 1224
39.3.2.5 STM Channel Compare Register (STM_CMPn) 1225
39.4 Functional description 1225

Chapter 40
Voltage Regulators and Power Supplies

40.1 Introduction 1227
40.2 Voltage regulators 1227
40.2.1 Block diagram 1228
40.2.2 External signals 1228
40.2.3 Detailed signal descriptions 1229
40.2.3.1 VDDR 1229
40.2.3.2 VRC 1229
40.3 Memory map and register definition 1229
40.3.1 Voltage Regulator Control Register (VREG_CTL) 1229
40.4 Functional description 1230
40.4.1 High Power or Main Regulator (HPREG) 1230
40.4.2 Low-power Regulator (LPREG) 1230
40.4.3 Ultra Low-power Regulator (ULPREG) 1230
40.4.4 Low Voltage Detectors (LVD) and Power On Reset (POR) 1230
40.4.5 VREG digital interface 1231
40.5 GPIO power supply configuration 1232
40.6 Power domain organization 1233

Chapter 41
Wakeup Unit (WKPU)

41.1 Overview 1237
41.2 Features 1238
41.3 External signal description 1239
41.4 Memory map and register description 1239
41.4.1 Memory map 1239
41.4.2 Register description 1240
41.4.2.1 NMI Status Flag Register (NSR) 1240
41.4.2.2 NMI Configuration Register (NCR) 1241
41.4.2.3 Wakeup/Interrupt Status Flag Register (WISR) 1242
41.4.2.4 Interrupt Request Enable Register (IRER) 1243
41.4.2.5 Wakeup Request Enable Register (WRER) 1243
41.4.2.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER) . . 1243

41.4.2.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER) . . 1244
41.4.2.8 Wakeup/Interrupt Filter Enable Register (WIFER) 1244
41.4.2.9 Wakeup/Interrupt Pullup Enable Register (WIPUER) 1245
41.5 Functional description 1245
41.5.1 General 1245
41.5.2 Non-Maskable Interrupts 1245
41.5.2.1 NMI Management 1246
41.5.3 External Wakeups/Interrupts 1247
41.5.3.1 External Interrupt Management 1248
41.5.4 On-Chip Wakeups 1249
41.5.4.1 On-Chip Wakeup Management 1249

Appendix A
Registers Under Protection

Appendix B
Register Map

Appendix C
Revision History

C.1 Changes between revisions 6 and 7 1323
C.2 Changes between revisions 5 and 6 1325
C.3 Changes between revisions 4 and 5 1329
C.4 Changes between revisions 3 and 4 1330
C.5 Changes between revisions 2 and 3 1335
C.6 Changes between revisions 1 and 2 1338
C.7 Changes between revisions 0 and 1 1342

Preface

Overview

The primary objective of this document is to define the functionality of the MPC5606S microcontroller for use by software and hardware developers. The MPC5606S is built on Power Architecture® technology and integrates technologies that are important for today's instrument cluster applications.

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

To locate any published errata or updates for this document, visit the Freescale web site at www.freescale.com.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5606S device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

Chapter organization and device-specific information

This document includes chapters that describe:

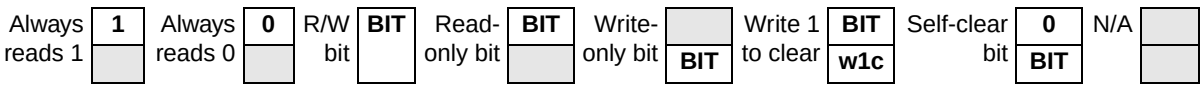
- The device as a whole
- The functionality of the individual modules on the device

In the latter, any device-specific information may be presented in the section "Information Specific to This Device" at the beginning of the chapter.

Key to register fields

Figure 0-1 provides a key for register figures and tables and the register summary,

Figure 0-1. Key to register fields



The conventions in Table 0-1 serve as a key for the register summary and individual register diagrams.

Table 0-1. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register field types	
R	Read only. Writing this bit has no effect
W	Write only
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that can be modified by hardware in some fashion other than by a reset
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
S	Set: Pattern on the data bus is ORed with and written into the register.
C	Clear: Pattern on the data bus is a mask. If a bit on the mask is set, then the corresponding register bit is cleared.
Reset values	
0	Resets to zero
1	Resets to one
—	Undefined at reset
u	Unaffected by reset
[*signal_name*]	Reset value is determined by polarity of indicated signal.

References

In addition to this reference manual, the following documents provide additional information on the operation of the MPC5606S:

- *IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)*
- *IEEE 1149.1-2001 standard—IEEE Standard Test Access Port and Boundary-Scan Architecture*
- *Power Architecture (http://www.freescale.com/files/32bit/doc/user_guide/BOOK_EUM.pdf)*

Chapter 1
Overview

1.1 Introduction

The MPC5606S family represents a new generation of 32-bit microcontrollers based on the Power Architecture®. It belongs to an expanding family of automotive-focused products targeted to address the next wave of instrument cluster applications by addressing the significant growth in color Thin Film Transistor (TFT) displays within the vehicle. The product architecture is designed to fulfill the system requirements of selected implementations on a single-chip solution by driving the TFT display directly. The memory footprint can be further expanded via the on-chip QuadSPI serial flash controller module.

The advanced and cost-efficient host processor core of the family complies with the Power Architecture embedded category, which is 100% user-mode compatible with the original PowerPC user instruction set architecture (UISA). It offers high performance processing optimized for low-power consumption, operating at speeds up to 64 MHz. The family itself offers a fully scalable solution from 256 KB up to 1 MB internal flash memory, and capitalizes on the available development infrastructure of current Power Architecture devices. It offers full support from available software drivers, operating systems, and configuration code to assist with users' implementations. Refer to Section 1.6, Developer environment, for more information.

The MPC5606S platform has a single level of memory hierarchy and supports a wide range of on-chip SRAM and internal flash memories. The 1 MB flash memory version (MPC5606S) outlined in detail within this document features 160 KB of on-chip graphics SRAM to buffer cost-effective color TFT displays driven via the on-chip Display Control Unit (DCU). Refer to Table 1-1, Table 1-2, and Table 1-3 for specific memory and feature sets of the product family members.

The MPC5606S microcontroller is designed to reduce development and production costs of TFT-based instrument cluster displays by providing a single-chip solution with the processing and storage capacity to host and execute real-time application software and drive the TFT display directly. Operating at speeds of up to 64 MHz, it offers high performance processing with low-power consumption. Memory and storage capacity can be further expanded via the on-chip Serial Peripheral Interface (SPI) and QuadSPI peripheral modules.

On-chip modules include:

- Single issue, 32-bit Power Architecture e200z0h CPU core complex
  - Compliant with PowerPC Book E instruction set architecture
  - Compatible with classic PowerPC instruction set
  - Includes Freescale Variable Length Encoding (VLE) enhancements for code size reduction
- On-chip ECC flash memory with flash controller
  - 1 MB primary flash—two 512 KB modules with prefetch buffer and 128-bit data access port
  - 64 KB data flash—separate 4 × 16 KB flash block for EEPROM Emulation with prefetch buffer and 128-bit data access port
- Up to 48 KB on-chip ECC RAM with SRAM controller

- 160 KB on-chip non-ECC graphics SRAM with RAM controller
- A Memory Protection Unit (MPU) provides basic memory access permission and ensures separation between different codes and data.
- A Boot Assist Module (BAM) with 8 KB dedicated ROM for embedded boot code
- Two frequency-modulated phase-locked loop (FMPLL) modules
- Interrupt controller (INTC) with 126 peripheral interrupt sources and eight software interrupts
- A 10-bit analog-to-digital converter (ADC) with a minimum conversion time of 1 µs
  - 16 internal channels
  - Up to 8 external channels
- Two enhanced Modular Input Output System (eMIOS) provide ability to generate or measure time events
  - One 8-channel, one 16-channel
  - 16-bit counter width
  - Configurable—can implement:
    - Up to 24 IC/OC channels
    - Up to 16 pulse width modulation (PWM) channels
    - Up to 5 modulus counters
- Two serial communication interface (LINFlex) modules:
  - LINFlex 0 and 1 are master-capable
  - LINFlex 0 is also slave-capable
- Two DSPI (Deserial Serial Peripheral Interface) modules for communications with external devices
- QuadSPI module
- Four Inter-integrated circuit (I^2C) internal bus controllers with master/slave bus interface
- Up to two enhanced full CAN (FlexCAN) modules with 64 configurable message buffers
- System timers:
  - One System Timer Module (STM)—included in processor platform
  - Four Periodic Interrupt Timer (PIT) timers (including ADC trigger)
  - One Real Time Counter (RTC) timer for timekeeping applications.
- Crossbar switch architecture for concurrent access to peripherals, flash, or RAM from multiple bus masters
- System Integration Unit Lite (SIUL) manages resets, external interrupts, GPIO, and pad control
  - 14 external interrupts
  - Supports up to 133 GPIO pads
  - 32-bit data bus width
- System Status and Configuration Module (SSCM)
  - Provides information for identification of the device, last boot mode, or debug status
  - Provides an entry point for censorship password mechanism

- Clock Generation Module (MC_CGM)
  - Generates system clock sources
  - Provides a unified register interface, enabling access to all clock sources.
- Clock Monitor Unit (CMU)
  - Monitors the integrity of the main crystal oscillator and the PLL
  - Acts as a frequency meter, measuring the frequency of one clock source vs. a reference clock
- Mode Entry Module (MC_ME) controls the System-on-Chip (SoC) mode, that is, Run, Halt, Stop, or Standby, and mode transition sequences. Also manages power control, voltage regulator, clock generation and clock management modules.
- Power Control Unit (MC_PCU) implements Standby mode entry/exit and controls connections to power domains
- Reset Generation Module (MC_RGM) manages reset assertion and release to the device at initial startup
- Sound generation module supports monophonic and polyphonic sound
- LCD controller module
- Six gauge drivers with Stepper Stall Detect (SSD)
- Display Control Unit (DCU) designed to interface with TFT LCD displays. Generates all signals required to drive the display and offers blending of four-plane data of up to 16 layers. Also has the capability of displaying real time digital video in the background plane.
- Second generation Direct Memory Access (DMA2) controller provides fast data transfer with minimal intervention from host processor
- On-chip voltage regulator controller for regulating 5 V down to 1.2 V for core logic
- JTAG (IEEE 1149.1) Nexus port for application development with optional Nexus 2+ trace port.
- The MPC5606S microcontroller is offered in the following packages:
  - 144-pin LQFP, 0.5mm pitch, 20 mm × 20 mm outline
  - 176-pin LQFP, 0.5mm pitch, 24 mm × 24 mm outline
  - 208-ball MAPBGA, 1 mm ball pitch, 17 mm × 17 mm outline (development purposes only; not a volume production package)

1.2 MPC5606S family comparison

Table 1-1 and Table 1-2 report the memory scaling of code flash memory and RAM.

Table 1-1. Code flash memory scaling

Memory Size	Start Address	End Address
256 KB	0x00000000	0x0003FFFF
512 KB	0x00000000	0x0007FFFF
1 MB	0x00000000	0x000FFFFF

Table 1-2. RAM memory scaling

Memory Size	Start Address	End Address
24 KB	0x40000000	0x40005FFF
48 KB	0x40000000	0x4000BFFF

Table 1-3 provides a summary of the different members of the MPC5606S family. This information is intended to provide an understanding of the range of functionality offered by this family.

1.3 Block diagram

Figure 1-1 shows a top-level block diagram of the MPC5606S.

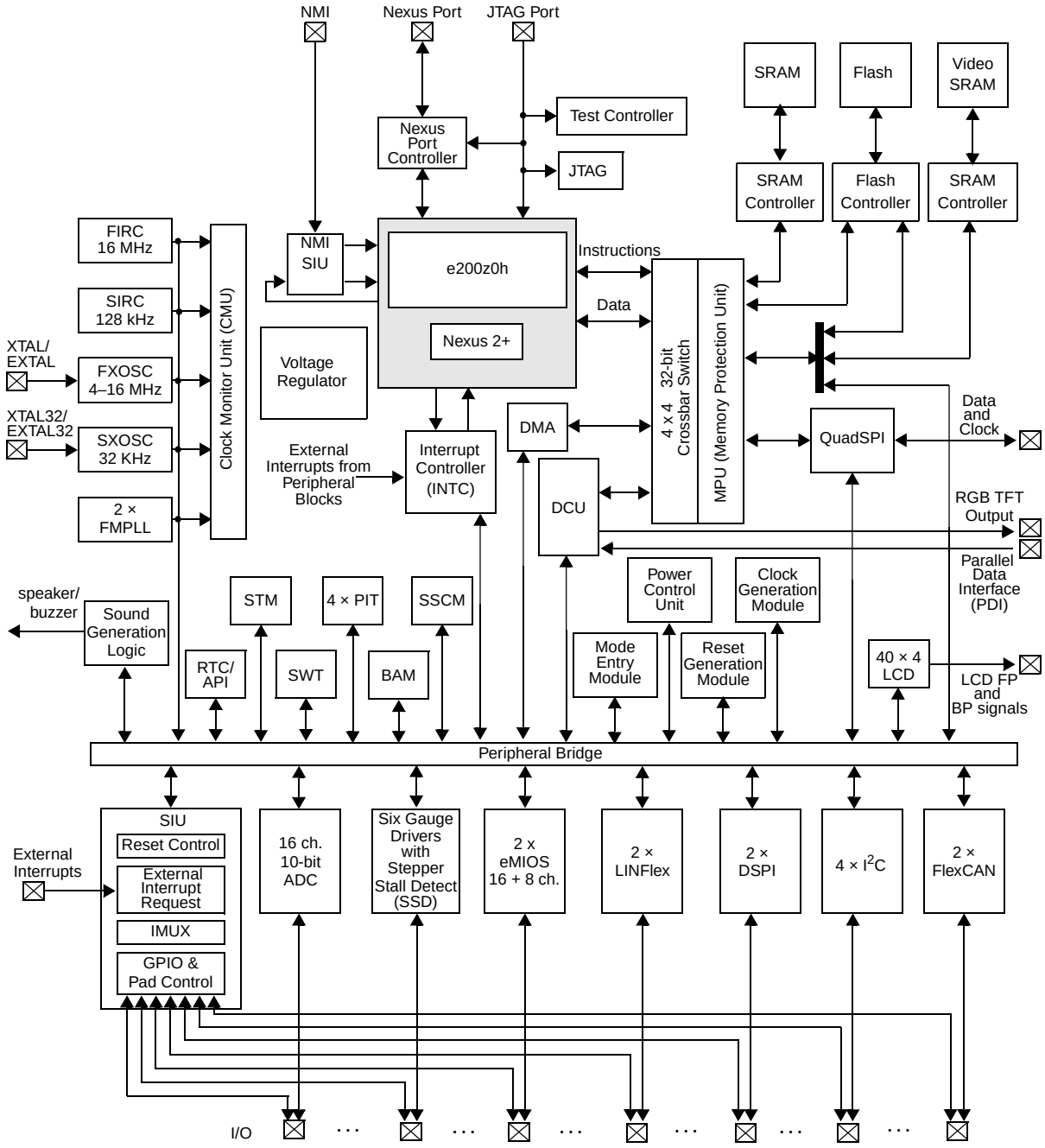


Figure 1-1. MPC5606S block diagram

1.4 Chip-level features

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

- Single issue, 32-bit Power Architecture Book E compliant CPU core complex (e200z0h)
  - Compatible with classic PowerPC instruction set
  - Includes variable length encoding (VLE) instruction set for smaller code size footprint; with the encoding of mixed 16-bit and 32-bit instructions, it is possible to achieve significant code size footprint reduction over conventional Book E compliant code
- On-chip ECC flash memory with flash controller
  - Up to 1 MB primary flash—two 512 KB modules with prefetch buffer and 128-bit data access port
  - 64 KB data flash—separate 4 × 16 KB flash block for EEPROM emulation with prefetch buffer and 128-bit data access port
- Up to 48 KB on-chip ECC SRAM with SRAM controller
- Up to 160 KB on-chip non-ECC graphics SRAM with SRAM controller
- Memory Protection Unit (MPU) with up to 12 region descriptors and 32-byte region granularity to provide basic memory access permission
- Interrupt Controller (INTC) with up to 127 peripheral interrupt sources and eight software interrupts
- 2 Frequency-Modulated Phase-Locked Loops (FMPLLs)
  - Primary FMPLL provides a 64 MHz system clock
  - Auxiliary FMPLL is available for use as an alternate, modulated or non-modulated clock source to eMIOS modules and as alternate clock to the DCU for pixel clock generation
- Crossbar switch architecture enables concurrent access of peripherals, flash memory, or RAM from multiple bus masters (AMBA 2.0 v6 AHB)
- 16-channel Enhanced Direct Memory Access controller (eDMA) with multiple transfer request sources using a DMA channel multiplexer
- Boot Assist Module (BAM) supports internal flash programming via a serial link (FlexCAN or LINFlex)
- Display Control Unit to drive TFT LCD displays
  - Includes processing of up to four planes that can be blended together
  - Offers a direct unbuffered hardware bit-blitter of up to 16 software-configurable dynamic layers in order to drastically minimize graphic memory requirements and provide fast animations
  - Programmable display resolutions are available up to WVGA
- Parallel Data Interface (PDI) for digital video input
- LCD segment driver module with two software programmable configurations:
  - Up to 40 frontplane drivers and 4 backplane drivers
  - Up to 38 frontplane drivers and 6 backplane drivers
- Stepper Motor Controller (SMC) module with high-current drivers for up to six instrument cluster gauges driven in full dual H-Bridge configuration including full diagnostics for short circuit detection
- Stepper motor return-to-zero and stall detection module

- Sound generation and playback utilizing PWM channels and eDMA; supports monotonic and polyphonic sound
- 24 eMIOS channels providing up to 16 PWM and 24 input capture / output compare channels
- 10-bit Analog-to-Digital Converter (ADC)
  - Maximum conversion time of 1 µs
  - Up to 16 internal channels, expandable to 23 via external multiplexing
- Up to 2 Deserial Serial Peripheral Interface (DSPI) modules for full-duplex, synchronous communications with external devices (extendable to include up to 8 multiplexed external channels)
- QuadSPI serial flash memory controller supporting single, dual, and quad modes of operation to interface to external serial flash memory; QuadSPI can be configured to function as another DSPI module (MPC5606S only)
- 2 Local Interconnect Network Flexible (LINFlex) controller modules capable of autonomous message handling (master), autonomous header handling (slave mode), and UART support; compliant with LIN protocol rev 2.1
- 2 full CAN 2.0B controllers with 64 configurable buffers each; bit rate programmable up to 1 Mbit/s
- Up to 4 Inter-integrated circuit (I^2C) internal bus controllers with master/slave bus interface
- Up to 133 configurable general purpose pins supporting input and output operations
- Real Time Counter (RTC) with multiple clock sources:
  - 128 kHz slow internal RC oscillator or 16 MHz fast internal RC oscillator supporting autonomous wakeup with 1 ms resolution with maximum timeout of 2 seconds
  - 32 kHz slow external crystal oscillator, supporting wakeup with 1 s resolution and maximum timeout of one hour
  - 4–16 MHz fast external crystal oscillator
- System timers:
  - 4-channel 32-bit System Timer Module (STM)—included in processor platform
  - 4-channel 32-bit Periodic Interrupt Timer (PIT) module
  - Software Watchdog Timer (SWT)
- System Integration Unit (SIU) module to manage resets, external interrupts, GPIO, and pad control
- System Status and Configuration Module (SSCM) to provide information for identification of the device, last boot mode, or debug status, and provides an entry point for the censorship password mechanism
- Clock Generation Module (MC_CGM) to generate system clock sources and provide a unified register interface, enabling access to all clock sources
- Clock Monitor Unit (CMU) to monitor the integrity of the main crystal oscillator and the PLL and act as a frequency meter, measuring the frequency of one clock source and comparing it to a reference clock

- Mode Entry Module (MC_ME) to control the device power mode, in other words, Run, Halt, Stop, or Standby control mode transition sequences, and manage the power control, voltage regulator, clock generation, and clock management modules
- Reset Generation Module (MC_RGM) to manage reset assertion and release to the device at initial startup
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 Class Two Plus standard
- Device/board boundary-scan testing supported per Joint Test Action Group (JTAG) of IEEE (IEEE 1149.1)
- On-chip voltage regulator controller for regulating the 3.3 or 5 V supply voltage down to 1.2 V for core logic (requires external ballast transistor)
- The MPC5606S microcontrollers are offered in the following packages:[1]
  - 144 LQFP, 0.5 mm pitch, 20 mm × 20 mm outline
  - 176 LQFP, 0.5 mm pitch, 24 mm × 24 mm outline
  - 208 MAPBGA, 1.0 mm pitch, 17 mm × 17 mm outline (not a production package; available in limited quantities for tool development only)

1.5 Feature details

1.5.1 Low-power operation

MPC5606S devices are designed for optimized low-power operation and dynamic power management of the core processor and peripherals. Power management features include software-controlled clock gating of peripherals and multiple power domains to minimize leakage in low-power modes.

There are two static low-power modes, Standby and Stop, and six dynamic power modes—five Run modes and Halt. Both low-power modes use clock gating to halt the clock for all or part of the device. Standby mode also uses power gating to automatically turn off the power supply to parts of the device to minimize leakage.

Standby mode turns off the power to the majority of the chip to offer the lowest power consumption mode. The contents of the cores, on-chip peripheral registers, and potentially some of the volatile memory are lost. Standby mode is configurable to make certain features available, with the disadvantage that these consume additional current:

- It is possible to retain the contents of the full RAM or only 8 KB.
- It is possible to enable the internal 16 MHz or 128 kHz RC oscillator, the external 4–16 MHz oscillator, or the external 32 KHz oscillator.
- It is possible to keep the LCD module active.

The device can be awakened from Standby mode from any of as many as 19 I/O pins, from a reset, or from a periodic wakeup using a low-power oscillator.

1. See the device comparison table or orderable parts summary for package offerings for each device in the family.

Stop mode maintains power to the entire device, thus allowing the retention of all on-chip registers and memory, and providing a faster recovery low-power mode than the lowest Standby mode. There is no need to reconfigure the device before executing code. The clocks to the core and peripherals are halted and can be optionally stopped to the oscillator or PLL at the expense of a slower startup time.

Stop mode is entered from Run mode only. Wakeup from Stop mode is triggered by an external event or by the internal periodic wakeup, if enabled.

Run modes are the primary operating modes where the entire device can be powered and clocked. In Run modes most processing activity is done. One default (Drun) and four dynamic Run modes are supported—Run0...3. The ability to configure and select different Run modes enables different clocks and power configurations to be supported with respect to each other, and to allow switching between different operating conditions. The necessary peripherals, clock sources, clock speed, and system clock prescalers can be independently configured for each of the four Run modes of the device.

Halt mode is a reduced activity, low-power mode intended for moderate periods of lower processing activity. In this mode the core system clocks are stopped but user-selected peripheral tasks can continue to run. It can be configured to provide more efficient power management features (switch-off PLL, flash memory, main regulator, etc.) at the cost of longer wakeup latency. The system returns to a Run mode as soon as an event or interrupt is pending.

Table 1-3 summarizes the operating modes of MPC5606S devices.

Table 1-3. Operating mode summary[1]

	Operating modes:	Run	Halt	Stop	Standby		POR
SoC features	Core	On	CG	CG	Off	Off	—
	Peripherals	OP	OP	CG	Off[2]	Off	—
	Flash memory	OP	OP	CG	Off	Off	—
	SRAM	On	On	CG	CG[3]	8 KB[4]	—
	Graphics RAM	On	On	CG	Off	Off	—
Clock sources	Main PLL	OP	OP	CG	Off	Off	—
	Auxiliary PLL	OP	OP	CG	Off	Off	—
	16 MHz IRC	On	On	OP	OP	OP	—
	FXOSC	OP	OP	OP	OP	OP	—
	128 kHz IRC	On	On	On	On	On	—
	32 KHz XOSC	OP	OP	OP	OP	OP	—
Periodic wakeup		—	OP	OP	OP	OP	—
Wakeup input		—	OP	OP	OP	OP	—
VREG mode		FP	FP	LP	LP	LP	—
Wakeup times[5]	VREG startup	—	—	50 µs	250 µs	250 µs	250 µs[6]
	IRC wakeup	—	—	4 µs	4 µs	8 µs	8 µs
	Flash memory recovery	—	—	20 µs	100 µs	100 µs	100 µs
	OSC stabilization	—	—	1 ms	1 ms	1 ms	1 ms
	PLL lock	—	—	200 µs	200 µs	200 µs	200 µs
	S/W reconfig	—	—	—	Var	Var	—
	Mode switch over	—	200.69 µs	24 µs	28 µs	28 µs	BAM

1 Table Key:
On—Powered and clocked
OP—Optionally configurable to be enabled or disabled (clock gated)
CG—Clock Gated, Powered but clock stopped
Off—Powered off and clock gated
FP—VREG Full Performance mode
LP—VREG low-power mode, reduced output capability of VREG but lower power consumption
Var—Variable duration, based on the required reconfiguration and execution clock speed
BAM—Boot Assist Module Software and Hardware used for device startup and configuration

2 The LCD can optionally be kept running while the device is in Standby mode.

3 All of the RAM content is retained, but not accessible in Standby mode.

4 8 KB of the RAM content is retained, but not accessible in Standby mode.

5 A high level summary of some key durations that need to be considered when recovering from low-power modes. This does not account for all durations at wakeup. Other delays will be necessary to consider, including but not limited to the external supply startup time.
IRC wakeup time must not be added to the overall wakeup time as it starts in parallel with the VREG.
All other wakeup times must be added to determine the total startup time.

6 This is the startup of the regulator that happens after the 5 V has reached beyond its POR range. If the external supply ramp rate is slow, measure from when VREG has crossed beyond the POR threshold; otherwise, this value will depend on the ramp rate of the external supply (VDDR).

Additional notes on low-power operation:

- Fast wakeup using the on-chip 16 MHz internal RC oscillator allows rapid execution from RAM on exit from low-power modes
- The 16 MHz internal RC oscillator supports low-speed code execution and clocking of peripherals when it is selected as the system clock and can also be used as the PLL input clock source to provide fast startup, without external oscillator delay
- MPC5606S devices include an internal voltage regulator that includes the following features:
  - Regulates input to generate all internal supplies
  - Manages power gating
  - Low-power regulators support operation when in Stop and Standby modes to minimize power consumption
  - Startup on-chip regulators in <50 µs for rapid exit of Stop and Standby modes
  - Low-voltage detection on main supply and 1.2 V regulated supplies

1.5.2 e200z0h core processor

The e200z0h processor is similar to other processors in the e200zx series, but supports only the VLE instruction set and does not include the signal processing extension for DSP applications or a floating point unit.

The e200z0h has all the features of the e200z0 plus:

- Branch acceleration using Branch Target Buffer (BTB)
- Supports independent instruction and data accesses to different memory subsystems, such as SRAM and flash memory via independent Instruction and Data BIUs

The e200z0h processor uses a four stage in-order pipeline for instruction execution.

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

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

The integer execution unit consists of:

- 32-bit Arithmetic Unit (AU)

- Logic Unit (LU)
- 32-bit Barrel shifter (Shifter)
- Mask-Insertion Unit (MIU)
- Condition Register manipulation Unit (CRU)
- Count-Leading-Zeros unit (CLZ)
- 8 × 32 hardware multiplier array
- Result feed-forward hardware
- Hardware divider

Most arithmetic and logical operations are executed in a single cycle with the exception of the divide and multiply instructions. A Count-Leading-Zeros unit operates in a single clock cycle. The Instruction Unit contains a PC incrementer and a dedicated Branch Address adder to minimize delays during change of flow operations. Branch target prefetching from the BTB is performed to accelerate certain taken branches. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Prefetched instructions are placed into an instruction buffer capable of holding four instructions.

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

Memory load and store operations are provided for byte, halfword, and word (32-bit) data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single-cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized. Also, a load-to-use dependency does not incur any pipeline bubbles for most cases.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power Architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as:

- Move
- Integer and floating-point compare
- Arithmetic
- Logical instructions

and provide a mechanism for testing and branching.

Vectored and autovectored interrupts are supported. Hardware-vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The CPU includes support for Variable Length Encoding (VLE) instruction enhancements. This allows the Power Architecture instruction set to be represented by a modified instruction set made up from a mixture of 16-bit and 32-bit instructions. This results in a significantly smaller code size footprint without affecting performance noticeably.

The CPU core is enhanced by an additional interrupt source, the Non-Maskable Interrupt (NMI). This interrupt source is routed directly from package pins, via edge detection logic in the SIU to the CPU,

bypassing the interrupt controller completely. Once the edge detection logic is programmed, it cannot be disabled, except by reset. The NMI is, as the name suggests, completely un-maskable and when asserted will always result in the immediate execution of the respective interrupt service routine. The NMI is not guaranteed to be recoverable.

The CPU core has an additional Wait for Interrupt instruction that is used in conjunction with low-power Stop mode. When Low-power Stop mode is selected, this instruction is executed to allow the system clock to be stopped. An external interrupt source or the system wakeup timer is used to restart the system clock and allow the CPU to service the interrupt.

Additional features include:

- Load/store unit
  - 1-cycle load latency
  - Misaligned access support
  - No load-to-use pipeline bubbles
- Thirty-two 32-bit general purpose registers (GPRs)
- Separate instruction bus and load/store bus Harvard architecture
- Reservation instructions for implementing read-modify-write constructs
- Multi-cycle divide (divw) and load multiple (lmw) store multiple (smw) multiple class instructions; can be interrupted to prevent increases in interrupt latency
- Extensive system development support through Nexus debug port

1.5.3 Crossbar switch (XBAR)

The XBAR multi-port crossbar switch supports simultaneous connections between four master ports and four slave ports. The crossbar supports a 32-bit address bus width and a 32-bit data bus width.

The crossbar allows four concurrent transactions to occur from any master port to any slave port, but one of those transfers must be an instruction fetch from internal flash. If a slave port is simultaneously requested by more than one master port, arbitration logic selects the higher priority master and grants it ownership of the slave port. All other masters requesting that slave port are stalled until the higher priority master completes its transactions. Requesting masters having equal priority are granted access to a slave port in round-robin fashion, based upon the ID of the last master to be granted access.

The crossbar provides the following features:

- Four master ports:
  - e200z0h core instruction port
  - e200z0h core complex load/store data port
  - eDMA controller
  - Display control unit
- Four slave ports:
  - One flash port dedicated to the CPU
  - Platform SRAM

  - QuadSPI serial flash controller
  - One slave port combining:
    - Flash port dedicated to the Display Control Unit and eDMA module
    - Graphics SRAM
    - Peripheral bridge
- 32-bit internal address bus, 32-bit internal data bus

1.5.4 Enhanced Direct Memory Access (eDMA)

The eDMA module is a controller capable of performing complex data movements via 16 programmable channels, with minimal intervention from the host processor. The hardware micro architecture includes a DMA engine, that performs source and destination address calculations, and the actual data movement operations, along with an SRAM-based memory containing the transfer control descriptors (TCD) for the channels. This implementation is utilized to minimize the overall block size. The eDMA module provides the following features:

- 16 channels support independent 8-, 16-, or 32-bit single value or block transfers.
- Supports variable-sized queues and circular queues.
- Source and destination address registers are independently configured to post-increment or remain constant.
- Each transfer is initiated by a peripheral, CPU, periodic timer interrupt, or eDMA channel request.
- Each DMA channel can optionally send an interrupt request to the CPU on completion of a single value or block transfer.
- DMA transfers possible between system memories, QuadSPI, DSPIs, I^2C, ADC, eMIOS, and General Purpose I/Os (GPIOs).
- Programmable DMA Channel Mux allows assignment of any DMA source to any available DMA channel with as many as 64 potential request sources.

1.5.5 Inter-IC communications module (I^2C)

The I^2C module features the following:

- As many as four I^2C modules supported
- Two-wire bi-directional serial bus for on-board communications
- Compatibility with I^2C bus standard
- Multimaster operation
- Software-programmable for one of 256 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection

- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

1.5.6 Interrupt Controller (INTC)

The INTC provides priority-based preemptive scheduling of interrupt requests, suitable for statically scheduled hard real-time systems.

For high-priority interrupt requests, the time from the assertion of the interrupt request from the peripheral to when the processor is executing the interrupt service routine (ISR) has been minimized. The INTC provides a unique vector for each interrupt request source for quick determination of which ISR needs to be executed. It also provides an ample number of priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. To allow the appropriate priorities for each source of interrupt request, the priority of each interrupt request is software-configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the priority ceiling protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource cannot preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests also can be used to break the work involved in servicing an interrupt request into a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR asserts a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of the peripheral ISR scheduling a task through the RTOS. The INTC provides the following features:

- Unique 9-bit vector for each of the possible 128 separate interrupt sources
- Eight software triggerable interrupt sources
- 16 priority levels with fixed hardware arbitration within priority levels for each interrupt source
- Ability to modify the ISR or task priority
  - Modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
- External NMI directly accessing the main core critical interrupt mechanism
- 32 external interrupts

1.5.7 QuadSPI serial flash controller

The QuadSPI module enables use of external serial flash memories supporting single, dual, and quad modes of operation. It features the following:

- Memory mapping of external serial flash memory
- Automatic serial flash read command generation by CPU, DMA, or DCU read access on AHB bus
- Supports single, dual, and quad serial flash read commands
- Flexible buffering scheme to maximize read bandwidth of serial flash

- Legacy mode allowing QuadSPI to be used as a standard DSPI (no DSI or CSI mode)

1.5.8 System Integration Unit (SIU)

The SIU controls MCU, pad configuration, external interrupt, general purpose I/O (GPIO) and internal peripheral multiplexing.

The GPIO features the following:

- As many as four levels of internal pin multiplexing, allowing exceptional flexibility in the allocation of device functions for each package
- Centralized general purpose input output (GPIO) control of as many as 132 input/output pins (package dependent)
- All GPIO pins can be independently configured to support pullup, pulldown, or no pull
- Reading and writing to GPIO supported both as individual pins and 16-bit-wide ports
- All peripheral pins can be alternatively configured as both general purpose input or output pins, except ADC channels which support alternative configuration as general purpose inputs
- Direct readback of the pin value supported on all digital output pins through the SIU
- Configurable digital input filter that can be applied to as many as 14 general purpose input pins for noise elimination on external interrupts
- Register configuration protected against change with soft lock for temporary guard or hard lock to prevent modification until next reset

1.5.9 Flash memory

The MPC5606S microcontroller has the following flash memory features:

- As nuch as 1 MB of burst flash memory
  - Typical flash memory access time: 0 wait state for buffer hits, 2 wait states for page buffer miss at 64 MHz
  - Two 4×128-bit page buffers with programmable prefetch control
    - One set of page buffers can be allocated for code-only, fixed partitions of code and data, all available for any access
    - One set of page buffers allocated to Display Controller Unit and the eDMA
  - 64-bit ECC with single-bit correction, double-bit detection for data integrity
  - 64 KB data flash memory — separate 4×16 KB flash block for EEPROM emulation with prefetch buffer and 128-bit data access port
- Small block flash memory arrangement to support features such as boot block, operating system block
- Hardware-managed flash memory writes, erases and verify sequences
- Censorship protection scheme to prevent flash memory content visibility
- Separate dedicated 64 KB data flash memory for EEPROM emulation
  - Four erase sectors each containing 16 KB of memory

— Offers Read-While-Write functionality from main program space
— Same data retention and program erase specification as main program flash memory array

1.5.10 SRAM

The MPC5606S microcontrollers have as much as 48 KB general-purpose on-chip SRAM with the following features:

- Typical SRAM access time: 0 wait-state for reads and 32-bit writes; 1 wait state for 8- and 16-bit writes if back to back with a read to same memory block
- 32-bit ECC with single-bit correction, double bit detection for data integrity
- Supports byte (8-bit), half word (16-bit), and word (32-bit) writes for optimal use of memory
- User transparent ECC encoding and decoding for byte, half word, and word accesses
- Separate internal power domain applied to full SRAM block, 8 KB SRAM block during Standby modes to retain contents during low-power mode.

1.5.11 On-chip graphics SRAM

The MPC5606S microcontroller has 160 KB on-chip graphics SRAM with the following features:

- Usable as general purpose SRAM
- Typical SRAM access time: 0 wait-state for reads and 32-bit writes
- Supports byte (8-bit), half word (16-bit), and word (32-bit) writes for optimal use of memory

1.5.12 Memory Protection Unit (MPU)

The MPU features the following:

- 12 region descriptors for per-master protection
- Start and end address defined with 32-byte granularity
- Overlapping regions supported
- Protection attributes can optionally include process ID
- Protection offered for three concurrent read ports
- Read and write attributes for all masters
- Execute and supervisor/user mode attributes for processor masters

1.5.13 Boot Assist Module (BAM)

The BAM is a block of read-only memory that is programmed once by Freescale. The BAM program is executed every time the MCU is started up or reset in normal mode. The BAM supports different modes of booting. They are:

- Booting from internal flash memory
- Serial boot loading (a program is downloaded into RAM via FlexCAN or LINFlex and then executed)

- Booting from external memory

Additionally the BAM:

- Enables and manages the transition of the MCU from reset to user code execution
- Configures device for serial bootload
- Enables multiple bootcode starting locations out of reset through implementation of search for valid Reset Configuration Halfword

1.5.14 Enhanced Modular Input/Output System (eMIOS)

MPC5606S microcontrollers have two eMIOS modules—one with 16 channels and one with eight—with input/output channels supporting a range of 16-bit input capture, output compare, and Pulse Width Modulation functions.

The modules are configurable and can implement 8-channel, 16-bit input capture/output compare or 16-channel, 16-bit output pulse width modulation/input compare/output compare. As many as five additional channels are configurable as modulus counters.

eMIOS other features include:

- Selectable clock source from main FMPLL, auxiliary FMPLL, external 4–16 MHz oscillator or 16 MHz internal RC oscillator
- Timed I/O channels with 16-bit counter resolution
- Buffered updates
- Support for shifted PWM outputs to minimize occurrence of concurrent edges
- Edge-aligned output pulse width modulation
  - Programmable pulse period and duty cycle
  - Supports 0% and 100% duty cycle
  - Shared or independent time bases
- Programmable phase shift between channels
- Selectable combination of pairs of eMIOS outputs to support sound generation
- DMA transfer support
- Selectable clock source from the primary FMPLL, auxiliary FMPLL, external 4–16 MHz oscillator, or the 16 MHz internal RC oscillator.

The channel configuration options for the 16-channel eMIOS module are summarized in Table 1-4.

Table 1-4. 16-Channel eMIOS module channel configuration

Channel mode	Channel number				
	8 IC/OC Counter	9–15 IC/OC	16 PWM Counter	17–22 PWM	23 PWM Counter
General Purpose Input/Output	X	X	X	X	X
Single Action Input Capture	X	X	X	X	X
Single Action Output Compare	X	X	X	X	X
Modulus Counter Buffered[1]	X		X		X
Output Pulse Width and Frequency Modulation Buffered			X	X	X
Output Pulse Width Modulation Buffered			X	X	X

1 Modulus up and down counters to support driving local and global counter buses.

The channel configuration options for the eight-channel eMIOS module are summarized in Table 1-5.

Table 1-5. Eight-channel eMIOS module channel configuration

Channel mode	Channel number		
	16 PWM Counter	17–22 PWM	23 PWM Counter
General Purpose Input/Output	X	X	X
Single Action Input Capture	X	X	X
Single Action Output Compare	X	X	X
Modulus Counter Buffered[1]	X		X
Output Pulse Width and Frequency Modulation Buffered	X	X	X
Output Pulse Width Modulation Buffered	X	X	X

1 Modulus up and down counters to support driving local and global counter buses.

1.5.15 Analog-to-Digital Converter (ADC)

The ADC features the following:

- 10-bit A/D resolution
- 0 to 5 V common mode conversion range
- Supports conversions speeds of up to 1 µs
- 16 internal and eight external channel support

- As many as 16 single-ended input channels
  - All channels configured to have alternate function as general purpose input/output pins
    - 10-bit ±3 counts accuracy (TUE)
- External multiplexer support to increase as many as 23 channels
  - Automatic 1 × 8 multiplexer control
  - External multiplexer connected to a dedicated input channel
  - Shared register between the eight external channels
- Result register available for every non-multiplexed channel
- Configurable left- or right-aligned result format
- Supports for one-shot, scan, and injection conversion modes
- Injection mode status bit implemented on adjacent 16-bit register for each result
  - Supports access to result and injection status with single 32-bit read
- Independent enabling of function for channels:
  - Offset refresh
- Conversion Triggering support
  - Internal conversion triggering from periodic interrupt timer (PIT)
- Four configurable analog comparator channels offering range comparison with triggered alarm
  - Greater than
  - Less than
  - Out of range
- All unused analog inputs can be used as general purpose input and output pins
- Power Down mode
- Optional support for DMA transfer of results

1.5.16 Deserial Serial Peripheral Interface (DSPI)

The deserial serial peripheral interface (DSPI) modules provide a synchronous serial interface for communication between the MPC5606S MCU and external devices.

The DSPI features the following:

- As many as two DSPI modules
- Full-duplex, synchronous transfers
- Master or slave operation
- Programmable master bit rates
- Programmable clock polarity and phase
- End-of-transmission interrupt flag
- Programmable transfer baud rate
- Programmable data frames from 4 to 16 bits

- As many as six chip select lines available, depending on package and pin multiplexing, enable 64 external devices to be selected using external muxing from a single DSPI
- Eight clock and transfer attributes registers
- Chip select strobe available as alternate function on one of the chip select pins for deglitching
- FIFOs for buffering as many as four transfers on the transmit and receive side
- General purpose I/O functionality on pins when not used for SPI
- Queueing operation possible through use of eDMA

1.5.17 FlexCAN

The MPC5606S MCU contains two controller area network (FlexCAN) modules. The FlexCAN module is a communication controller implementing the CAN protocol according to Bosch Specification version 2.0B. The CAN protocol was designed to be used primarily as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

The FlexCan modules offer the following:

- Compliant with CAN protocol specification, Version 2.0B active
- 64 mailboxes, each configurable as transmit or receive
  - Mailboxes configurable while module remains synchronized to CAN bus
- Transmit features
  - Supports configuration of multiple mailboxes to form message queues of scalable depth
  - Arbitration scheme according to message ID or message buffer number
  - Internal arbitration to guarantee no inner or outer priority inversion
  - Transmit abort procedure and notification
- Receive features
  - Individual programmable filters for each mailbox
  - Eight mailboxes configurable as a 6-entry receive FIFO
  - Eight programmable acceptance filters for receive FIFO
- Programmable clock source
  - System clock
  - Direct oscillator clock to avoid PLL jitter
- Listen-only mode capabilities
- CAN Sampler
  - Can catch the first message sent on the CAN network while the MPC5606S is stopped; this guarantees a clean startup of the system without missing messages on the CAN network
  - CAN sampler is connected to one of the CAN RX pins

1.5.18 Serial communication interface module (LINFlex)

The MPC5606S devices include as many as two LINFlex modules and support for LIN Master mode, LIN Slave mode, and UART mode. The modules are LIN state machine-compliant to the LIN 1.3 and 2.0 and 2.1 specifications and handle LIN frame transmission and reception without CPU intervention.

Other features include:

- Autonomous LIN frame handling
- Message buffer to store identifier and as many as 8 data bytes
- Supports message length as long as 64 bytes
- Detection and flagging of LIN errors
- Sync field, Delimiter, ID parity, Bit, Framing, Checksum, and Timeout errors
- Classic or extended checksum calculation
- Configurable break duration as long as 36-bit times
- Programmable baud rate prescalers (13-bit mantissa, 4-bit fractional)
- Diagnostic features
  - Loopback
  - Self-test
  - LIN bus stuck dominant detection
- Interrupt-driven operation with 16 interrupt sources
- LIN slave mode features
  - Autonomous LIN header handling
  - Autonomous LIN response handling
  - Discarding of irrelevant LIN responses using as many as 16 ID filters
- UART mode
  - Full-duplex operation
  - Standard non-return-to-zero (NRZ) mark/space format
  - Data buffers with 4-byte receive, 4-byte transmit
  - Configurable word length (8-bit or 9-bit words)
  - Error detection and flagging
    - Parity, noise, and framing errors
  - Interrupt driven operation with four interrupt sources
  - Separate transmitter and receiver CPU interrupt sources
  - 16-bit programmable baud-rate modulus counter and 16-bit fractional
  - Two receiver wakeup methods

1.5.19 System clocks and clock generation modules

The system clock on the MPC5606S can be derived from an external oscillator, an on-chip FMPLL, or the internal 16 MHz oscillator.

- Source system clock frequency can be changed via an on-chip programmable clock divider (÷1 to ÷32)
- Additional programmable peripheral bus clock divider ratio (÷1 to ÷16)
- Two on-chip FMPLLs—the primary module and an auxiliary module
  - Each FMPLL features:
    - Input clock frequency from 4 MHz to 16 MHz
    - Lock detect circuitry continuously monitoring lock status
    - Loss Of Clock (LOC) detection for reference and feedback clocks
    - On-chip loop filter (for improved electromagnetic interference performance and reduction of number of external components required)
    - Support for frequency ramping from PLL
  - The primary FMPLL module is for use as a system clock source; the auxiliary FMPLL is available for use as an alternate, modulated or non-modulated clock source to eMIOS modules and as alternate clock to the DCU for pixel clock generation
- The main oscillator provides the following features:
  - Input frequency range 4–16 MHz
  - Square-wave input mode
  - Oscillator input mode 3.3 V (5.0 V)
  - Automatic level control
  - PLL reference
- MPC5606S includes a 32 KHz low-power external oscillator for slow execution, reduced power consumption, and Real Time Clock
- Dedicated internal 128 kHz RC oscillator for low-power mode operation and self wakeup
  - ±10% accuracy across voltage and temperature (after factory trimming)
  - Trimming registers to support improved accuracy with in-application calibration
- Dedicated 16 MHz internal RC oscillator
  - Used as default clock source out of reset
  - Provides a clock for rapid startup from low-power modes
  - Provides a backup clock in the event of PLL or external oscillator clock failure
  - Offers an independent clock source for the watchdog timer
  - ±5% accuracy across voltage and temperature (after factory trimming)
  - Trimming registers to support frequency adjustment with in-application calibration

1.5.20 Periodic Interrupt Timer module (PIT)

The PIT features the following:

- Four general-purpose interrupt timers
- As many as two dedicated interrupt timers for triggering ADC conversions
- 32-bit counter resolution

- Clocked by system clock frequency
- 32-bit counter for Real Time Interrupt, clocked from main external oscillator

1.5.21 Real Time Counter (RTC)

The RTC supports wakeup from low-power modes or Real Time Clock generation

- Configurable resolution for different timeout periods
  - 1 s resolution for >1 hour period
  - 1 ms resolution for 2 second period
- Selectable clock sources from external 32 KHz crystal, external 4–16 MHz crystal, internal 128 kHz RC oscillator, or divided internal 16 MHz RC oscillator

1.5.22 System Timer Module (STM)

The STM is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

1.5.23 Software Watchdog Timer (SWT)

The Watchdog features the following:

- Watchdog can be activated by software or enabled out of reset
- Supports normal or windowed mode
- Watchdog timer value writable once after reset
- Configurable response on timeout: reset, interrupt, or interrupt followed by reset
- Selectable clock source for main system clock or internal 16 MHz RC oscillator clock

1.5.24 Display Control Unit (DCU)

The DCU is a display controller designed to drive TFT LCD displays capable of driving screens with resolution as high as Wide Quarter Video Graphics Array (WQVGA), with 16 layers and four planes with real time alpha-blending.

The DCU generates all the necessary signals required to drive the display: up to 24-bit RGB data bus, Pixel Clock, Data Enable, Horizontal-Sync and Vertical-Sync.

The internal memory resources of the Spectrum allow easy management of complex graphics contents (pictures, icons, languages, fonts) on a color TFT panel in up to WQVGA sizes. All the data fetches from

internal and/or external memory are performed by the internal four-channel DMA of the DCU providing a high speed/low latency access to the system backbone.

Control Descriptors (CDs) associated with each layer enable effective merging of different color formats into one plane to optimize use of internal memory buffers. A layer may be constructed from graphic content of various color formats including 1bpp, 2bpp, 4bpp, 8bpp, 16bpp, 24bpp, and 24bpp+alpha. The ability of the DCU to handle input data in formats as low as 1bpp, 2bpp, and 4bpp enables highly efficient use of internal memory resources of the MPC5606S. A special tiled mode can be enabled on any of the 16 layers to repeat a pattern, optimizing graphic memory usage.

A hardware cursor can be managed independently of the layers at blending level, increasing the efficient use of internal DCU resources.

To secure the content of all critical information to be displayed, a safety mode can be activated to check the integrity of critical data along the whole system data path from the memory to the TFT pads.

The DCU features the following:

- Display color depth: up to 24 bpp
- Generation of all RGB and control signals for TFT
- Four-layer blending at each pixel position
- Maximum number of input layers: 16 (fixed priority)
- Dynamic Look-Up Table (color and gamma look-up)
- α–blending range: up to 256 levels
- Transparency mode for font or single foreground color graphics
- Gamma correction
- Tiled mode on all the layers
- Hardware cursor
- Critical display content integrity monitoring for Functional Safety support
- Internal Direct Memory Access (DMA) module to transfer data from internal and/or external memory

1.5.25 Parallel Data Interface (PDI)

The PDI is a digital interface used to receive external digital video or graphic content into the DCU.

The PDI input is directly injected into the DCU background plane FIFO. When the PDI is activated, all the DCU synchronization is extracted from the external video stream to guarantee the synchronization of the two video sources.

The PDI can be used to:

- Connect a video camera output directly to the PDI
- Connect a secondary display driver as slave with a minimum of extra cost
- Connect a device gathering various video sources
- Provide flexibility to allow the DCU to be used in slave mode (external synchronization)

The PDI features the following:

- Supported color modes:
  - 8-bit mono
  - 8-bit color multiplexed
  - RGB565
  - 16-bit/18-bit RAW color
- Supported synchronization modes:
  - Embedded ITU-R BT.656-4 (RGB565 mode 2)
  - HSYNC, VSYNC
  - Data enable
- Direct interface with DCU background plane FIFO
- Synchronization generation for the DCU

1.5.26 Liquid Crystal Display (LCD) driver

The LCD driver module has two configurations allowing a maximum of 160 or 228 LCD segments:

- As many as 40 frontplane drivers and four backplane drivers
- As many as 38 frontplane drivers and six backplane drivers

Each segment is controlled and can be masked by a corresponding bit in the LCD RAM.

Four to six multiplex modes (1/1, 1/2, 1/3, 1/4, 1/5, 1/6 duty), and three bias (1/1, 1/2, 1/3) methods are available. All frontplane and backplane pins can be multiplexed with other port functions.

The LCD driver module features the following:

- Programmable frame clock generator from different clock sources:
  - System clock
  - Internal RC oscillator
- Programmable bias voltage level selector
- On-chip generation of all output voltage levels
  - LCD voltage reference taken from main 5 V supply
- LCD RAM – contains the data to be displayed on the LCD
  - Data can be read from or written to the display RAM at any time
- End-of-frame interrupt:
  - Optimize data refresh without visual artifacts
  - Selectable number of frames between each interrupt
- Contrast adjustment using programmable internal voltage reference
- Remapping capability of four or six backplanes with frontplanes
  - Increases pin selection flexibility

- In low-power modes, LCD operation can be suspended under software control; the LCD can also operate in low-power modes, clocked by the internal 128 kHz IRC or external 32 KHz crystal oscillator
- Selectable output current boost during transitions

1.5.27 Stepper Motor Controller (SMC)

The SMC module is a PWM motor controller suitable to drive instruments in a cluster configuration or any other loads requiring a PWM signal. The motor controller has twelve PWM channels associated with two pins each (24 pins in total).

The SMC module includes the following features:

- 10/11-bit PWM counter
- 11-bit resolution with selectable PWM dithering function
- Left-, right-, or center-aligned PWM
- Output slew rate control
- Output short-circuit detection

This module is suited for, but not limited to, driving small stepper and air core motors used in instrumentation applications. This module can be used for other motor control or PWM applications that match the frequency, resolution, and output drive capabilities of the module.

1.5.28 Stepper Stall Detect (SSD)

The stepper stall detector (SSD) module provides a circuit to measure and integrate the induced voltage on the non-driven coil of a stepper motor using full steps when the gauge pointer is returning to zero (RTZ).

The SSD module features the following:

- Programmable full step state
- Programmable integration polarity
- Blanking (recirculation) state
- 16-bit integration accumulator register
- 16-bit modulus down counter with interrupt

1.5.29 Sound Generation Logic (SGL)

The SGL has two modes of operation:

- Amplitude-modulated PWM mode for low-cost buzzers using any two eMIOS channels:
  - Monophonic signal with amplitude control
  - 8-bit amplitude resolution
  - Ability to mix any two eMIOS channels
  - Requires simple external RC lowpass filter
- Digital sample mode for higher quality sound using one eMIOS channel and eDMA

- Up to 10-bit audio amplitude resolution
- Polyphonic sound synthesis
- Playback of sample-based waveforms
- Text-to-speech possibility
- Requires external lowpass filter

1.5.30 IEEE 1149.1 JTAG Controller (JTAGC)

JTAGC features the following:

- Backward compatible to standard JTAG IEEE 1149.1-2001 test access port (TAP) interface
- Support for boundary scan testing

1.5.31 Nexus Development Interface (NDI)

Nexus features the following:

- Per IEEE-ISTO 5001-2003
- Nexus 2 Plus features supported
  - Static debug
  - Watchpoint messaging
  - Ownership trace messaging
  - Program trace messaging
  - Real time read/write of any internally memory-mapped resources through JTAG pins
  - Overrun control, which selects whether to stall before Nexus overruns or else keep executing and allow overwrite of information
  - Watchpoint triggering, watchpoint triggers program tracing
- Configured via the IEEE 1149.1 (JTAG) port
- Nexus Auxiliary port supported on the 176 LQFP and 208-pin BGA package FOR DEVELOPMENT ONLY
  - Narrow Auxiliary Nexus port supporting support trace, with two MDO pins
  - Wide Auxiliary Nexus port supporting higher bandwidth trace, with four MDO pins

1.6 Developer environment

The MPC5606S MCU family uses tools and third-party developers that offer a widespread, established network of tool and software vendors. It also features a high-performance Nexus debug interface.

The following development support is available:

- Automotive evaluation boards (EVB) featuring CAN, LIN interfaces, and more
- Compilers
- Debuggers

- JTAG and Nexus interfaces

The following software support is available:

- OSEK solutions will be available from multiple third parties
- CAN and LIN drivers
- AutoSAR package

1.7 How to use the MPC5606S documents

This section:

- Describes how the MPC5606S documents provide information on the microcontroller
- Makes recommendations on how to use the documents in a system design

1.7.1 The MPC5606S document set

The MPC5606S document set comprises:

- This reference manual (provides information on the features of the logical blocks on the device and how they are integrated with each other)
- The *MPC5606S Microcontroller Data Sheet* (specifies the electrical characteristics of the device)
- The device product brief

The following reference documents (available online at www.freescale.com) are also available to support the CPU on this device:

- *Programmer's Reference Manual for Freescale Embedded Processors*
- *e200z0 Power Architecture Core Reference Manual*
- *Variable-Length Encoding (VLE) Programming Environments Manual*

The aforementioned documents describe all of the functional and electrical characteristics of the MPC5606S microcontroller.

Depending on your task, you may need to refer to multiple documents to make design decisions. However, in general the use of the documents can be divided up as follows:

- Use the reference manual (this document) during software development and when allocating functions during system design.
- Use the data sheet when designing hardware and optimizing power consumption.
- Use the CPU reference documents when doing detailed software development in assembly language or debugging complex software interactions.

1.7.2 Reference manual content

The content in this document focuses on the functionality of the microcontroller rather than its performance. Most chapters describe the functionality of a particular on-chip module, such as a CAN controller or timer. The remaining chapters describe how these modules are integrated into the memory map, how they are powered and clocked, and the pinout of the device.

In general, when an individual module is enabled for use, all of the detail required to configure and operate it is contained in the dedicated chapter. In some cases there are multiple implementations of this module; however, there is only one chapter for each type of module in use. For this reason, the address of registers in each module is normally provided as an offset from a base address that can be found in Chapter 2, Memory Map. The benefit of this approach is that software developed for a particular module can be easily reused on this device and on other related devices that use the same modules.

The steps to enable a module for use vary, but typically require configuration of the integration features of the microcontroller. The module will normally have to be powered and enabled at the system level, then a clock may have to be explicitly chosen, and finally, if required, the input and output connections to the external system must be configured.

The primary integration chapters of the reference manual contain most of the information required to enable the modules. There are special cases where a chapter may describe module functionality and some integration features for convenience — for example, the microcontroller input/output (SIUL) module. Integration and functional content is provided in the manual as shown in Table 1-6.

Table 1-6. Reference manual integration and functional content

Chapter	Integration content	Functional content
Overview	• The main features on chip • A summary of the functions provided by each module	—
Memory Map	How the memory map is allocated, including: • Internal RAM • Flash memory • External memory-mapped resources and the location of the registers used by the peripherals[1]	—
Signal Description	How the signals from each of the modules are combined and brought to a particular pin on a package	—
Boot Assist Module	CPU boot sequence from reset	Implementation of the boot options if internal flash memory is not used
Clock Description	Clocking architecture of the device (which clock is available for the system and each peripheral)	Description of operation of different clock sources
DMA Channel Mux	Source values for module DMA channels	How to connect a module DMA channel to the eDMA module
Interrupt Controller	Interrupt vector table	Operation of the module
Mode Entry Module	Module numbering for control and status	Operation of operating modes
System Integration Unit Lite	How input signals are mapped to individual modules including external interrupt pins	Operation of GPIO
Voltage regulators and power supplies	Power distribution to the MCU and in particular to different I/O banks	—
Wakeup Unit	Allocation of inputs to the Wakeup Unit	Operation of the wakeup feature

[1] To find the address of a register in a particular module take the start address of the module given in the memory map and add the offset for the register given in the module chapter.

1.8 Using the MPC5606S

There are many different approaches to designing a system using the MPC5606S, so the guidance in this section is provided as an example of how the documents can be applied to this task.

Familiarity with the MPC5606S modules can help ensure that its features are being optimally used in a system design. Therefore, the current chapter is a good starting point. Further information on the detailed features of each module are provided within the module chapters. These, combined with the current chapter, should provide a good introduction to the functions available on the MCU.

1.8.1 Hardware design

The MPC5606S requires that certain pins be connected to particular power supplies, system functions, and other voltage levels for operation.

The MPC5606S internal logic operates from 1.2 V (nominal) supplies that are normally supplied by the on-chip voltage regulator from a 5 V or 3.3 V supply. The 5 V and 3.3 V supplies are also used to supply the input/output pins on the MCU. This means that different input/output ports can operate at different voltages simultaneously. Chapter 3, Signal Description, describes the power supply pin names, numbers, and their purposes. For more detail on the voltage supply of each pin, see Chapter 40, Voltage Regulators and Power Supplies; that chapter also describes the use of the required external ballast transistor to generate the 1.2 V. For specifications of the voltage ranges and limits and decoupling of the power supplies, see the *MPC5606S Microcontroller Data Sheet*.

Certain pins have dedicated functions that affect the behavior of the MCU after reset. These include pins to force test or alternate boot conditions and debug features. These are described in Chapter 3, Signal Description, and a hardware designer should take care that these pins are connected to ensure correct operation.

Beyond power supply and pins that have special functions, there are also pins that have special system purposes such as oscillator and reset pins. These are also described in Chapter 3, Signal Description. The reset pin is bidirectional and its function is closely tied to the reset generation module [Chapter 31, Reset Generation Module (MC_RGM)]. The crystal oscillator pins are dedicated to this function but the oscillator is not started automatically after reset. The oscillator module is described in Chapter 8, Clock Description, along with the internal clock architecture and the other oscillator sources on-chip.

1.8.2 Input/output pins

The majority of the pins on the MCU are input/output pins that may either operate as general purpose pins or be connected to a particular on-chip module. The arrangement allows a function to be available on several pins. The system designer should allocate the function for the pin before connecting to external hardware. The software should then choose the correct function to match the hardware. The pad characteristics can vary depending on the functions on the pad. Chapter 3, Signal Description, describes each pad type (for example, SLOW, M1, or SMD). Two pads may be able to carry the same function but

have different pad types. The electrical specification of the pads is described in the *MPC5606S Microcontroller Data Sheet*, dependent on the function enabled and the pad type.

There are four modules that configure the various functions available:

- System Integration Unit Lite (SIUL)
- Wakeup Unit (WKPU)
- LCD
- 32 KHz oscillator (SXOSC)

The SIUL configures the digital pin functions. Each pin has a register (PCR) in the module that allows selection of the output functions connected to the pin. The available settings for the PCR are described in Section , . Inputs are selected using the PSMI registers; these are described in Chapter 37, System Integration Unit Lite (SIUL). (PSMI registers connect a module to one of several pins, whereas the PCR registers connect a pin to one of several modules).

The WKPU provides the ability to cause interrupts and wake the MCU from low-power modes. It operates independently from the SIUL.

In addition to digital I/O functions there are special functions that provide analog functionality. These are listed in Section , . The special functions are enabled independently from the digital I/O, which means that the digital function on the pin must be disabled when the special function is active. The LCD module and the SXOSC oscillator are enabled in the modules. The ADC functions are enabled using the PCRs.

1.8.3 Software design

Certain modules provide system integration functions, and other modules (such as timers) provide specific functions.

From reset, the modules involved in configuring the system for application software are:

- Boot Assist Module (BAM) — determines the selected boot source.
- Reset Generation Module (MC_RGM) — determines the behavior of the MCU when various reset sources are triggered and reports the source of the reset.
- Mode Entry Module (MC_ME) — controls the MCU's operating mode the MCU and configures the peripherals and clocks and power supplies for each of the modes.
- Power Control Unit (MC_PCU) — determines which power domains (see Section Chapter 40, Voltage Regulators and Power Supplies) are active.
- Clock Generation Module (MC_CGM) — chooses the clock source for the system and many peripherals.

After reset, the MCU will automatically select the appropriate reset source and begin to execute code. At this point the system clock is the 16 MHz FIRC oscillator, the CPU is in supervisor mode, and all the memory is available. Initialization is required before most peripherals may be used and before the SRAM can be read (since the SRAM is protected by ECC, the syndrome will generally be uninitialized after reset and reads would fail the check). Accessing disabled features causes error conditions or interrupts.

A typical startup routine would involve initializing the software environment including stacks, heaps, variable initialization, and so on, and configuring the MCU for the application.

The MC_ME module enables the modules and other features like clocks. It is therefore an essential part of the initialization and operation software. In general, the software will configure an MC_ME mode to make certain peripherals, clocks, and memory active, and then switch to that mode.

Chapter 8, Clock Description, includes a graphic of the clock architecture of the MCU. This can be used to determine how to configure the MC_CGM module. In general, software will configure the module to enable the required clocks and PLLs and route these to the active modules.

After these steps are complete it is possible to configure the input/output pins and the modules for the application.

1.8.4 Other features

The MC_ME module manages low-power modes, and so it is likely that it will be used to switch into different configurations (module sets, clocks) depending on the application requirements.

The MCU includes two other features (both described in Chapter 4, Safety) to improve the integrity of the application:

- It is possible to enable a software watchdog (SWT) immediately at reset or afterwards to help detect code runaway.
- Individual register settings can be protected from unintended writes using the features of the Register Protection module. The protected registers are shown in Appendix A, Registers Under Protection.

Other integration functionality is provided by the System Status and Configuration Module (SSCM), described in Chapter 38, System Status and Configuration Module (SSCM).

Chapter 2
Memory Map

Table 2-1 shows the system memory map for the MPC5606S. All addresses on the MPC5606S, including those that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each IP block.

Table 2-1. MPC5606S system memory map

AP		Size [KB]	5602S	5604S	5606S	Region[1]
Start Address	End Address					
On-chip Flash Memories (Code Flash)						
0x00000000	0x00007FFF	32	Yes	Yes	Yes[2]	Code Flash Array 0
0x00008000	0x0000BFFF	16	Yes	Yes	Yes[2]	Code Flash Array 0
0x0000C000	0x0000FFFF	16	Yes	Yes	Yes[2]	Code Flash Array 0
0x00010000	0x00017FFF	32	Yes	Yes	Yes[2]	Code Flash Array 0
0x00018000	0x0001FFFF	32	Yes	Yes	Yes[2]	Code Flash Array 0
0x00020000	0x0003FFFF	128	Yes	Yes	Yes[2]	Code Flash Array 0
0x00040000	0x0005FFFF	128	No	Yes	Yes[2]	Code Flash Array 0
0x00060000	0x0007FFFF	128	No	Yes	Yes[2]	Code Flash Array 0
0x00080000	0x0009FFFF	128	No	No	Yes[2]	Code Flash Array 1
0x000A0000	0x000BFFFF	128	No	No	Yes[2]	Code Flash Array 1
0x000C0000	0x000DFFFF	128	No	No	Yes[2]	Code Flash Array 1
0x000E0000	0x000FFFFF	128	No	No	Yes[2]	Code Flash Array 1
0x00100000	0x001FFFFF	1024	—	—	—	Reserved
On-chip Flash Memories (Shadow for Code Flash)						
0x00200000	0x00203FFF	16	Yes	Yes	Yes	Code Flash Array 0 Shadow Sector
0x00204000	0x003FFFFF	2032	—	—	—	Reserved
On-chip Flash Memories (Test for Code Flash)						
0x00400000	0x00403FFF	16	Yes	Yes	Yes	Code Flash Array 0 Test Sector
0x00404000	0x0047FFFF	496	—	—	—	Reserved
0x00480000	0x00483FFF	16	No	No	Yes	Code Flash Array 1 Test Sector
0x00484000	0x007FFFFF	3568	—	—	—	Reserved

Table 2-1. MPC5606S system memory map (continued)

AP		Size [KB]	5602S	5604S	5606S	Region[1]
Start Address	End Address					
On-chip Flash Memories (Data Flash)						
0x00800000	0x00803FFF	16	Yes	Yes	Yes	Data Flash Array 0
0x00804000	0x00807FFF	16	Yes	Yes	Yes	Data Flash Array 0
0x00808000	0x0080BFFF	16	Yes	Yes	Yes	Data Flash Array 0
0x0080C000	0x0080FFFF	16	Yes	Yes	Yes	Data Flash Array 0
0x00810000	0x00BFFFFF	4032	—	—	—	Reserved
On-chip Flash Memories (Test for Data Flash)						
0x00C00000	0x00C03FFF	16	Yes	yes	Yes	Data Flash Array 0 Test Sector
0x00C04000	0x00FFFFFF	2032	—	—	—	Reserved
Emulation Mapping						
0x01000000	0x1FFFFFFF	507904	Yes	Yes	Yes	Flash Emulation Mapping
0x20000000	0x3FFFFFFF	524288	—	—	—	Reserved
SRAM						
0x40000000	0x40001FFF	8	Yes	Yes	Yes	SRAM (ECC protection, standby support)
0x40002000	0x40005FFF	16	Yes	Yes	Yes	SRAM (ECC protection, in PD2)
0x40006000	0x4000BFFF	24	No	Yes	Yes	SRAM (ECC protection, in PD2)
0x4000C000	0x5FFFFFFF	524240	—	—	—	Reserved
0x60000000	0x60027FFF	160	No	No	Yes	Graphics SRAM (no ECC protection, no standby support)
0x60028000	0x7FFFFFFF	524128	—	—	—	Reserved
External, Memory-mapped Serial Flash (supports one [Quad]SPI Serial Flash)						
0x80000000	0x87FFFFFF	131072	No	No	Yes	External Serial Flash Memory 0
0x88000000	0xBFFFFFFF	917504	—	—	—	Reserved
PBRIDGE(1) — Off Platform Peripherals (mirrored to PBRIDGE(0) memory range 0xFFE80000–0xFFEFFFFF)						
0xC3F80000	0xC3F87FFF	32	—	—	—	Reserved
0xC3F88000	0xC3F8BFFF	16	Yes	Yes	Yes	Code Flash 0 Configuration (CFLASH0)

Table 2-1. MPC5606S system memory map (continued)

AP		Size [KB]	5602S	5604S	5606S	Region[1]
Start Address	End Address					
0xC3F8C000	0xC3F8FFFF	16	Yes	Yes	Yes	Data Flash 0 Configuration (DFLASH0)
0xC3F90000	0xC3F93FFF	16	Yes	Yes	Yes	System Integration Unit Lite (SIUL)
0xC3F94000	0xC3F97FFF	16	Yes	Yes	Yes	WakeUp Unit (WKPU)
0xC3F98000	0xC3F9FFFF	32	—	—	—	Reserved
0xC3FA0000	0xC3FA3FFF	16	Yes	Yes	Yes	Enhanced Modular I/O Subsystem 0 (eMIOS0)
0xC3FA4000	0xC3FA7FFF	16	Yes	Yes	Yes	Enhanced Modular I/O Subsystem 1 (eMIOS1)
0xC3FA8000	0xC3FAFFFF	32	—	—	—	Reserved
0xC3FB0000	0xC3FB3FFF	16	No	No	Yes	Code Flash 1 Configuration (CFLASH1)
0xC3FB4000	0xC3FD7FFF	144	—	—	—	Reserved
0xC3FD8000	0xC3FDBFFF	16	Yes	Yes	Yes	System Status and Configuration Module (SSCM)
0xC3FDC000	0xC3FDFFFF	16	Yes	Yes	Yes	Mode Entry Module (MC_ME)
0xC3FE0000	0xC3FE3FFF	16	Yes	Yes	Yes	Clock Generation Module (MC_CGM, XOSC, IRCOSC, FMPLL_0, FMPLL_1, CMU0, CMU1)
0xC3FE4000	0xC3FE7FFF	16	Yes	Yes	Yes	Reset Generation Module (MC_RGM)
0xC3FE8000	0xC3FEBFFF	16	Yes	Yes	Yes	Power Control Unit (MC_PCU)
0xC3FEC000	0xC3FEFFFF	16	Yes	Yes	Yes	Real Time Counter (RTC/API)
0xC3FF0000	0xC3FF3FFF	16	Yes	Yes	Yes	Periodic Interrupt Timer (PIT/RTI)
0xC3FF4000	0xC3FFFFFF	48	—	—	—	Reserved
PBRIDGE(0) — Off Platform Peripherals (new range)						
0xFFE00000	0xFFE03FFF	16	Yes	Yes	Yes	Analog to Digital Converter 0 (ADC0)
0xFFE04000	0xFFE2FFFF	176			O	Reserved
0xFFE30000	0xFFE33FFF	16	Yes	Yes	Yes	Inter-IC Bus Interface Controller 0 (I^2C0)

Table 2-1. MPC5606S system memory map (continued)

AP		Size [KB]	5602S	5604S	5606S	Region[1]
Start Address	End Address					
0xFFE34000	0xFFE37FFF	16	Yes	Yes	Yes	Inter-IC Bus Interface Controller 1 (I^2C1)
0xFFE38000	0xFFE3BFFF	16	No	No	Yes	Inter-IC Bus Interface Controller 2 (I^2C2)
0xFFE3C000	0xFFE3FFFF	16	No	No	Yes	Inter-IC Bus Interface Controller 3 (I^2C3)
0xFFE40000	0xFFE43FFF	16	Yes	Yes	Yes	LINFlex 0
0xFFE44000	0xFFE47FFF	16	Yes	Yes	Yes	LINFlex 1
0xFFE48000	0xFFE5FFFF	96	—	—	—	Reserved
0xFFE60000	0xFFE60FFF	4	Yes	Yes	Yes	Stepper Motor Control (SMC)
0xFFE61000	0xFFE617FF	2	Yes	Yes	Yes	Stepper Stall Detect (SSD0)
0xFFE61800	0xFFE61FFF	2	Yes	Yes	Yes	Stepper Stall Detect (SSD1)
0xFFE62000	0xFFE627FF	2	Yes	Yes	Yes	Stepper Stall Detect (SSD2)
0xFFE62800	0xFFE62FFF	2	Yes	Yes	Yes	Stepper Stall Detect (SSD3)
0xFFE63000	0xFFE637FF	2	Yes	Yes	Yes	Stepper Stall Detect (SSD4)
0xFFE63800	0xFFE63FFF	2	Yes	Yes	Yes	Stepper Stall Detect (SSD5)
0xFFE64000	0xFFE6FFFF	48	—	—	—	Reserved
0xFFE70000	0xFFE73FFF	16	Yes	Yes	Yes	CAN Sampler (CANSP)
0xFFE74000	0xFFE77FFF	16	Yes	Yes	Yes	LCD Controller 0 (LCD0)
0xFFE78000	0xFFE7BFFF	16	Yes	Yes	Yes	Sound Generation Logic (SGL)
0xFFE7C000	0xFFE7FFFF	16	Yes	Yes	Yes	Display Control Unit 0 (DCU0)
PBRIDGE(0) — Off Platform Mirror From PBRIDGE(1) (Mirrored range from PBRIDGE(1) range 0xC3F80000–0xC3FFFFFF) (new range)						
0xFFE80000	0xFFEFFFFF	512	Yes	Yes	Yes	Reserved
PBRIDGE(0) — On Platform Peripherals (existing eSys range)						
0xFFF00000	0xFFF03FFF	16	Yes	Yes	Yes	PBRIDGE0
0xFFF04000	0xFFF07FFF	16	Yes	Yes	Yes	XBAR
0xFFF08000	0xFFF0FFFF	16	—	—	—	Reserved

Table 2-1. MPC5606S system memory map (continued)

AP		Size [KB]	5602S	5604S	5606S	Region[1]
Start Address	End Address					
0xFFF10000	0xFFF13FFF	16	Yes	Yes	Yes	Memory Protection Unit (MPU)
0xFFF14000	0xFFF37FFF	144	—	—	—	Reserved
0xFFF38000	0xFFF3BFFF	16	Yes	Yes	Yes	Software Watchdog (SWT0)
0xFFF3C000	0xFFF3FFFF	16	Yes	Yes	Yes	System Timer Module (STM0)
0xFFF40000	0xFFF43FFF	16	Yes	Yes	Yes	Error Correction Status Module
0xFFF44000	0xFFF47FFF	16	Yes	Yes	Yes	Direct Memory Access Controller 2 (DMA2x)
0xFFF48000	0xFFF4BFFF	16	Yes	Yes	Yes	Interrupt Controller (INTC)
0xFFF4C000	0xFFF8FFFF	272	—	—	—	Reserved
0xFFF90000	0xFFF93FFF	16	Yes	Yes	Yes	DSPI 0
0xFFF94000	0xFFF97FFF	16	Yes	Yes	Yes	DSPI 1
0xFFF98000	0xFFFA7FFF	64	—	—	—	Reserved
0xFFFA8000	0xFFFABFFF	16	No	No	Yes	QuadSPI 0
0xFFF9C000	0xFFFBFFFF	144	—	—	—	Reserved
0xFFFC0000	0xFFFC3FFF	16	Yes	Yes	Yes	FlexCan 0 (CAN0)
0xFFFC4000	0xFFFC7FFF	16	No	Yes	Yes	FlexCan 1 (CAN1)
0xFFFC8000	0xFFFDBFFF	80	—	—	—	Reserved
0xFFFDC000	0xFFFDFFFF	16	Yes	Yes	Yes	DMA Channel Multiplexer (DMA_MUX)
0xFFFE0000	0xFFFFBFFF	112	—	—	—	Reserved
0xFFFFC000	0xFFFFFFFF	16	Yes	Yes	Yes	Boot Assist Module (BAM)

1 The contents of memory addresses marked as reserved, and individual bits within a memory address that are marked as reserved may return any value when read unless otherwise indicated.

2 Flash sector can be accessed via both slave ports. Arbitration logic required in case two different masters attempt to access the same address at the same time.

Chapter 3
Signal Description

3.1 Introduction

The following sections provide signal descriptions and related information about functionality and configuration.

3.2 Package pinouts

The 144- and 176-pin LQFP pinouts and the 208 MAPBGA ball map are provided in the following figures.

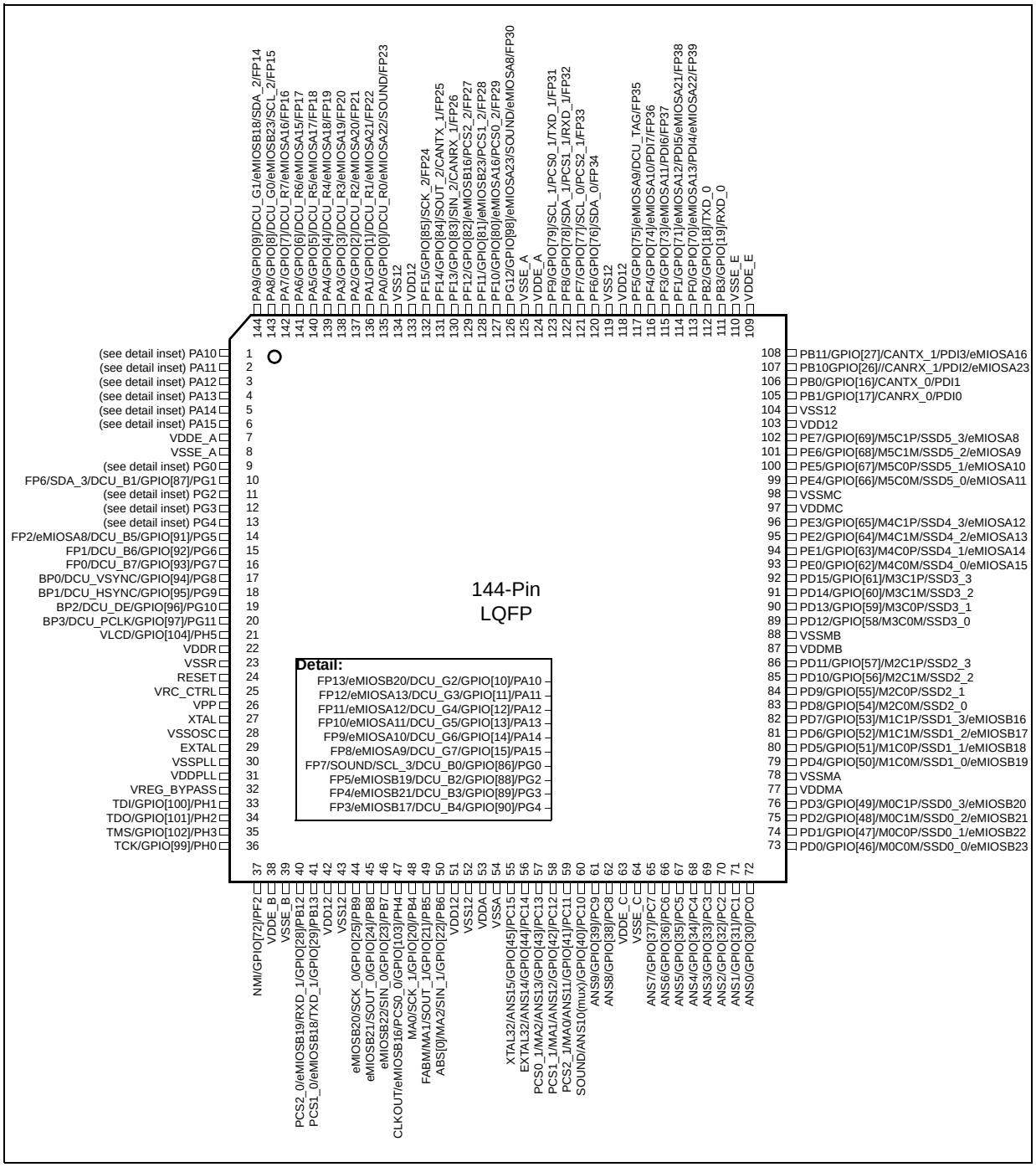


Figure 3-1. LQFP 144-pin configuration (top view)[1]

1. Availability of port pin alternate functions depends on product selection.

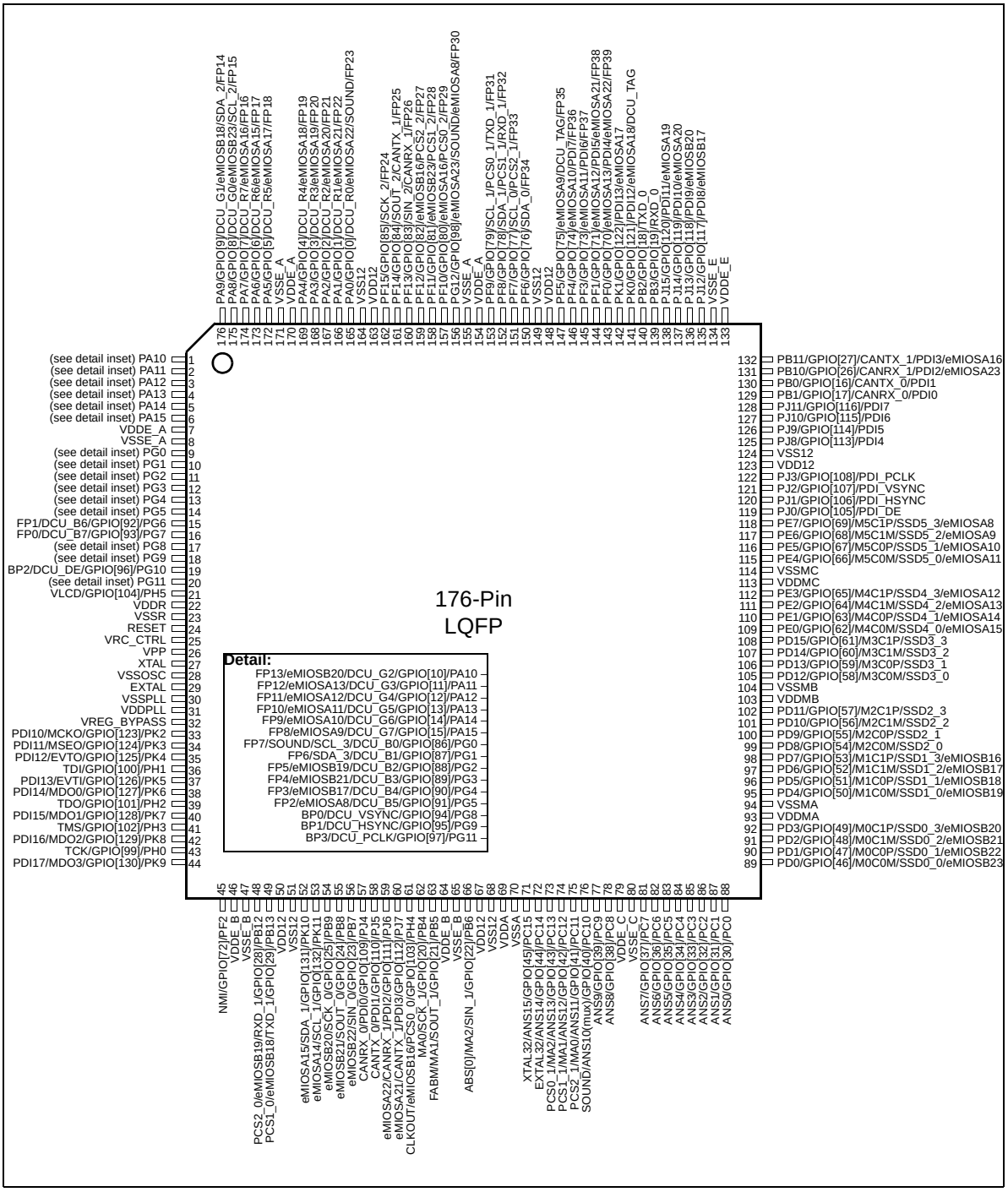


Figure 3-2. LQFP 176-pin configuration (top view)[1]

1. Availability of port pin alternate functions depends on product selection.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	PA0	PJ0	PJ1	PJ3	PJ5	PJ7	PJ14	PF0	PF5	PK9	PK5	NC	NC	PF10	PF11	PF12
B	PA1	VDDE_A	PJ2	PJ4	PJ6	PJ8	PJ15	PF1	PF6	NC	PK6	PK2	NC	NC	VDDE_E	PF13
C	PA2	PA3	VDDE_A	PJ9	PJ10	PJ12	PK0	PF3	PF7	NC	PK7	PK3	NC	VDDE_E	NC	PF14
D	PA4	PA5	PG0	VDD12	PJ11	PJ13	PK1	PF4	VDD12	PG12	PK8	PK4	VDD12	NC	NC	PF15
E	PA6	PA7	PG1	PG2									NC	NC	NC	NC
F	PA8	PA9	PG3	PG4									NC	NC	NC	NC
G	PA10	PA11	PG5	PG6			VSS	VSS	VSS	VSS			NC	PE7	PE1	NC
H	PA12	PA13	PA15	PG7			VSS	VSS	VSS	VSS			PE5	PE6	VDDMC	VSSMC
J	RESET	PA14	PG8	PG10			VSS	VSS	VSS	VSS			PE4	PE2	PE0	PD8
K	XTAL	VDDE_A	PG9	PG11			VSS	VSS	VSS	VSS			PE3	PD13	PD9	PD7
L	VSSPLL	VDDPLL	NMI/PF2	MDO3									PD15	PD12	VDDMB	VSSMB
M	EXTAL	VPP	PH3	VREG BYPASS									PD14	PD11	PD5	PD6
N	VDDR	VLCD	PH2	VDD12	PK11	PK10	PB8	PB5	PC13	PC9	PC6	PB11	VDDMA	PD10	PD4	PD3
P	VRC_ CTRL	PH1	VDDE_B	MDO2	MDO1	PB13	PB7	PB4	PC12	PC8	PC5	PC3	PB10	NC	PD2	PD1
R	PH0	VDDE_B	EVTO	PF9	PH4	PB12	PB6	PC15	PC11	PC7	PC4	PC2	PB3	PB2	VDDE_B	PD0
T	MCKO	MSEO	EVTI	PF8	MDO0	PB9	VDDE_C	PC14	PC10	VSSA	VDDA	PC1	PC0	PB1	PB0	VSSMA

Figure 3-3. 208 MAPBGA configuration[1] [2]

1. NC = Not connected.
2. 208 MAPBGA available only as development package for Nexus2+.

3.3 Pad configuration during reset phases

All pads have a fixed configuration under reset.

During the startup phase, all pads are forced to tristate.

After startup phase, all pads are floating with the following exceptions:

- PB[5] (FAB) is pulldown. Without external strong pullup the device starts fetching from flash.
- RESET pad is driven low. This is released only after PHASE2 reset completion.
- Main oscillator pads (EXTAL, XTAL) are tristate.
- Nexus output pads (MDO[*n*], MCKO, EVTO, MSEO) are forced to output.
- The following pads are pullup:
  - PB[6]
  - PH[0]
  - PH[1]
  - PH[3]
  - EVTI

3.4 Voltage supply pins

Voltage supply pins are used to provide power to the device. Two dedicated pins are used for 1.2 V regulator stabilization.

There is a preferred startup sequence for devices in the MPC5606S family. That sequence is described in the next paragraphs.

Broadly, the supply voltages can be grouped as follows:

- VREG HV supply (V_{DDR})
- Generic I/O supply
  - V_{DDA}
  - V_{DDE_A}
  - V_{DDE_B}
  - V_{DDE_C}
  - V_{DDE_E}
  - V_{DDMA}
  - V_{DDMB}
  - V_{DDMC}
  - V_{DDPLL}
- LV supply (V_{DD12})

The preferred order of ramp up is as follows:

1. Generic I/O supply

2. VREG HV supply (V_{DDR} should be the last HV supply to ramp up; it is also OK if all HV and generic I/O supplies including V_{DDR} ramp up together)
3. LV supply

The reason for following this sequence is to ensure that when VREG releases its LVDs, the I/O and other HV segments are powered properly. This is important because the MPC5606S does not monitor LVDs on I/O HV supplies.

Table 3-1. Voltage supply pin descriptions

Supply Pin	Function	Pin number	
		144 LQFP	176 LQFP
VDD12[1]	1.2 V core supply	42, 51, 103, 118, 133	50, 67, 123, 148, 163
VDDA	3.3 V/5 V ADC supply source	53	69
VDDE_A	3.3 V/5 V I/O supply	7, 124	7, 154, 170
VDDE_B	3.3 V/5 V I/O supply	38	46, 64
VDDE_C	3.3 V/5 V I/O supply	63	79
VDDE_E	3.3 V/5 V I/O supply	109	133
VDDMA[2]	Motor pads 5 V supply	77	93
VDDMB[2]	Motor pads 5 V supply	87	103
VDDMC[2]	Motor pads 5 V supply	97	113
VDDPLL	1.2 V PLL supply	31	31
VDDR	VREG reg supply	22	22
VPP[3]	9 V–12 V flash test analog write signal	26	26
VSS	Digital ground	8, 23, 39, 43, 52, 64, 104, 110, 119, 125, 134	8, 23, 47, 51, 68, 80, 124, 134, 149, 155, 164, 65, 171
VSSA	ADC ground	54	70
VSSMA	Stepper motor ground	78	94
VSSMB	Stepper motor ground	88	104
VSSMC	Stepper motor ground	98	114
VSSOSC	MHz oscillator ground	28	28
VSSPLL	PLL ground	30	30

1 Decoupling capacitors must be connected between these pins and the nearest V_{SS12} pin.
2 All stepper motor supplies need to be at same level (3.3 V or 5 V).
3 This signal needs to be connected to ground during normal operation.

3.5 Pad types

The pads available for system pins and functional port pins are described in:

- The port pin summary table

- The pad type descriptions
- The description of the pad configuration registers in Chapter 37, System Integration Unit Lite (SIUL)
- The device data sheet

3.6 System pins

The system pins are listed in Table 3-2.

Table 3-2. System pin descriptions

System pin	Function	I/O direction	Pad type	RESET config	Pin No.		
					144 LQFP	176 LQFP	208 MAPBGA
$\overline{\text{RESET}}$	Bidirectional reset with Schmitt-Trigger characteristics and noise filter.	I/O	M	Input, weak pullup	24	24	J1
XTAL	Analog input of the oscillator amplifier circuit. Needs to be grounded if oscillator bypass mode is used.	I	X	—	27	27	K1
EXTAL	Analog output of the oscillator amplifier circuit. Input for the clock generator in bypass mode.		X	—	29	29	M1
VRC_CTRL	VREG ballast control gain.	—	—	—	25	25	P1
VREG_BYPASS[1]	Pin used for factory testing.	I	X	—	32	32	M4

1 VREG_BYPASS should be pulled down externally.

3.7 Debug pins

The debug pins are listed in Table 3-3.

Table 3-3. Debug pin descriptions

Debug pin	Function	Pad type	I/O direction for debug	RESET config[1]	Pin number			
					144 LQFP	176 LQFP[2]	208 MAPBGA	
							Muxed	Dedi-cated[3]
TCK	JTAG test clock	S	I	Input, Pullup	36	43	R1	—
TDI	JTAG test data in	S	I	Input, Pullup	33	36	P2	—

Table 3-3. Debug pin descriptions (continued)

Debug pin	Function	Pad type	I/O direction for debug	RESET config[1]	Pin number			
					144 LQFP	176 LQFP[2]	208 MAPBGA	
							Muxed	Dedi-cated[3]
TDO	JTAG test data out	M1	O	Output, None	34	39	N3	—
TMS	JTAG test mode select	S	I	Input, Pullup	35	41	M3	—
EVTI	Nexus event input	M1	I	None, None	—	37	A11	—
EVTI	Nexus event input	M1	I	Input, Pullup	—	—	—	T3
EVTO	Nexus event output	M1	O	None, None	—	35	D12	—
EVTO	Nexus event output	M1	O	Input, Pullup	—	—	—	R3
MCKO	Nexus message clock output	F	O	None, None	—	33	B12	—
MCKO	Nexus message clock output	F	O	Input, Pullup	—	—	—	T1
MDO0	Nexus message data output 0	M1	O	None, None	—	38	B11	—
MDO0	Nexus message data output 0	M1	O	Input, Pullup	—	—	—	T5
MDO1	Nexus message data output 1	M1	O	None, None	—	40	C11	—
MDO1	Nexus message data output 1	M1	O	Input, Pullup	—	—	—	P5
MDO2	Nexus message data output 2	M1	O	None, None	—	42	D11	—
MDO2	Nexus message data output 2	M1	O	Input, Pullup	—	—	—	P4
MDO3	Nexus message data output 3	M1	O	None, None	—	44	A10	—
MDO3	Nexus message data output 3	M1	O	Input, Pullup	—	—	—	L4
MSEO	Nexus message start/end output	M1	O	None, None	—	34	C12	—
MSEO	Nexus message start/end output	M1	O	Input, Pullup	—	—	—	T2

1 See note for dedicated pins for 208 MAPBGA package.

2 On the 176 LQFP package, the Nexus debug pins are multiplexed with other GPIO. The 208 MAPBGA package provides dedicated Nexus debug pins as well as multiplexed Nexus debug pins. The multiplexing is described in the port pin summary table.

3 On the 208 MAPBGA package, the dedicated Nexus debug output pins (MDO[0:3] and MSEO) may drive an unknown value (high or low) immediately after startup but before the first clock edge propagates through the device, instead of being weakly pulled low. This may cause high currents if the pins are tied to a supply/ground in the application. If not used, these pins may be left unconnected.

3.8 Functional ports

The functional port pins are listed in Table 3-4.

Table 3-4. Port pin summary

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PA[0]	PCR[0]	Option 0 Option 1 Option 2 Option 3	GPIO[0] DCU_R0 eMIOSA[22] SOUND	FP23	SIUL DCU PWM/Timer Sound	I/O	M1	None, None	135	165	A1
PA[1]	PCR[1]	Option 0 Option 1 Option 2 Option 3	GPIO[1] DCU_R1 eMIOSA[21] —	FP22	SIUL DCU PWM/Timer —	I/O	M1	None, None	136	166	B1
PA[2]	PCR[2]	Option 0 Option 1 Option 2 Option 3	GPIO[2] DCU_R2 eMIOSA[20] —	FP21	SIUL DCU PWM/Timer —	I/O	M1	None, None	137	167	C1
PA[3]	PCR[3]	Option 0 Option 1 Option 2 Option 3	GPIO[3] DCU_R3 eMIOSA[19] —	FP20	SIUL DCU PWM/Timer —	I/O	M1	None, None	138	168	C2
PA[4]	PCR[4]	Option 0 Option 1 Option 2 Option 3	GPIO[4] DCU_R4 eMIOSA[18] —	FP19	SIUL DCU PWM/Timer —	I/O	M1	None, None	139	169	D1
PA[5]	PCR[5]	Option 0 Option 1 Option 2 Option 3	GPIO[5] DCU_R5 eMIOSA[17] —	FP18	SIUL DCU PWM/Timer —	I/O	M1	None, None	140	172	D2
PA[6]	PCR[6]	Option 0 Option 1 Option 2 Option 3	GPIO[6] DCU_R6 eMIOSA[15] —	FP17	SIUL DCU PWM/Timer —	I/O	M1	None, None	141	173	E1
PA[7]	PCR[7]	Option 0 Option 1 Option 2 Option 3	GPIO[7] DCU_R7 eMIOSA[16] —	FP16	SIUL DCU PWM/Timer —	I/O	M1	None, None	142	174	E2

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PA[8]	PCR[8]	Option 0 Option 1 Option 2 Option 3	GPIO[8] DCU_G0 eMIOSB[23] SCL_2	FP15	SIUL DCU PWM/Timer I^2C_2	I/O	M1	None, None	143	175	F1
PA[9]	PCR[9]	Option 0 Option 1 Option 2 Option 3	GPIO[9] DCU_G1 eMIOSB[18] SDA_2	FP14	SIUL DCU PWM/Timer I^2C_2	I/O	M1	None, None	144	176	F2
PA[10]	PCR[10]	Option 0 Option 1 Option 2 Option 3	GPIO[10] DCU_G2 eMIOSB[20] —	FP13	SIUL DCU PWM/Timer —	I/O	M1	None, None	1	1	G1
PA[11]	PCR[11]	Option 0 Option 1 Option 2 Option 3	GPIO[11] DCU_G3 eMIOSA[13] —	FP12	SIUL DCU PWM/Timer —	I/O	M1	None, None	2	2	G2
PA[12]	PCR[12]	Option 0 Option 1 Option 2 Option 3	GPIO[12] DCU_G4 eMIOSA[12] —	FP11	SIUL DCU PWM/Timer —	I/O	M1	None, None	3	3	H1
PA[13]	PCR[13]	Option 0 Option 1 Option 2 Option 3	GPIO[13] DCU_G5 eMIOSA[11] —	FP10	SIUL DCU PWM/Timer —	I/O	M1	None, None	4	4	H2
PA[14]	PCR[14]	Option 0 Option 1 Option 2 Option 3	GPIO[14] DCU_G6 eMIOSA[10] —	FP9	SIUL DCU PWM/Timer —	I/O	M2	None, None	5	5	J2
PA[15]	PCR[15]	Option 0 Option 1 Option 2 Option 3	GPIO[15] DCU_G7 eMIOSA[9] —	FP8	SIUL DCU PWM/Timer —	I/O	M1	None, None	6	6	H3

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PB[0]	PCR[16]	Option 0 Option 1 Option 2 Option 3	GPIO[16] CANTX_0 PDI1 —	—	SIUL FlexCAN_0 PDI —	I/O	M1	None, None	106	130	T15
PB[1]	PCR[17]	Option 0 Option 1 Option 2 Option3	GPIO[17] CANRX_0 PDI0 —	—	SIUL FlexCAN_0 PDI —	I/O	S	None, None	105	129	T14
PB[2]	PCR[18]	Option 0 Option 1 Option 2 Option3	GPIO[18] TXD_0 — —	—	SIUL LINFlex_0 — —	I/O	S	None, None	112	140	R14
PB[3]	PCR[19]	Option 0 Option 1 Option 2 Option3	GPIO[19] RXD_0 — —	—	SIUL LINFlex_0 — —	I/O	S	None, None	111	139	R13
PB[4]	PCR[20]	Option 0 Option 1 Option 2 Option 3	GPIO[20] SCK_1 MA0 —	—	SIUL DSPI_1 ADC —	I/O	M1	None, None	48	62	P8
PB[5]	PCR[21]	Option 0 Option 1 Option 2 Option 3	GPIO[21] SOUT_1 MA1 FABM	—	SIUL DSPI_1 ADC Control	I/O	M1	Input, Pulldown	49	63	N8
PB[6]	PCR[22]	Option 0 Option 1 Option 2 Option 3	GPIO[22] SIN_1 MA2 ABS[0]	—	SIUL DSPI_1 ADC Control	I/O	S	Input, Pullup	50	66	R7
PB[7]	PCR[23]	Option 0 Option 1 Option 2 Option 3	GPIO[23] SIN_0 eMIOSB[22] —	—	SIUL DSPI_0 PWM/Timer —	I/O	S	None, None	46	56	P7

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PB[8]	PCR[24]	Option 0 Option 1 Option 2 Option 3	GPIO[24] SOUT_0 eMIOSB[21] —	—	SIUL DSPI_0 PWM/Timer —	I/O	M1	None, None	45	55	N7
PB[9]	PCR[25]	Option 0 Option 1 Option 2 Option 3	GPIO[25] SCK_0 eMIOSB[20] —	—	SIUL DSPI_0 PWM/Timer —	I/O	M1	None, None	44	54	T6
PB[10]	PCR[26]	Option 0 Option 1 Option 2 Option 3	GPIO[26] CANRX_1 PDI2 eMIOSA[23]	—	SIUL FlexCAN_1 PDI PWM/Timer	I/O	S	None, None	107	131	P13
PB[11]	PCR[27]	Option 0 Option 1 Option 2 Option 3	GPIO[27] CANTX_1 PDI3 eMIOSA[16]	—	SIUL FlexCAN_1 PDI PWM/Timer	I/O	M1	None, None	108	132	N12
PB[12]	PCR[28]	Option 0 Option 1 Option 2 Option 3	GPIO[28] RXD_1 eMIOSB[19] PCS2_0	—	SIUL LINFlex_1 PWM/Timer DSPI_0	I/O	S	None, None	40	48	R6
PB[13]	PCR[29]	Option 0 Option 1 Option 2 Option 3	GPIO[29] TXD_1 eMIOSB[18] PCS1_0	—	SIUL LINFlex_1 PWM/Timer DSPI_0	I/O	S	None, None	41	49	P6
PB[14]	—	—	Reserved	—	—	—	—	—	—	—	—
PB[15]	—	—	Reserved	—	—	—	—	—	—	—	—
PC[0]	PCR[30]	Option 0 Option 1 Option 2 Option 3	GPIO[30] — — —	ANS[0]	SIUL — — —	I/O	J	None, None	72	88	T13

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PC[1]	PCR[31]	Option 0 Option 1 Option 2 Option 3	GPIO[31] — — —	ANS[1]	SIUL — — —	I/O	J	None, None	71	87	T12
PC[2]	PCR[32]	Option 0 Option 1 Option 2 Option 3	GPIO[32] — — —	ANS[2]	SIUL — — —	I/O	J	None, None	70	86	R12
PC[3]	PCR[33]	Option 0 Option 1 Option 2 Option 3	GPIO[33] — — —	ANS[3]	SIUL — — —	I/O	J	None, None	69	85	P12
PC[4]	PCR[34]	Option 0 Option 1 Option 2 Option 3	GPIO[34] — — —	ANS[4]	SIUL — — —	I/O	J	None, None	68	84	R11
PC[5]	PCR[35]	Option 0 Option 1 Option 2 Option 3	GPIO[35] — — —	ANS[5]	SIUL — — —	I/O	J	None, None	67	83	P11
PC[6]	PCR[36]	Option 0 Option 1 Option 2 Option 3	GPIO[36] — — —	ANS[6]	SIUL — — —	I/O	J	None, None	66	82	N11
PC[7]	PCR[37]	Option 0 Option 1 Option 2 Option 3	GPIO[37] — — —	ANS[7]	SIUL — — —	I/O	J	None, None	65	81	R10
PC[8]	PCR[38]	Option 0 Option 1 Option 2 Option 3	GPIO[38] — — —	ANS[8]	SIUL — — —	I/O	J	None, None	62	78	P10

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PC[9]	PCR[39]	Option 0 Option 1 Option 2 Option 3	GPIO[39] — — —	ANS[9]	SIUL — — —	I/O	J	None, None	61	77	N10
PC[10]	PCR[40]	Option 0 Option 1 Option 2 Option 3	GPIO[40] — SOUND —	ANS[10]	SIUL — SGL —	I/O	J	None, None	60	76	T9
PC[11]	PCR[41]	Option 0 Option 1 Option 2 Option 3	GPIO[41] — MA0 PCS2_1	ANS[11]	SIUL — ADC DSPI_1	I/O	J	None, None	59	75	R9
PC[12]	PCR[42]	Option 0 Option 1 Option 2 Option 3	GPIO[42] — MA1 PCS1_1	ANS[12]	SIUL — ADC DSPI_1	I/O	J	None, None	58	74	P9
PC[13]	PCR[43]	Option 0 Option 1 Option 2 Option 3	GPIO[43] — MA2 PCS0_1	ANS[13]	SIUL — ADC DSPI_1	I/O	J	None, None	57	73	N9
PC[14]	PCR[44]	Option 0 Option 1 Option 2 Option 3	GPIO[44] — — —	ANS[14] EXTAL32	SIUL — — —	I/O	J	None, None	56	72	T8
PC[15]	PCR[45]	Option 0 Option 1 Option 2 Option 3	GPIO[45] — — —	ANS[15] XTAL32	SIUL — — —	I/O	J	None, None	55	71	R8
PD[0]	PCR[46]	Option 0 Option 1 Option 2 Option 3	GPIO[46] M0C0M SSD0_0 eMIOSB[23]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	73	89	R16

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PD[1]	PCR[47]	Option 0 Option 1 Option 2 Option 3	GPIO[47] M0C0P SSD0_1 eMIOSB[22]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	74	90	P16
PD[2]	PCR[48]	Option 0 Option 1 Option 2 Option 3	GPIO[48] M0C1M SSD0_2 eMIOSB[21]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	75	91	P15
PD[3]	PCR[49]	Option 0 Option 1 Option 2 Option 3	GPIO[49] M0C1P SSD0_3 eMIOSB[20]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	76	92	N16
PD[4]	PCR[50]	Option 0 Option 1 Option 2 Option 3	GPIO[50] M1C0M SSD1_0 eMIOSB[19]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	79	95	N15
PD[5]	PCR[51]	Option 0 Option 1 Option 2 Option 3	GPIO[51] M1C0P SSD1_1 eMIOSB[18]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	80	96	M15
PD[6]	PCR[52]	Option 0 Option 1 Option 2 Option 3	GPIO[52] M1C1M SSD1_2 eMIOSB[17]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	81	97	M16
PD[7]	PCR[53]	Option 0 Option 1 Option 2 Option 3	GPIO[53] M1C1P SSD1_3 eMIOSB[16]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	82	98	K16
PD[8]	PCR[54]	Option 0 Option 1 Option 2 Option 3	GPIO[54] M2C0M SSD2_0 —	—	SIUL SMC SSD —	I/O	SMD	None, None	83	99	J16

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PD[9]	PCR[55]	Option 0 Option 1 Option 2 Option 3	GPIO[55] M2C0P SSD2_1 —	—	SIUL SMC SSD —	I/O	SMD	None, None	84	100	K15
PD[10]	PCR[56]	Option 0 Option 1 Option 2 Option 3	GPIO[56] M2C1M SSD2_2 —	—	SIUL SMC SSD —	I/O	SMD	None, None	85	101	N14
PD[11]	PCR[57]	Option 0 Option 1 Option 2 Option 3	GPIO[57] M2C1P SSD2_3 —	—	SIUL SMC SSD —	I/O	SMD	None, None	86	102	M14
PD[12]	PCR[58]	Option 0 Option 1 Option 2 Option 3	GPIO[58] M3C0M SSD3_0 —	—	SIUL SMC SSD —	I/O	SMD	None, None	89	105	L14
PD[13]	PCR[59]	Option 0 Option 1 Option 2 Option 3	GPIO[59] M3C0P SSD3_1 —	—	SIUL SMC SSD —	I/O	SMD	None, None	90	106	K14
PD[14]	PCR[60]	Option 0 Option 1 Option 2 Option 3	GPIO[60] M3C1M SSD3_2 —	—	SIUL SMC SSD —	I/O	SMD	None, None	91	107	M13
PD[15]	PCR[61]	Option 0 Option 1 Option 2 Option 3	GPIO[61] M3C1P SSD3_3 —	—	SIUL SMC SSD —	I/O	SMD	None, None	92	108	L13
PE[0]	PCR[62]	Option 0 Option 1 Option 2 Option 3	GPIO[62] M4C0M SSD4_0 eMIOSA[15]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	93	109	J15

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PE[1]	PCR[63]	Option 0 Option 1 Option 2 Option 3	GPIO[63] M4C0P SSD4_1 eMIOSA[14]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	94	110	G15
PE[2]	PCR[64]	Option 0 Option 1 Option 2 Option 3	GPIO[64] M4C1M SSD4_2 eMIOSA[13]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	95	111	J14
PE[3]	PCR[65]	Option 0 Option 1 Option 2 Option 3	GPIO[65] M4C1P SSD4_3 eMIOSA[12]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	96	112	K13
PE[4]	PCR[66]	Option 0 Option 1 Option 2 Option 3	GPIO[66] M5C0M SSD5_0 eMIOSA[11]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	99	115	J13
PE[5]	PCR[67]	Option 0 Option 1 Option 2 Option 3	GPIO[67] M5C0P SSD5_1 eMIOSA[10]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	100	116	H13
PE[6]	PCR[68]	Option 0 Option 1 Option 2 Option 3	GPIO[68] M5C1M SSD5_2 eMIOSA[9]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	101	117	H14
PE[7]	PCR[69]	Option 0 Option 1 Option 2 Option 3	GPIO[69] M5C1P SSD5_3 eMIOSA[8]	—	SIUL SMC SSD PWM/Timer	I/O	SMD	None, None	102	118	G14
PE[8]	—	—	Reserved	—	—	—	—	—	—	—	—
PE[9]	—	—	Reserved	—	—	—	—	—	—	—	—
PE[10]	—	—	Reserved	—	—	—	—	—	—	—	—
PE[11]	—	—	Reserved	—	—	—	—	—	—	—	—

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PE[12]	—	—	Reserved	—	—	—	—	—	—	—	—
PE[13]	—	—	Reserved	—	—	—	—	—	—	—	—
PE[14]	—	—	Reserved	—	—	—	—	—	—	—	—
PE[15]	—	—	Reserved	—	—	—	—	—	—	—	—
PF[0]	PCR[70]	Option 0 Option 1 Option 2 Option 3	GPIO[70] eMIOSA[13] PDI4 eMIOSA[22]	FP39	SIUL PWM/Timer PDI PWM/Timer	I/O	S	None, None	113	143	A8
PF[1]	PCR[71]	Option 0 Option 1 Option 2 Option 3	GPIO[71] eMIOSA[12] PDI5 eMIOSA[21]	FP38	SIUL PWM/Timer PDI PWM/Timer	I/O	S	None, None	114	144	B8
PF[2]	PCR[72]	Option 0 Option 1 Option 2 Option 3	GPIO[72] NMI — —	—	SIUL NMI — —	I/O	S	None, None	37	45	L3
PF[3]	PCR[73]	Option 0 Option 1 Option 2 Option 3	GPIO[73] eMIOSA[11] PDI6 —	FP37	SIUL PWM/Timer PDI —	I/O	M1	None, None	115	145	C8
PF[4]	PCR[74]	Option 0 Option 1 Option 2 Option 3	GPIO[74] eMIOSA[10] PDI7 —	FP36	SIUL PWM/Timer PDI —	I/O	M1	None, None	116	146	D8
PF[5]	PCR[75]	Option 0 Option 1 Option 2 Option 3	GPIO[75] eMIOSA[9] DCU_TAG —	FP35	SIUL PWM/Timer DCU —	I/O	M1	None, None	117	147	A9
PF[6]	PCR[76]	Option 0 Option 1 Option 2 Option 3	GPIO[76] SDA_0 — —	FP34	SIUL I^2C_0 — —	I/O	S	None, None	120	150	B9

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PF[7]	PCR[77]	Option 0 Option 1 Option 2 Option 3	GPIO[77] SCL_0 PCS2_1 —	FP33	SIUL I^{2}C_0 DSPI_1 —	I/O	S	None, None	121	151	C9
PF[8]	PCR[78]	Option 0 Option 1 Option 2 Option 3	GPIO[78] SDA_1 PCS1_1 RXD_1	FP32	SIUL I^{2}C_1 DSPI_1 LINFlex_1	I/O	S	None, None	122	152	T4
PF[9]	PCR[79]	Option 0 Option 1 Option 2 Option 3	GPIO[79] SCL_1 PCS0_1 TXD_1	FP31	SIUL I^{2}C_1 DSPI_1 LINFlex_1	I/O	S	None, None	123	153	R4
PF[10]	PCR[80]	Option 0 Option 1 Option 2 Option 3	GPIO[80] eMIOSA[16] PCS0_2 —	FP29	SIUL PWM/Timer QuadSPI —	I/O	M1	None, None	127	157	A14
PF[11]	PCR[81]	Option 0 Option 1 Option 2 Option 3	GPIO[81] eMIOSB[23] IO2/PCS1_2[6] —	FP28	SIUL PWM/Timer QuadSPI —	I/O	M1	None, None	128	158	A15
PF[12]	PCR[82]	Option 0 Option 1 Option 2 Option 3	GPIO[82] eMIOSB[16] IO3/PCS2_2[6] —	FP27	SIUL PWM/Timer QuadSPI —	I/O	M1	None, None	129	159	A16
PF[13]	PCR[83]	Option 0 Option 1 Option 2 Option 3	GPIO[83] IO0/SIN_2[6] CANRX_1 —	FP26	SIUL QuadSPI FlexCAN_1 —	I/O	M1	None, None	130	160	B16
PF[14]	PCR[84]	Option 0 Option 1 Option 2 Option 3	GPIO[84] IO1/SOUT_2[6] CANTX_1 —	FP25	SIUL QuadSPI FlexCAN_1 —	I/O	M1	None, None	131	161	C16

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PF[15]	PCR[85]	Option 0 Option 1 Option 2 Option 3	GPIO[85] SCK_2 — —	FP24	SIUL QuadSPI — —	I/O	F	None, None	132	162	D16
PG[0]	PCR[86]	Option 0 Option 1 Option 2 Option 3	GPIO[86] DCU_B0 SCL_3 SOUND	FP7	SIUL DCU I^2C_3 SGL	I/O	M2	None, None	9	9	D3
PG[1]	PCR[87]	Option 0 Option 1 Option 2 Option 3	GPIO[87] DCU_B1 SDA_3 —	FP6	SIUL DCU I^2C_3 —	I/O	M1	None, None	10	10	E3
PG[2]	PCR[88]	Option 0 Option 1 Option 2 Option 3	GPIO[88] DCU_B2 eMIOSB[19] —	FP5	SIUL DCU PWM/Timer —	I/O	M2	None, None	11	11	E4
PG[3]	PCR[89]	Option 0 Option 1 Option 2 Option 3	GPIO[89] DCU_B3 eMIOSB[21] —	FP4	SIUL DCU PWM/Timer —	I/O	M1	None, None	12	12	F3
PG[4]	PCR[90]	Option 0 Option 1 Option 2 Option 3	GPIO[90] DCU_B4 eMIOSB[17] —	FP3	SIUL DCU PWM/Timer —	I/O	M2	None, None	13	13	F4
PG[5]	PCR[91]	Option 0 Option 1 Option 2 Option 3	GPIO[91] DCU_B5 eMIOSA[8] —	FP2	SIUL DCU PWM/Timer —	I/O	M1	None, None	14	14	G3
PG[6]	PCR[92]	Option 0 Option 1 Option 2 Option 3	GPIO[92] DCU_B6 — —	FP1	SIUL DCU — —	I/O	M2	None, None	15	15	G4

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PG[7]	PCR[93]	Option 0 Option 1 Option 2 Option 3	GPIO[93] DCU_B7 — —	FP0	SIUL DCU — —	I/O	M1	None, None	16	16	H4
PG[8]	PCR[94]	Option 0 Option 1 Option 2 Option 3	GPIO[94] DCU_VSYNC — —	BP0	SIUL DCU — —	I/O	M2	Input, None	17	17	J3
PG[9]	PCR[95]	Option 0 Option 1 Option 2 Option 3	GPIO[95] DCU_HSYNC — —	BP1	SIUL DCU — —	I/O	M1	Input, None	18	18	K3
PG[10]	PCR[96]	Option 0 Option 1 Option 2 Option 3	GPIO[96] DCU_DE — —	BP2	SIUL DCU — —	I/O	M2	None, None	19	19	J4
PG[11]	PCR[97]	Option 0 Option 1 Option 2 Option 3	GPIO[97] DCU_PCLK — —	BP3	SIUL DCU — —	I/O	M1	None, None	20	20	K4
PG[12]	PCR[98]	Option 0 Option 1 Option 2 Option 3	GPIO[98] eMIOSA[23] SOUND eMIOSA[8]	FP30	SIUL PWM/Timer SGL PWM/Timer	I/O	S	None, None	126	156	D10
PG[13]	—	—	Reserved	—	—	—	—	—	—	—	—
PG[14]	—	—	Reserved	—	—	—	—	—	—	—	—
PG[15]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[0][7]	PCR[99]	Option 0 Option 1 Option 2 Option 3	GPIO[99] TCK — —	—	SIUL JTAG — —	I/O	S	Input, Pullup	36	43	R1

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PH[1][7]	PCR[100]	Option 0 Option 1 Option 2 Option 3	GPIO[100] TDI — —	—	SIUL JTAG — —	I/O	S	Input, Pullup	33	36	P2
PH[2][7]	PCR[101]	Option 0 Option 1 Option 2 Option 3	GPIO[101] TDO — —	—	SIUL JTAG — —	I/O	M1	Output, None	34	39	N3
PH[3][7]	PCR[102]	Option 0 Option 1 Option 2 Option 3	GPIO[102] TMS — —	—	SIUL JTAG — —	I/O	S	Input, Pullup	35	41	M3
PH[4]	PCR[103]	Option 0 Option 1 Option 2 Option 3	GPIO[103] PCS0_0 eMIOSB[16] CLKOUT	—	SIUL DSPI_0 PWM/Timer Control	I/O	F	None, None	47	61	R5
PH[5]	PCR[104]	Option 0 Option 1 Option 2 Option 3	GPIO[104] VLCD[8] — —	—	SIUL LCD — —	I/O	S	None, None	21	21	N2
PH[6]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[7]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[8]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[9]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[10]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[11]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[12]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[13]	—	—	Reserved	—	—	—	—	—	—	—	—
PH[14]	—	—	Reserved	—	—	—	—	—	—	—	—

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PH[15]	—	—	Reserved	—	—	—	—	—	—	—	—
PJ[0]	PCR[105]	Option 0 Option 1 Option 2 Option 3	GPIO[105] PDI_DE — —	—	SIUL PDI — —	I/O	S	None, None	—	119	A2
PJ[1]	PCR[106]	Option 0 Option 1 Option 2 Option 3	GPIO[106] PDI_HSYNC — —	—	SIUL PDI — —	I/O	S	None, None	—	120	A3
PJ[2]	PCR[107]	Option 0 Option 1 Option 2 Option 3	GPIO[107] PDI_VSYNC — —	—	SIUL PDI — —	I/O	S	None, None	—	121	B3
PJ[3]	PCR[108]	Option 0 Option 1 Option 2 Option 3	GPIO[108] PDI_PCLK — —	—	SIUL PDI — —	I/O	M1	None, None	—	122	A4
PJ[4]	PCR[109]	Option 0 Option 1 Option 2 Option 3	GPIO[109] PDI[0] CANRX_0 —	—	SIUL PDI FlexCAN_0 —	I/O	S	None, None	—	57	B4
PJ[5]	PCR[110]	Option 0 Option 1 Option 2 Option 3	GPIO[110] PDI[1] CANTX_0 —	—	SIUL PDI FlexCAN_0 —	I/O	M1	None, None	—	58	A5
PJ[6]	PCR[111]	Option 0 Option 1 Option 2 Option 3	GPIO[111] PDI[2] CANRX_1 eMIOSA[22]	—	SIUL PDI FlexCAN_1 PWM/Timer	I/O	S	None, None	—	59	B5
PJ[7]	PCR[112]	Option 0 Option 1 Option 2 Option 3	GPIO[112] PDI[3] CANTX_1 eMIOSA[21]	—	SIUL PDI FlexCAN_1 PWM/Timer	I/O	M1	None, None	—	60	A6

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PJ[8]	PCR[113]	Option 0 Option 1 Option 2 Option 3	GPIO[113] PDI[4] — —	—	SIUL PDI — —	I/O	S	None, None	—	125	B6
PJ[9]	PCR[114]	Option 0 Option 1 Option 2 Option 3	GPIO[114] PDI[5] — —	—	SIUL PDI — —	I/O	S	None, None	—	126	C4
PJ[10]	PCR[115]	Option 0 Option 1 Option 2 Option 3	GPIO[115] PDI[6] — —	—	SIUL PDI — —	I/O	S	None, None	—	127	C5
PJ[11]	PCR[116]	Option 0 Option 1 Option 2 Option 3	GPIO[116] PDI[7] — —	—	SIUL PDI — —	I/O	S	None, None	—	128	D5
PJ[12]	PCR[117]	Option 0 Option 1 Option 2 Option 3	GPIO[117] PDI[8] eMIOSB[17] —	—	SIUL PDI PWM/Timer —	I/O	M1	None, None	—	135	C6
PJ[13]	PCR[118]	Option 0 Option 1 Option 2 Option 3	GPIO[118] PDI[9] eMIOSB[20] —	—	SIUL PDI PWM/Timer —	I/O	M1	None, None	—	136	D6
PJ[14]	PCR[119]	Option 0 Option 1 Option 2 Option 3	GPIO[119] PDI[10] eMIOSA[20] —	—	SIUL PDI PWM/Timer —	I/O	M1	None, None	—	137	A7
PJ[15]	PCR[120]	Option 0 Option 1 Option 2 Option 3	GPIO[120] PDI[11] eMIOSA[19] —	—	SIUL PDI PWM/Timer —	I/O	M1	None, None	—	138	B7

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PK[0]	PCR[121]	Option 0 Option 1 Option 2 Option 3	GPIO[121] PDI[12] eMIOSA[18] DCU_TAG	—	SIUL PDI PWM/Timer DCU	I/O	M1	None, None	—	141	C7
PK[1]	PCR[122]	Option 0 Option 1 Option 2 Option 3	GPIO[122] PDI[13] eMIOSA[17] —	—	SIUL PDI PWM/Timer —	I/O	M1	None, None	—	142	D7
PK[2]	PCR[123]	Option 0 Option 1 Option 2 Option 3	GPIO[123] MCKO PDI[10] —	—	SIUL Nexus PDI —	I/O	F	None, None	—	33	B12
PK[3]	PCR[124]	Option 0 Option 1 Option 2 Option 3	GPIO[124] MSEO PDI[11] —	—	SIUL Nexus PDI —	I/O	M1	None, None	—	34	C12
PK[4]	PCR[125]	Option 0 Option 1 Option 2 Option 3	GPIO[125] EVTO PDI[12] —	—	SIUL Nexus PDI —	I/O	M1	None, None	—	35	D12
PK[5]	PCR[126]	Option 0 Option 1 Option 2 Option 3	GPIO[126] EVTI PDI[13] —	—	SIUL Nexus PDI —	I/O	M1	None, None	—	37	A11
PK[6]	PCR[127]	Option 0 Option 1 Option 2 Option 3	GPIO[127] MDO0 PDI[14] —	—	SIUL Nexus PDI —	I/O	M1	None, None	—	38	B11
PK[7]	PCR[128]	Option 0 Option 1 Option 2 Option 3	GPIO[128] MDO1 PDI[15] —	—	SIUL Nexus PDI —	I/O	M1	None, None	—	40	C11

Table 3-4. Port pin summary (continued)

Port pin	PCR register	Alternate function[1]	Function	Special function[2]	Peripheral[3]	I/O direction	Pad type[4]	RESET config.[5]	Pin number		
									144 LQFP	176 LQFP	208 MAPBGA
PK[8]	PCR[129]	Option 0 Option 1 Option 2 Option 3	GPIO[129] MDO2 PDI[16] —	—	SIUL Nexus PDI —	I/O	M1	None, None	—	42	D11
PK[9]	PCR[130]	Option 0 Option 1 Option 2 Option 3	GPIO[130] MDO3 PDI[17] —	—	SIUL Nexus PDI —	I/O	M1	None, None	—	44	A10
PK[10]	PCR[131]	Option 0 Option 1 Option 2 Option 3	GPIO[131] SDA_1 eMIOSA[15] —	—	SIUL I^2C_1 PWM/Timer —	I/O	S	None, None	—	52	N6
PK[11]	PCR[132]	Option 0 Option 1 Option 2 Option 3	GPIO[132] SCL_1 eMIOSA[14] —	—	SIUL I^2C_1 PWM/Timer —	I/O	S	None, None	—	53	N5
PK[12]	—	—	Reserved	—	—	—	—	—	—	—	—
PK[13]	—	—	Reserved	—	—	—	—	—	—	—	—
PK[14]	—	—	Reserved	—	—	—	—	—	—	—	—
PK[15]	—	—	Reserved	—	—	—	—	—	—	—	—

1 Alternate functions are chosen by setting the values of the PCR[n].PA bitfields inside the SIUL module. PCR[*n*].PA=00 → Option 0; PCR[*n*].PA=01 → Option 1; PCR[*n*].PA=10 → Option 2; PCR[*n*].PA=11 → Option 3. This is intended to select the output functions; to use one of the input functions, the PCR[*n*].IBE bit must be written to 1, regardless of the values selected in the PCR[*n*].PA bitfields. For this reason, the value corresponding to an input-only function is reported as —.

2 Special functions are enabled independently from the standard digital pin functions. Enabling standard I/O functions in the PCR registers may interfere with their functionality. ADC functions are enabled using the PCR[APC] bit; other functions are enabled by enabling the respective module.

3 Using the PSMI registers in the System Integration Unit Lite (SIUL), different pads can be multiplexed to the same peripheral input. Please see the SIUL chapter of the *MPC5606S Microcontroller Reference Manual* for details.

4 See Table 3-5.

5 Reset configuration is given as I/O direction and pull, for example, “Input, Pullup”.

6 This option on this pin has alternate functions that depend on whether the QuadSPI is in SPI mode or in serial flash mode (SFM).

7 Out of reset, pins PH[0:3] are available as JTAG pins (TCK, TDI, TDO, and TMS, respectively). It is up to the user to configure pins PH[0:3] when needed.

8 This pin can be used for LCD supply pin VLCD. Refer to the voltage supply pin descriptions in the MPC5606S data sheet for details.

Table 3-5. Pad type descriptions

Abbreviation[1]	Description
F	Fast (with GPIO and digital alternate function)
J	Slow pads with analog muxing (built for ADC channels)
M1	Medium (with GPIO and digital alternate function)
M2	Programmable medium/slow pad (programmed via the slew rate control in the PCR): Slew rate disabled: Slow driver configuration (AC/DC parameters same as for a slow pad) Slew rate enabled: Medium driver configuration (AC/DC parameters same as for a medium pad)
S	Slow (with GPIO and digital alternate function)
SMD	Stepper motor driver (with slew rate control)
X	Oscillator

1 The pad descriptions refer to the different Pad Configuration Register (PCR) types. Chapter 37, System Integration Unit Lite (SIUL), for the features available for each pad type.

3.9 Signal details

Table 3-6. Signal details

Signal	Peripheral	Description
ABS[0]	BAM	Alternate Boot Select. Gives an option to boot by downloading code via CAN or LIN.
ANS[0:15]	ADC	Inputs used to bring into the device sensor-based signals for A/D conversion. ANS[0:15] connect to ATD channels [32:47].
MA[0:2]	ADC	These three control bits are output to enable the selection for an external Analog Mux for expansion channels. The available 8 multiplexed channels connect to ATD channels [64:71].
FABM		Force Alternate Boot mode. Forces the device to boot from the external bus (Can or LIN). If not asserted, the device boots up from the lowest flash sector containing a valid boot signature.
DCU_DE	DCU	Indicates that valid pixels are present.
DCU_HSYNC	DCU	Horizontal sync pulse for TFT-LCD display.
DCU_PCLK	DCU	Output pixel clock for TFT-LCD display.
DCU_R[0:7], DCU_G[0:7], DCU_B[0:7]	DCU	Red, green and blue color 8-bit pixel values for TFT-LCD displays.
DCU_TAG	DCU	Indicates when a tagged pixel is present in safety mode.
DCU_VSYNC	DCU	Vertical sync pulse for TFT-LCD display.
PCS[0..2]_0, PCS[0..2]_1	DSPI	Peripheral chip selects when device is in Master mode; not used in slave modes.
SCK_0, SCK_1	DSPI	SPI clock signal—bidirectional.

Table 3-6. Signal details (continued)

Signal	Peripheral	Description
SIN_0, SIN_1	DSPI	SPI data input signal.
SOUT_0, SOUT_1	DSPI	SPI data output signal.
PCS0_2	QuadSPI	Peripheral chip select for serial flash mode or chip select 0 for SPI master mode.
IO2/PCS1_2	QuadSPI	Chip select 1 for SPI master mode and bidirectional IO2 for serial flash mode.
IO3/PCS2_2	QuadSPI	Chip select 2 for SPI master mode and bidirectional IO3 for serial flash mode.
IO0/SIN_2	QuadSPI	Data input signal for SPI master and slave modes and bidirectional IO0 for serial flash mode.
IO1/SOUT_2	QuadSPI	Data output signal for SPI master and slave modes and bidirectional IO1 for serial flash mode.
SCK_2	QuadSPI	Clock output signal for SPI master and serial flash modes and clock input signal for SPI slave mode.
eMIOSA[8:23], eMIOSB[16:23]	eMIOS	Enhanced Modular Input Output System. 16+8 channel eMIOS for timed input or output functions.
CANRX_0, CANRX_1	FlexCAN	Receive (RX) pins for the CAN bus transceiver.
CANTX_0, CANTX_1	FlexCAN	Transmit (TX) pins for the CAN bus transceiver.
SCL_0, SCL_1, SCL_2, SCL_3	I^2C	Bidirectional serial clock compatible with I^2C specifications.
SDA_0, SDA_1, SDA_2, SDA_3	I^2C	Bidirectional serial data compatible with I^2C specifications.
TCK	JTAG	Debug port serial clock as per JTAG specifications.
TDI	JTAG	Debug port serial data input port as per JTAG standards specifications.
TDO	JTAG	Debug port serial data output port as per JTAG standards specifications.
TMS	JTAG	Debug port Test Mode Select signal for the JTAG TAP controller state machine and indicates various state transitions for the TAP controller in the device.
BP[0:3]	LCD	Backplane signals from the LCD controlling the backplane reference voltage for the LCD display.
FP[0:39]	LCD	Frontplane signals for LCD segments.

Table 3-6. Signal details (continued)

Signal	Peripheral	Description
$\overline{\text{EVTI}}$	Nexus	Nexus2+ event input trigger.
$\overline{\text{EVTO}}$	Nexus	Nexus2+ event output trigger.
MCKO	Nexus	Output clock for the development tool.
MDO[0:3]	Nexus	Message output port pins that send information bits to the development tools for messages such as Branch Trace Message (BTM), Ownership Trace Message (OTM), Data Trace Message (DTM). Only available in reduced port mode.
$\overline{\text{MSEO}}$	Nexus	Output pin—Indicates the start or end of the variable length message on the MDO pins.
PDI[0:17]	DCU (PDI)	Video/graphic data in various RGB modes input to the DCU.
PDI_DE	DCU (PDI)	Input signal indicates the validity of pixel data on the Input PDI data bus.
PDI_HSYNC	DCU (PDI)	Input indicates the timing reference for the start of each frame line for the PDI Input data.
PDI_PCLK	DCU (PDI)	Input pixel clock from PDI.
PDI_VSYNC	DCU (PDI)	Input indicates the timing reference for the start of a frame for the PDI input data.
RXD_0	LINFlex	SCI/LIN Receive data signal—This port is used to download the code for the BAM boot sequence.
RXD_1	LINFlex	SCI/LIN Receive data signal. Input pad for the LIN SCI module. Connects to the internal LIN second port.
TXD_0	LINFlex	SCI/LIN Transmit data signal. This port is used to download the code for the BAM boot sequence.
TXD_1	LINFlex	SCI/LIN Transmit data signal—Transmit (output) port for the second LIN module in the chip.
SOUND	SGL	Sound signal to the speaker/buzzer.
SSD[0:5]_0 SSD[0:5]_1 SSD[0:5]_2 SSD[0:5]_3	SSD	Bidirectional control of stepper motors using stall detection module.
M[0:5]C0M M[0:5]C0P M[0:5]C1M M[0:5]C1P	SMC	Controls stepper motors in various configurations.
CLKOUT	MC_CGM	Output clock—It can be selected from several internal clocks of the device from the clock generation module.

Chapter 4
Safety

4.1 Register protection

4.1.1 Introduction

4.1.1.1 Overview

The Register Protection module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The registers that are protected on this device are listed in Appendix A, Registers Under Protection.

The protection module is located between the module under protection and the PBRIDGE. This is shown in Figure 4-1.

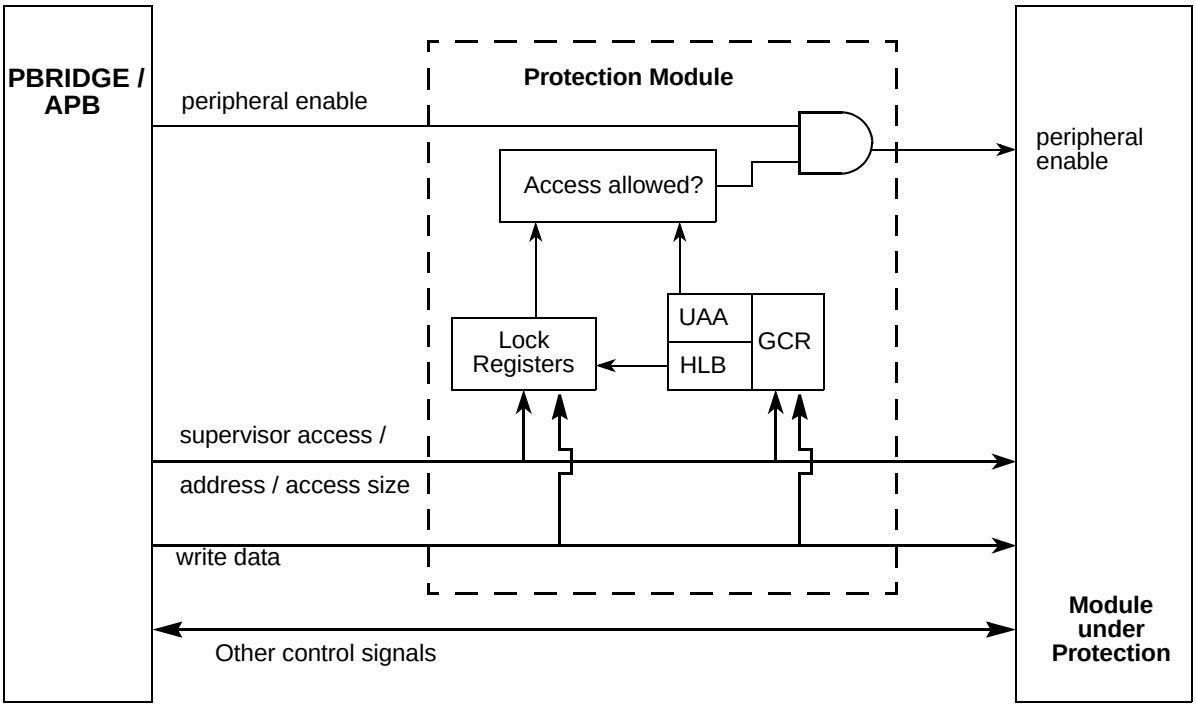


Figure 4-1. Register Protection block diagram

4.1.1.2 Features

The Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Address mirror automatically sets corresponding lock bit

- Once configured, lock bits can be protected from changes

4.1.1.3 Modes of operation

The Register Protection module is operable when the module under protection is operable. For further details about the availability please see Appendix A, Registers Under Protection.

4.1.2 External signal description

There are no external signals.

4.1.3 Memory map and register description

This section provides a detailed description of the memory map of a module using the Register Protection. The original 16 KB module memory space is divided into five areas as shown in Figure 4-2.

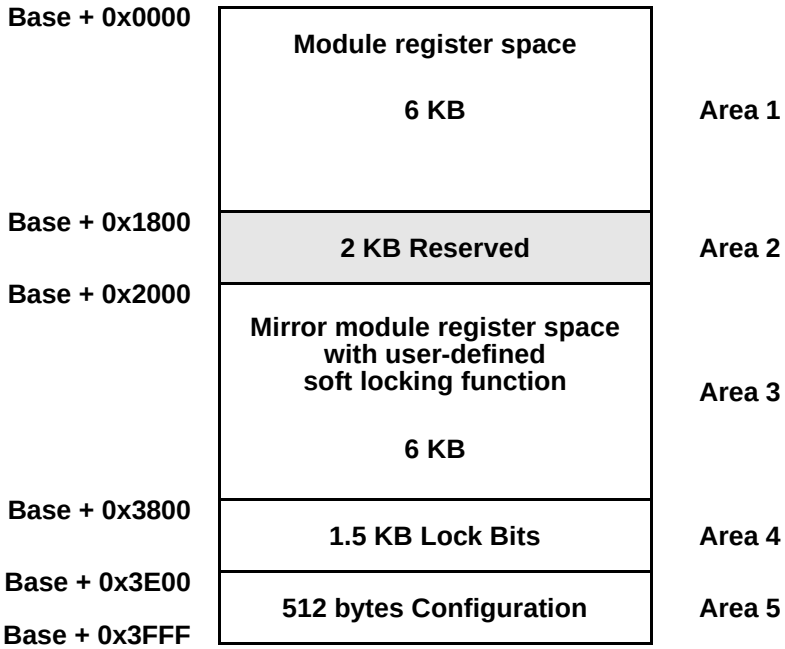


Figure 4-2. Register protection memory diagram

Area 1 is 6 KB starting at address 0x0000 and holds the normal functional module registers and is transparent for all read/write operations.

Area 2 is 2 KB starting at address 0x1800 and is a reserved area, that cannot be accessed.

Area 3 is 6 KB starting at address 0x2000 and is a mirror of area 1. A read/write access to these 0x2000 + X addresses will read/write the register at address X. As a side effect, a write access to address 0x2000 + X will set the optional Soft Lock Bits for this address X in the same cycle as the register at address X is written. Not all registers in area 1 need to have protection defined by associated Soft Lock

Bits. For unprotected registers at address Y, accesses to address 0x2000+Y will be identical to accesses at address Y. Soft Lock Bits are available in area 4 only for registers implemented in area 1 and defined as protectable.

Area 4 is 1.5 KB large and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.

Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

4.1.3.1 Memory map

Table 4-1 gives an overview of the Register Protection registers implemented.

Table 4-1. Register protection memory map

Address Offset	Use	Location
0x0000	Module Register 0 (MR0)	on page 114
0x0001	Module Register 1 (MR1)	on page 114
0x0002	Module Register 2 (MR2)	on page 114
0x0003–0x17FF	Module Register 3 (MR3)–Module Register 6143(MR6143)	on page 114
0x1800–0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	on page 114
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	on page 114
0x2002–0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2)– Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	on page 114
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	on page 114
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	on page 114
0x3802–0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11– Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143	on page 114
0x3E00–0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	on page 115

NOTE

Reserved registers in area #2 will be handled according to the protected IP (module under protection).

4.1.3.2 Register description

This section describes in address order all the Register Protection registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

4.1.3.2.1 Module Registers (MR0-6143)

This is the lower 6K module memory space, which holds all the functional registers of the module that is protected by the Register Protection module.

4.1.3.2.2 Module Register and Set Soft Lock Bit (LMR0–6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to an MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBR*n*.SLB*m*, according to the mapping described in Table 4-2.

4.1.3.2.3 Soft Lock Bit Register (SLBR0–1535)

These registers hold the Soft Lock Bits for the protected registers in memory area #1.

Address> Base + 0x3800–0x3DFF — Access: Supervisor read/write, User read -only

	0	1	2	3	4	5	6	7
R	0	0	0	0	SLB0	SLB1	SLB2	SLB3
W	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

Figure 4-3. Soft Lock Bit Register (SLBR*n*)

Table 4-2. SLBR*n* field descriptions

Field	Description
WE0 WE1 WE2 WE3	Write Enable Bits for Soft Lock Bits (SLB): WE0 enables writing to SLB0 WE1 enables writing to SLB1 WE2 enables writing to SLB2 WE3 enables writing to SLB3 0 SLB is not modified 1 Value is written to SLB
SLB0 SLB1 SLB2 SLB3	Soft Lock Bits for one MR*n* register: SLB0 can block accesses to MR[(*n* × 4) + 0] SLB1 can block accesses to MR[(*n* × 4) + 1] SLB2 can block accesses to MR[(*n* × 4) + 2] SLB3 can block accesses to MR[(*n* × 4) + 3] 0 Associated MR*n* byte is unprotected and writable 1 Associated MR*n* byte is locked against write accesses

Table 4-3 gives some examples of how SLBR*n*.SLB and MR*n* go together:

Table 4-3. Soft lock bits vs. protected address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

4.1.3.2.4 Global Configuration Register (GCR)

This register is used to make global configurations related with the Register Protection.

Address: 0x3FFC Access:Supervisor read/write, User read -only

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

Figure 4-4. Global Configuration Register (GCR)

Table 4-4. GCR field descriptions

Field	Description
HLB	Hard Lock Bit. This register cannot be cleared once it is set by software. It can only be cleared by a system reset. 0 All SLB bits are accessible and can be modified. 1 All SLB bits are write-protected and cannot be modified.
UAA	User Access Allowed. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module. 1 The registers in the module under protection can be accessed in the User mode without any additional restrictions.

NOTE

The GCR.UAA bit has no effect on the allowed access modes for the registers in the Register Protection module.

4.1.4 Functional description

4.1.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address = multiples of 4)
- 16-bit (address = multiples of 2)
- 8-bit (address = multiples of 1)
- Unprotected (address = multiples of 1)

This section gives examples for various protection configurations.

For all addresses that are protected there are SLBR*n*.SLB*m* bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected, the corresponding SLBR*n*.SLB*m* bit is always 0b0, no matter what software is writing to.

4.1.4.2 Change lock settings

To change the setting whether an address is locked or unlocked, the corresponding SLBR*n*.SLB*m* bit needs to be changed. This can be done using the following methods:

- Modify the SLBR*n*.SLB*m* directly by writing to area #4
- Set the SLBR*n*.SLB*m* bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

4.1.4.2.1 Change lock settings directly via area #4

In memory area #4, the lock bits are located. They can be modified by writing to them. Each SLBR*n*.SLB*m* bit has a mask bit SLBR*n*.WE*m*, which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 4-5 shows two modification examples. The left example shows a write access to the SLBR*n* register specifying a mask value that allows modification of all SLBR*n*.SLB*m* bits. The example on the right specifies a mask that only allows modification of the bits SLBR*n*.SLB[3:1].

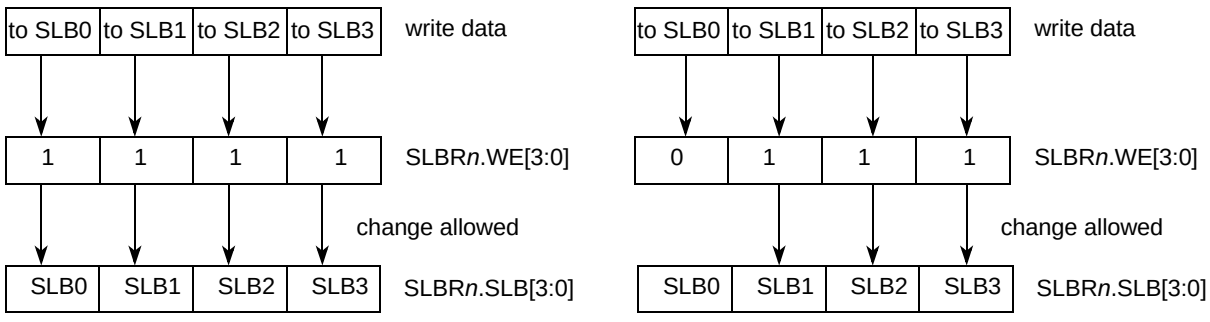


Figure 4-5. Change lock settings directly via area #4

Figure 4-5 showed four registers that can be protected 8-bit wise. In Figure 4-6 registers with 16-bit protection and in Figure 4-7 registers with 32-bit protection are shown:

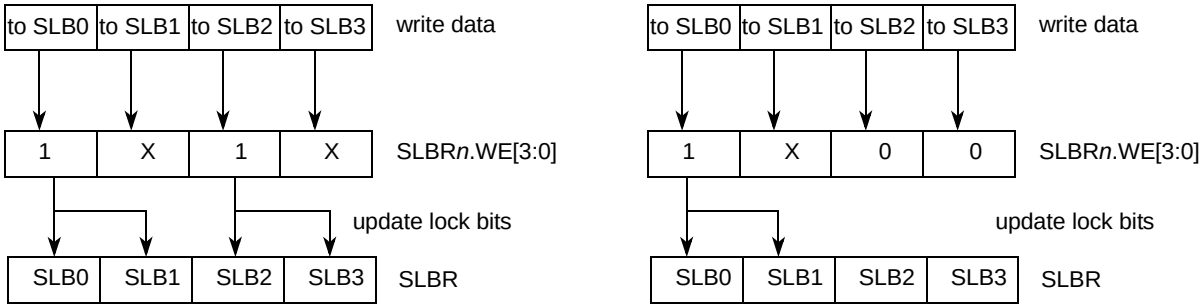


Figure 4-6. Change lock settings for 16-bit protected addresses

On the right side of Figure 4-6 it is shown that the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[1] also. This is done as the address reflected by SLBR*n*.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBR*n*.WE[0] must be set while SLBR*n*.WE[1] does not matter. As the enable bits SLBR*n*.WE[3:2] are cleared the lock bits SLBR*n*.SLB[3:2] remain unchanged.

In the example on the left side of Figure 4-6 the data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] and the data written to SLBR*n*.SLB[2] is mirrored to SLBR*n*.SLB[3], as for both registers the write enables are set.

In Figure 4-7 a 32-bit wise protected register is shown. When SLBR*n*.WE[0] is set the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[3:1] also. Otherwise SLBR*n*.SLB[3:0] remains unchanged.

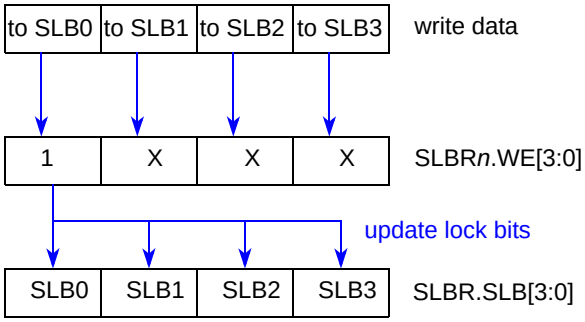


Figure 4-7. Change lock settings for 32-bit protected addresses

Figure 4-8 shows an example that has a mixed protection size configuration:

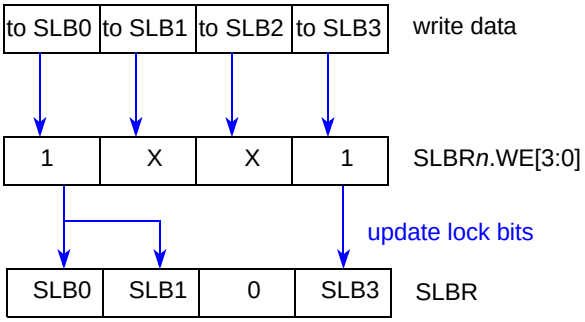


Figure 4-8. Change lock settings for mixed protection

The data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBR*n*.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBR*n*.SLB[3] is written to SLBR*n*.SLB[3].

4.1.4.2.2 Enable locking via mirror module space (area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. Figure 4-9 shows one example:

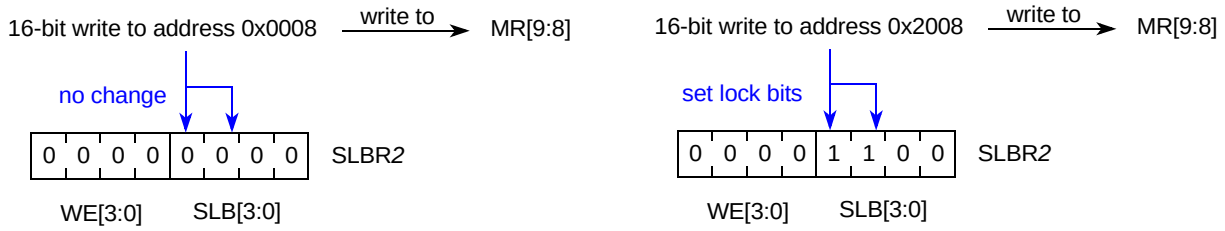


Figure 4-9. Enable locking via mirror module space (area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of Figure 4-6).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of Figure 4-6).

Figure 4-10 shows an example where some addresses are protected and some are not:

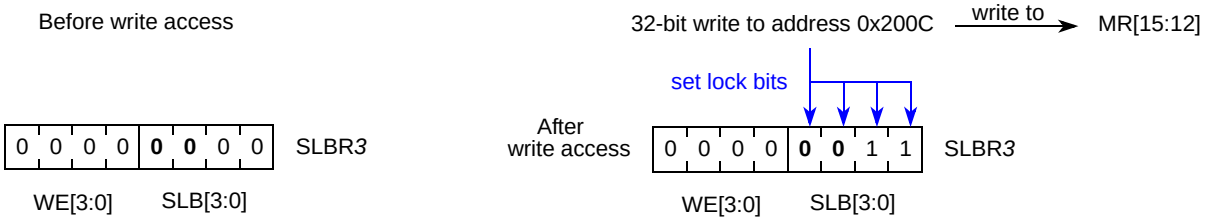


Figure 4-10. Enable locking for protected and unprotected addresses

In the example in Figure 4-10, addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C, only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

NOTE

Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

4.1.4.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in Section 4.1.4.2.1, Change lock settings directly via area #4, and Section 4.1.4.2.2, Enable locking via mirror module space (area #3), is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until after a system reset.

4.1.4.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to Figure 4-2.

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes, a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set, a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set.

4.1.5 Reset

The reset state of each individual bit is shown within the Register description section (See Section 4.1.3.2, Register description). In summary, after reset, locking for all MR*n* registers is disabled. The registers can be accessed in Supervisor mode only.

4.2 Software Watchdog Timer (SWT)

4.2.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT

requires periodic execution of a watchdog servicing sequence. Writing the sequence resets the timer to a specified timeout period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial timeout, a reset is always generated on a second consecutive timeout.

The SWT provides a window functionality. When this functionality is programmed, the servicing action should take place within the defined window. When occurring outside the defined period, the SWT will generate a reset.

4.2.2 Features

The SWT has the following features:

- 32-bit timeout register to set the timeout period
- The unique SWT counter clock is the undivided low-power internal oscillator (IRC 128 khz), no other clock source can be selected
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial timeout
- Master access protection
- Hard and soft configuration lock bits
- The SWT is started on Phase1 exit and counts unconditionally during Phase2 to monitor flash boot sequence. SWT is held in reset during Phase3 but can start again at Phase3 exit depending on the value of the shadow flash configuration bit NVUSR0[WATCHDOG_EN].

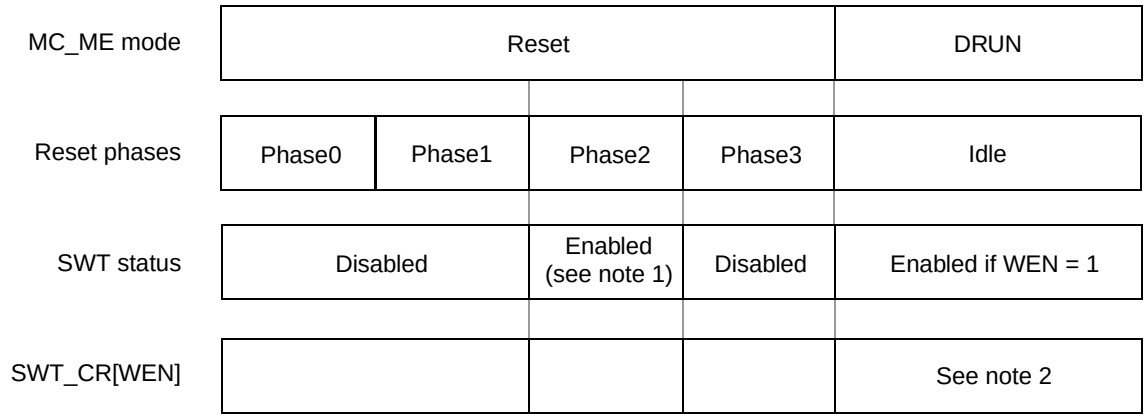
4.2.3 Modes of operation

When enabled, the SWT is always active in all modes except Standby when it is always disabled.

To simplify software development it is possible to temporarily suspend the SWT counter while the MCU is stopped by a debugger. While the MCU is running the SWT counter runs normally. This feature is enabled by setting the SWT_CR[FRZ] bit and is only available when the CPU has debug mode active (see the CPU reference manual for more information on debug mode and support).

Software watchdog is not available in Standby mode. As soon as out of Standby mode, the SWT behaves as in a usual “out of reset” situation.

Figure 4-11 shows the operation timing diagram of the SWT. Table 4-5 describes the SWT operation after reset.



Notes:
1) The SWT is started on Phase1 exit and counts unconditionally during Phase2 to monitor the flash memory boot sequence.
2) Value copied from configuration bit NVUSR0[WATCHDOG_EN] in the shadow flash memory (software can modify it later)

Figure 4-11. SWT operation timing diagram

Table 4-5. SWT operation after reset

SWT_CR [WEN]	MCU mode	CPU debug active	SWT_CR [FRZ]	SWT operation
0	—	No	0 or 1	Off
1	Normal (MC_ME modes DRUN, Run0:3, Halt, Stop, Safe)	No	0 or 1	Running
	Debug[1] (MC_ME modes DRUN, Run0:3, Halt, Stop, Safe)	Yes	0	Running
		Yes	1	Halted
0 or 1	Standby	No	No	Off

1 SWT Debug mode occurs when the processor is stopped due to user specified debug criteria such as breakpoint.

4.2.4 External signal description

The SWT module does not have any external interface signals.

4.2.5 Memory map and register description

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set, then the SWT generates a system reset on an invalid access. Otherwise, a bus error is generated. If either the HLK or SLK bits in the SWT_CR are set then the SWT_CR, SWT_TO, and SWT_WN registers are read-only.

4.2.5.1 Memory map

The SWT memory map is shown in Table 4-6.

Table 4-6. SWT memory map

Offset from SWT_BASE	Register	Access[1]	Reset Value	Location
0x0000	SWT Control Register (SWT_CR)	R/W	0xFF00_011B	on page 122
0x0004	SWT Interrupt Register (SWT_IR)	R/W	0x0000_0000	on page 123
0x0008	SWT Timeout Register (SWT_TO)	R/W	0x0003_FDE0	on page 124
0x000C	SWT Window Register (SWT_WN)	R/W	0x0000_0000	on page 125
0x0010	SWT Service Register (SWT_SR)	W	0x0000_0000	on page 125
0x0014	SWT Counter Output Register (SWT_CO)	R	0x0000_0000	on page 126
0x0018	SWT Service Key Register (SWT_SK)	W	0x0000_0000	on page 126
0x001C–0x3FFF	Reserved			

1 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

4.2.5.2 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT. This register is read only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	KEY	RIA	WND	ITR	HLK	SLK	CSL	0	FRZ	WEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	See note

Note: SWT_CR[WEN] value is copied from configuration bit NVUSR0[WATCHDOG_EN] during reset

Figure 4-12. SWT Control Register (SWT_CR)

The SWT_CR reset value is 0x8000_011A or 0x8000_011B, corresponding to MAP0 = 1 (only CPU access allowed), RIA = 1 (reset on invalid SWT access), SLK = 1 (soft lock), CSL = 1 (IRC clock source for counter), FRZ = 1 (freeze available while debugging), WEN = 0 or 1 (copied from configuration bit NVUSR0[WATCHDOG_EN]).

Table 4-7. SWT_CR field descriptions

Field	Description
MAPn	Master Access Protection for Master n. The platform bus master assignments are device specific. 0 Access for the master is not enabled 1 Access for the master is enabled
KEY	Keyed Service Mode. 0 Fixed Service Sequence. The fixed sequence 0xA602, 0xB480 is used to service the watchdog. 1 Keyed Service mode. Two pseudorandom key values are used to service the watchdog.
RIA	Reset on Invalid Access. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1
WND	Window mode. 0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
ITR	Interrupt Then Reset. 0 Generate a reset on a timeout 1 Generate an interrupt on an initial timeout, reset on a second consecutive timeout
HLK	Hard Lock. This bit is only cleared at reset. 0 SWT_CR, SWT_TO and SWT_WN are read/write registers if SLK=0 1 SWT_CR, SWT_TO and SWT_WN are read only registers
SLK	Soft Lock. This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO and SWT_WN are read/write registers if HLK=0 1 SWT_CR, SWT_TO and SWT_WN are read only registers
CSL	Clock Selection. Selects the LP IRC 128 khz oscillator clock that drives the internal timer. CSL bit can be written.The status of the bit has no effect on counter clock selection on this device. 0 System clock. (Not applicable on this device) 1 Oscillator clock.
FRZ	Freeze available during debug. This function is only operational when the CPU is in an active debug mode. 0 SWT counter continues to run independent of the CPU status 1 SWT counter is stopped when the CPU is stopped by a debugger
WEN	Watchdog Enabled. 0 SWT is disabled 1 SWT is enabled

4.2.5.3 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the timeout interrupt flag.

Address: Base + 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-13. SWT Interrupt Register (SWT_IR)

Table 4-8. SWT_IR field descriptions

Field	Description
TIF	Timeout Interrupt Flag. The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request. 1 Interrupt request due to an initial timeout.

4.2.5.4 SWT Timeout Register (SWT_TO)

The SWT Timeout (SWT_TO) register contains the 32-bit timeout period. This register is read only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: Base + 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WTO[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WTO[15:0]															
Reset	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0

Figure 4-14. SWT Timeout Register (SWT_TO)

The default counter value (SWT_TO_RST) is 1280 (0x500 hexadecimal), which corresponds to 10 ms with a 128 kHz clock.

Table 4-9. SWT_TO Register field descriptions

Field	Description
WTO	Watchdog timeout period in clock cycles. An internal 32-bit down counter is loaded with this value or 0x100, whichever is greater, when the service sequence is written or when the SWT is enabled.

4.2.5.5 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: Base + 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WST[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WST[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-15. SWT Window Register (SWT_WN)

Table 4-10. SWT_WN Register field descriptions

Field	Description
WST	Window start value. When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

4.2.5.6 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) service register is the target for service sequence writes used to reset the watchdog timer.

Address: Base + 0x0010 Access: User write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	WSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-16. SWT Service Register (SWT_SR)

Table 4-11. SWT_SR field descriptions

Field	Description
WSC	Watchdog Service Code.This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SL]). To service the watchdog, the value 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SL]), the value 0xC520 followed by 0xD928 is written to the WSC field.

4.2.5.7 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read only register that shows the value of the internal down counter when the SWT is disabled.

Address: Base + 0x0014 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNT[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-17. SWT Counter Output Register (SWT_CO)

Table 4-12. SWT_CO Register field descriptions

Field	Description
CNT	Watchdog Count. When the watchdog is disabled (SWT_CR[WEN]=0) this field shows the value of the internal down counter. When the watchdog is enabled the value of this field is 0x0000_0000. Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

4.2.5.8 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: Base + 0x0018 Access: User write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-18. SWT Service Key Register (SWT_SK)

Table 4-13. SWT_SK field descriptions

Field	Description
SK	Service Key.This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[KEY] is set, the next key value to be written to the SWT_SR is (17 × SK + 3) mod 2^{16}.

4.2.6 Functional description

The SWT is a 32-bit timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes a a control register (SWT_CR), an interrupt register (SWT_IR), timeout register (SWT_TO), a window register (SWT_WN), a service register (SWT_SR) and a counter output register (SWT_CO).

The SWT_CR includes bits to enable the timer, set configuration options and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is 0 when exiting Reset mode if the flash user option bit 31 (WATCHDOG_EN) is `0'. If the reset value of WATCHDOG_EN is 1, the SWT_CR[WEN] bit is set and the watchdog starts operation automatically after reset is released.

The SWT_TO register holds the watchdog timeout period in clock cycles unless the value is less than 0x100, in which case the timeout period is set to 0x100. This timeout period is loaded into an internal 32-bit down counter when the SWT is enabled and each time a valid service sequence is written. The SWT_CR[CSL] bit selects which clock (system or oscillator) is used to drive the down counter.

NOTE

The initial value of the SWT timer counter may not be correct when starting after reset. Therefore, it is important to clear the SWT as soon as possible after reset to minimize an unexpected SWT event. The SWT will behave correctly once it has been cleared at least once.

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked the SWT_CR, SWT_TO and SWT_WN registers are read only. The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset. The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register. The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

When enabled, the SWT requires periodic execution of the watchdog servicing sequence. The service sequence is a write of 0xA602 followed by a write of 0xB480 to the SWT_SR[WSC] field. Writing the service sequence loads the internal down counter with the timeout period. There is no timing requirement between the two writes. The service sequence logic ignores unlock sequence writes and recognizes the 0xA602, 0xB480 sequence regardless of previous writes. Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the timeout period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example,

if the SWT_TO register is set to 5000 and SWT_WN register is set to 1000 then the service sequence must be performed in the last 20% of the timeout period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

The interrupt then reset bit SWT_CR[ITR] controls the action taken when a timeout occurs. If the SWT_CR[ITR] bit is not set, a reset is generated immediately on a timeout. If the SWT_CR[ITR] bit is set, an initial timeout causes the SWT to generate an interrupt and load the down counter with the timeout period. If the service sequence is not written before the second consecutive timeout, the SWT generates a system reset. The interrupt is indicated by the timeout interrupt flag SWT_IR[TIF]. The interrupt request is cleared by writing a one to the SWT_IR[TIF] bit.

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO can be used during a software self-test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the timeout value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO read to determine if the internal down counter is working properly.

Chapter 5
Analog-to-Digital Converter (ADC)

5.1 Overview

5.1.1 Device-specific features

- 10-bit resolution
- As many as 16 channels, expandable to 23 channels via external multiplexing
- Minimum conversion time of 1 µs
- Conversion triggering sources:
  - Software
  - PIT channel 2 (for normal conversion trigger only)
- Two different sampling and conversion time registers CTR[1:2] (extended internal channels, external channels)
- As many as 23 data registers for storing converted data; conversion information, such as mode of operation (normal, injected), is associated to data value
- Conversions on external channels managed in the same way as internal channels, making it transparent to the application
- External decode signals (3 bits) to control the external analog multiplexers
- One Shot/Scan modes
- Chain Injection mode
- Hardware-triggered DMA transfer requests
- Power-down mode
- Two different Abort functions provide the ability to abort either single-channel conversion or chain conversion
- Four programmable analog watchdogs with interrupt capability
  - Allows continuous hardware monitoring of as many as four analog input channels
  - Alternate analog thresholds
- Auto-clock-off

5.1.2 Device-specific implementation

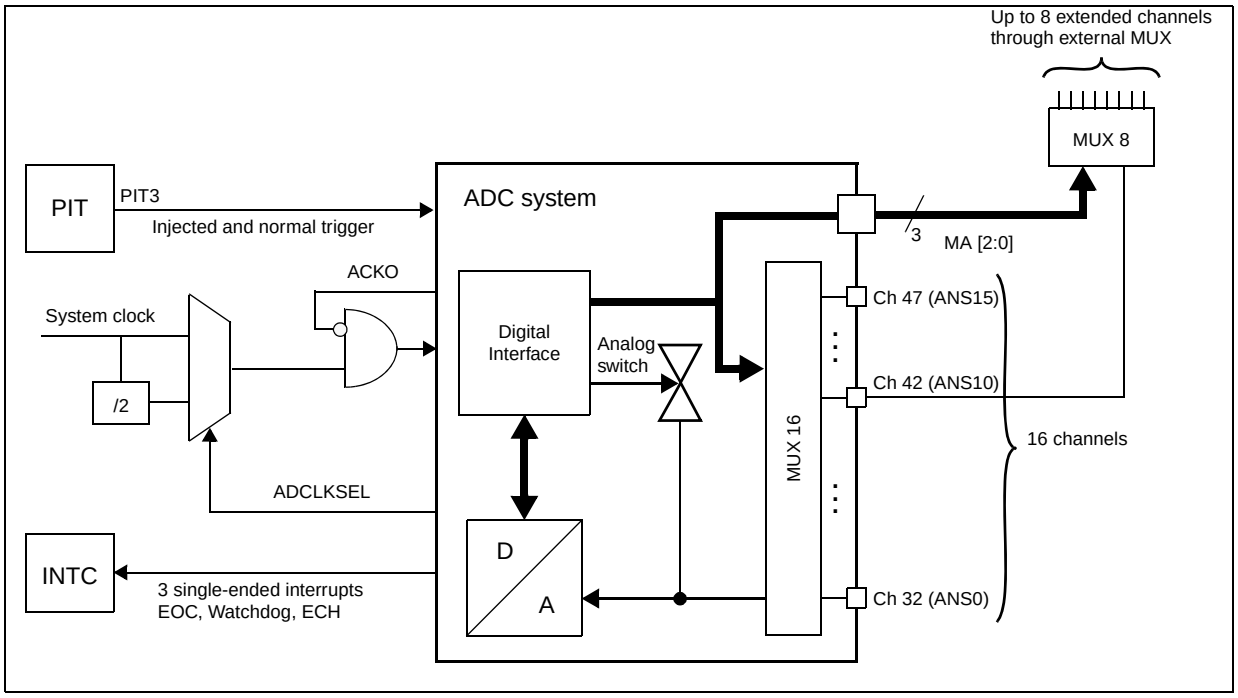


Figure 5-1. ADC implementation

5.2 Introduction

The analog-to-digital converter (ADC) block provides accurate and fast conversions for a wide range of applications. An ADC analog part has its corresponding digital interface (ADCDig).

The ADC digital interface contains advanced features for normal or injected conversion. A conversion can be triggered by software or hardware (PIT3).

There are several types of input channels:

- Internal extended, ANS (internally multiplexed standard accuracy channels)
- External, ANX (externally multiplexed standard accuracy channels)

The mask registers present within the ADCDig can be programmed to configure which channel is to be converted.

Several external decode signals MA[2:0] (multiplexer address) are provided for external channel selection and are available as alternate functions on GPIO.

A conversion timing register for configuring different sampling and conversion times is associated to each channel type.

Analog watchdogs allow continuous hardware monitoring.

5.3 Functional description

5.3.1 Analog channel conversion

Two conversion modes are available within the ADCDig:

- Normal conversion
- Injected conversion

5.3.1.1 Normal conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (NCMR). Each channel can be individually enabled by setting 1 in the corresponding field of NCMR registers. Mask registers must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (NSTART bit in the Main Status Register (MSR) is reset).

5.3.1.2 Start of normal conversion

By programming the configuration bits in the Main Configuration Register (MCR), the normal conversion can be started in two ways:

- By software (TRGEN reset)—If the external trigger enable bit is reset, the conversion chain starts when the NSTART bit in the MCR is set.
- By trigger (TRGEN set)—An on-chip internal signal triggers an ADC conversion. The settings in the MCR select how conversions are triggered based on these internal signals:
  - If the EDGLEV (edge/level selection) bit in the MCR is cleared, then when a rising/falling edge (depending on the EDGE bit in MCR) is detected in the signal, the NSTART bit is set in the MSR and the programmed conversion starts. EDGE = 0 selects a falling edge. EDGE = 1 selects a rising edge.
  - If the EDGLEV bit in the MCR is set, the conversion is started if and only if the NSTART bit in the MCR is set and the programmed level on the trigger signal is detected. The level is selected using the EDGE bit in the MCR. EDGE = 0 means that the start of conversion is enabled if the signal is low. If EDGE = 1, the start of conversion is enabled when the signal is high.

Table 5-1. Configurations for starting normal conversion

Type of conversion start	MCR				MSR	Result
	TRGEN	NSTART	EDGLEV	EDGE	NSTART	
Software	0	1	—	—	1	Conversion chain starts
Trigger	1	—	0	0	1	A falling edge detected in a trigger signal sets the NSTART bit in the MSR and starts the programmed conversion.
				1		A rising edge detected in a trigger signal sets the NSTART bit in the MSR and starts the programmed conversion.
Trigger	1	1	1	0	1	The conversion is started if the programmed level on the trigger signal is detected: the start of conversion is enabled if the external pin is low.
				1	1	The conversion is started if the programmed level on the trigger signal is detected: the start of conversion is enabled if the external pin is high.

The NSTART status bit in the MSR is automatically set when the normal conversion starts. At the same time the NSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running conversion is completed.

If the content of all the normal conversion mask registers is zero (that is, no channel is selected) the conversion operation is considered completed and the interrupt ECH (see further) is immediately issued after the start of conversion.

5.3.1.3 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot
- Scan

To enter one of these modes, it is necessary to program the MODE bit in the MCR. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital, as shown in Figure 5-2.

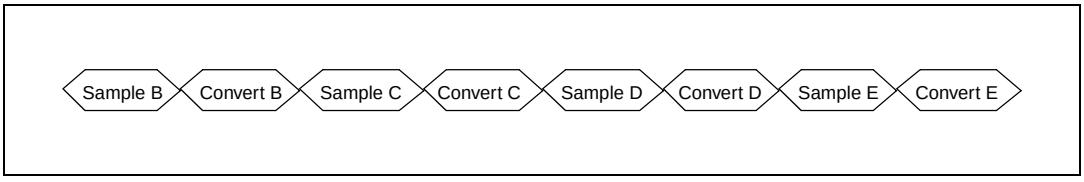


Figure 5-2. Normal conversion flow

In One Shot mode (MODE = 0) a sequential conversion specified in the NCMR registers is performed only once. At the end of each conversion, the digital result of the conversion is stored in the corresponding data register.

Example 5-1. One Shot mode (MODE = 0)

Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the One Shot mode. MODE = 0 is set for One Shot mode. Conversion starts from channel B followed by conversion of channels D and E. At the end of conversion of channel E the scanning of channels stops.

The NSTART status bit in the MSR is automatically set when the normal conversion starts. At the same time the NSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running conversion is completed.

In Scan mode (MODE = 1), a sequential conversion of N channels specified in the NCMR registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

The NSTART status bit in the MSR is automatically set when the normal conversion starts. Unlike One Shot mode, the NSTART bit in the MCR is not reset. It can be reset by software when the user needs to stop scan mode. In that case, the ADC completes the current scan conversion and, after the last conversion, also resets the NSTART bit in the MSR.

Example 5-2. Scan mode (MODE = 1)

Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in Scan mode. MODE = 1 is set for Scan mode. Conversion starts from channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of channels D and E. This sequence repeats itself till the NSTART bit in the MCR is reset by software.

If the conversion is started by an external trigger and EDGLEV is 0, the NSTART bit in the MCR is not set. As a consequence, the only way to stop scan mode conversion once it has started is to set the MODE bit to 0.

At the end of each conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit) and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit).

5.3.1.4 Injected channel conversion

A conversion chain can be injected into the ongoing normal conversion by configuring the Injected Conversion Mask Registers (JCMR). As normal conversion, each channel can be individually selected. This injected conversion can only occur in One Shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped, as shown in Figure 5-3.

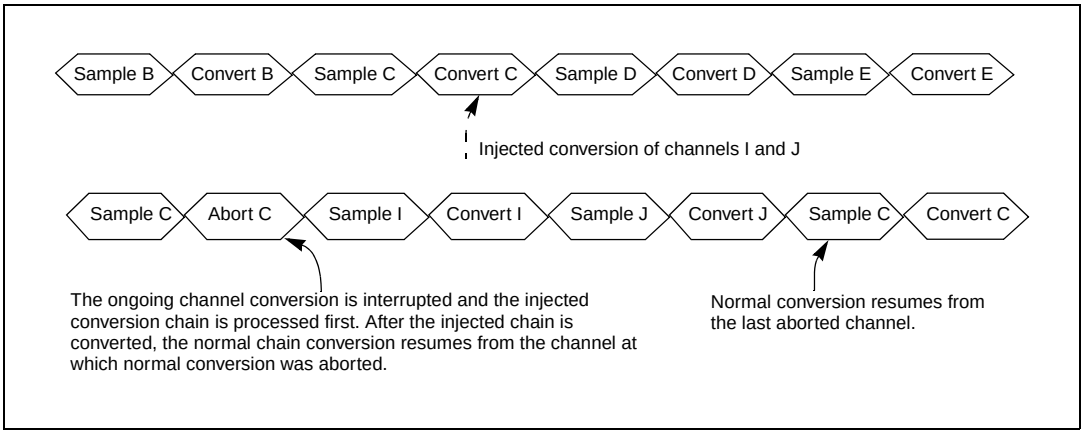


Figure 5-3. Injected sample/conversion sequence

The injected conversion can be started by software setting the JSTART bit in the MCR; the current conversion is suspended and the injected chain is converted. At the end of the chain, the JSTART bit in the MSR is reset and the normal chain conversion is resumed.

The JSTART status bit in the MSR is automatically set when the injected conversion starts. At the same time the JSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running injected conversion is completed.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

If the content of all the injected conversion mask registers is zero (that is, no channel is selected) the interrupt JECH is immediately issued after the start of conversion.

Once started, injected chain conversion cannot be interrupted by any other conversion type (it can, however, be aborted; see Section 5.3.1.5, Abort conversion).

5.3.1.5 Abort conversion

Two different abort functions are provided.

- The user can abort the ongoing conversion by setting the ABORT bit in the MCR. The current conversion is aborted and the conversion of the next channel of the chain is immediately started (generating a new start pulse to the Analog ADC). In the case of an abort operation, the NSTART/JSTART bit remains set and the ABORT bit is reset after the conversion of the next channel starts. The EOC corresponding to the aborted channel is not generated. This behavior is true for normal or triggered/injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported, generating an ECH interrupt.
- It is also possible to abort the current chain conversion by setting the ABORTCHAIN bit in the MCR. In that case the behavior of the ADC depends on the MODE bit. In fact, if scan mode is disabled, the NSTART bit is automatically reset together with the ABORTCHAIN bit. Otherwise, if the MODE bit is set to 1, a new chain conversion is started. The EOC of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

When a chain conversion abort is requested (ABORTCHAIN bit is set) while an injected conversion is running over a suspended normal conversion, both injected chain and normal conversion chain are aborted (both the NSTART and JSTART bits are also reset).

NOTE

Setting either the ABORT or ABORTCHAIN bit when no conversion is taking place can cause undetermined operation of the next programmed conversion chain.

5.3.2 Analog clock generator and conversion timings

The clock frequency can be selected by programming the ADCLKSEL bit in the MCR. When this bit is set to 1 the ADC clock has the same frequency as the system clock. Otherwise, the ADC clock is half of the system clock frequency. The ADCLKSEL bit can be written only in power-down mode.

When the internal divider is not enabled (ADCCLKSEL = 1), it is important that the associated clock divider in the clock generation module be 1. This is needed to ensure 50% clock duty cycle.

The direct clock should basically be used only in low-power mode when the device is using only the 16 MHz fast internal RC oscillator, but the conversion still requires a 16 MHz clock (an 8 MHz clock is not fast enough).

In all other cases, the ADC should use the clock divided by two internally.

5.3.3 ADC sampling and conversion timing

In order to support different loading and switching times, several different Conversion Timing registers (CTR) are present. There is one register per channel type. INPLATCH and INPCMP configurations are limited when the system clock frequency is greater than 20 MHz.

When a conversion is started, the ADC connects the internal sampling capacitor to the respective analog input pin, allowing the capacitance to charge up to the input voltage value. The time to load the capacitor is referred to as sampling time. After completion of the sampling phase, the evaluation phase starts and all the bits corresponding to the resolution of the ADC are estimated to provide the conversion result.

The conversion times are programmed via the bit fields of the CTR. Bit fields INPLATCH, INPCMP and INPSAMP are used to define the total conversion duration (T_{conv}) and in particular the partition between sampling phase duration (T_{sample}) and total evaluation phase duration (T_{eval}).

The sampling phase duration is:

$$T_{sample} = (INPSAMP - ndelay) \bullet T_{ck}$$

$$INPSAMP \geq 3$$

where *ndelay* is equal to 0.5 if INPSAMP is less than or equal to 0x06, otherwise it is 1. INPSAMP must be greater than or equal to 3 (hardware requirement).

The total evaluation phase duration is:

$$T_{eval} = 10 \bullet T_{biteval} = 10 \bullet (INPCMP \bullet T_{ck})$$

$$(INPCMP \geq 1) \quad \text{and} \quad (INPLATCH < INPCMP)$$

INPCMP must be greater than or equal to 1 and INPLATCH must be less than INPCMP (hardware requirements).

The total conversion duration is (not including external multiplexing):

$$T_{conv} = T_{sample} + T_{eval} + (ndelay \bullet T_{ck})$$

The timings refer to the unit T_{ck}, where f_{ck} = (1/2 x ADC peripheral set clock). The maximum clock frequency is specified in Table 5-2.

Table 5-2. Max AD_clk frequency and related configuration settings

INPLATCH	INPCMP	INPSAMP	AD_clk f_{max} (MHz)	T_{sample} min. (ns)
0	0x1	0x3	20	125
0	0x1	0x4	20 + 4%	168
1	0x2	0x4	20 + 4%	168
1	0x2	0x5	20 + 4%	135
1	0x3	0x7	32 + 4%	132
1	0x3	0x7	40 + 4%	128
1	0x3	0x8	50 + 4%	134
1	0x3	0x9	60 + 4%	128

Table 5-3 lists the possible combinations by configuring the AD_clk at 60 MHz.

Table 5-3. ADC sampling and conversion timing

INPLATCH	INPCMP	INPSAMP	T_{sample}[1]	T_{eval}	ndelay	T_{conv}
1	11	0000 1001	8 × Tck	30 × Tck	1 × Tck	39 × Tck[2]
1	11	0000 1010	9 × Tck	30 × Tck	1 × Tck	40 × Tck
1	11	0000 1011	10 × Tck	30 × Tck	1 × Tck	41 × Tck
1	11	0000 1100	11 × Tck	30 × Tck	1 × Tck	42 × Tck
1	11	0000 1101	12 × Tck	30 × Tck	1 × Tck	43 × Tck
1	11	0000 1110	13 × Tck	30 × Tck	1 × Tck	44 × Tck
1	11	0000 1111	14 × Tck	30 × Tck	1 × Tck	45 × Tck
...	...	...	...	...	...	...
1	11	1111 1100	251 × Tck	30 × Tck	1 × Tck	282 × Tck

Table 5-3. ADC sampling and conversion timing (continued)

INPLATCH	INPCMP	INPSAMP	T_{sample}[1]	T_{eval}	ndelay	T_{conv}
1	11	1111 1101	252 × Tck	30 × Tck	1 × Tck	283 × Tck
1	11	1111 1110	253 × Tck	30 × Tck	1 × Tck	284 × Tck
1	11	1111 1111	254 × Tck	30 × Tck	1 × Tck	285 × Tck

1 Represents the number of clock cycles that this operation will last

2 The ADC minimum conversion time at 60 MHz frequency is 39 × Tck; that corresponds to 650 ns.

5.3.4 Programmable analog watchdog

5.3.4.1 Introduction

The analog watchdogs are used for determining whether the result of a channel conversion lies within a given guarded area (as shown in Figure 5-4) specified by an upper and a lower threshold value named THRH and THRL respectively.

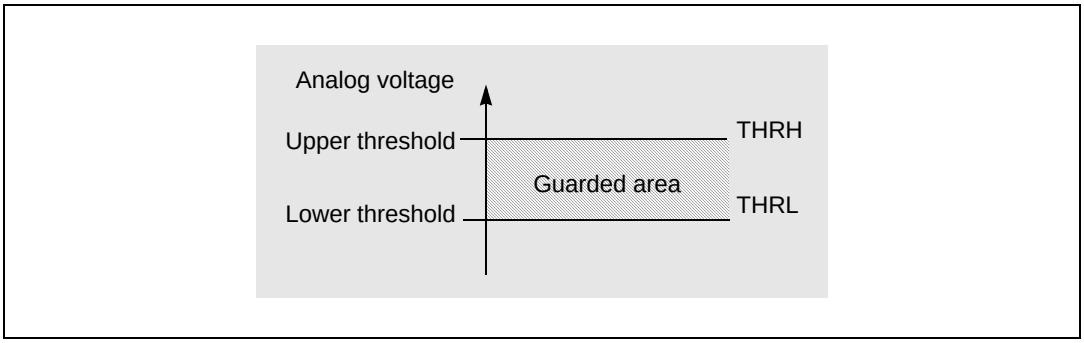


Figure 5-4. Guarded area

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guarded area, then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in Table 5-4. Depending on the mask bits MSKWDGxL and MSKWDGxH in the WTIMR, an interrupt is generated on threshold violation.

Table 5-4. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRL <= converted data <= THRH

The channel on which the analog watchdog is to be applied is selected by the THRCH field in the TRC registers. The analog watchdog is enabled by setting the corresponding THREN bit in the same register.

The lower and higher threshold values for the analog watchdog are programmed using the registers THRHLR.

For example, if channel number 3 is to be monitored with threshold values in THRHLR1, then the THRCH field is programmed in the TRC1 register to select channel number 3.

A set of threshold registers (THRHLRx and TRCx) can be linked only to a single channel for a particular THRCH value. If another channel is to be monitored with the same threshold values, then the THRCH field in the TRCx register has to be programmed again.

NOTE

If the higher threshold for the analog watchdog is programmed lower than the lower threshold and the converted value is less than the lower threshold, then the WDGxL interrupt for the low threshold violation is set. Otherwise, if the converted value is greater than the lower threshold (consequently also greater than the higher threshold), then the interrupt WDGxH for high threshold violation is set. Thus, the user should avoid that situation as it could lead to misinterpretation of the watchdog interrupts.

5.3.5 DMA functionality

A DMA request can be programmed after the conversion of every channel by setting the respective masking bit in the DMAR registers. The DMAR masking registers must be programmed before starting any conversion. There is one DMAR per channel type.

The DMA transfers can be enabled using the DMAEN bit of DMAE register. When the DCLR bit of DMAE register is set, then the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

5.3.6 Interrupts

The ADC generates the following maskable interrupt signals:

- ADC_EOC interrupt requests
  - EOC (end of conversion)
  - ECH (end of chain)
  - JEOC (end of injected conversion)
  - JECH (end of injected chain)
- WDGxL and WDGxH (watchdog threshold) interrupt requests

Interrupts are generated during the conversion process to signal events such as End Of Conversion, as explained in the register description for CEOCFR. Two 7-bit registers named CEOCFR (Channel Pending Registers) and IMR (Interrupt Mask Register) are provided in order to check and enable the interrupt request to EIC module.

Interrupts can be individually enabled on a channel-by-channel basis by programming the CIMR (Channel Interrupt Mask Register).

Several Channel Interrupt Pending Registers are also provided in order to signal which of the channels' measurement has been completed.

The analog watchdog interrupts are handled by two 8-bit registers, WTISR (Watchdog Threshold Interrupt Status Register) and WTIMR (Watchdog Threshold Interrupt Mask Register), in order to check and enable the interrupt request to the EIC module. The Watchdog interrupt source sets two pending bits—WDG*x*H and WDG*x*L—in the WTISR for each of the four channels being monitored.

The CEOCFR contains the interrupt pending request status. If the user wants to clear a particular interrupt event status, then writing a 1 to the corresponding status bit clears the pending interrupt flag (at this write operation all the other bits of the CEOCFR must be maintained at 0).

5.3.7 External decode signals delay

The ADC provides several external decode signals to select which external channel has to be converted. In order to take into account the control switching time of the external analog multiplexer, a Decode Signals Delay register (DSDR) is provided. The delay between the decoding signal selection and the actual start of conversion can be programmed by writing the field DSD[0:7]. When this programmed delay is taking place, the ADCSTATUS[0:2] field in the Main Status Register (MSR) will display the value 010 (wait state).

5.3.8 Power-down mode

The analog part of the ADC can be put in low-power mode by setting the PWDN bit in the MCR. After releasing the reset signal the ADC analog module is kept in power-down mode by default, so this state must be exited before starting any operation by resetting the appropriate bit in the MCR.

The power-down mode can be requested at any time by setting the PWDN bit in the MCR. If a conversion is ongoing, the ADC hard macrocell cannot immediately enter the power-down mode. In fact, the ADC enters power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit.

Bit ADCSTATUS[0] in the MSR is set only when ADC enters power-down mode.

After the power-down phase is completed the process ongoing before the power-down phase must be restarted manually (by setting the appropriate START bit).

Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidde

5.3.9 Auto-clock-off mode

To reduce power consumption during the Idle mode of operation (without going into power-down mode), an "auto-clock-off" feature can be enabled by setting the ACKO bit in the MCR. When enabled, the analog clock is automatically switched off when no operation is ongoing, that is, no conversion is programmed by the user.

5.4 Register descriptions

5.4.1 Introduction

Table 5-5 lists the ADC registers with their address offsets and reset values.

Table 5-5. ADC digital registers

Register name	Address offset (hex)	Reset value
Main Configuration Register (MCR)	000	0x0000_0001
Main Status Register (MSR)	004	0x0000_0001
Reserved	008–00C	—
Interrupt Status Register (ISR)	010	0x0000_0000
Reserved	014	—
Channel Pending Register (CEOCFR1)	018	0x0000_0000
Channel Pending Register (CEOCFR2)	01C	0x0000_0000
Interrupt Mask Register (IMR)	020	0x0000_0000
Reserved	024	—
Channel Interrupt Mask Register (CIMR1)	028	0x0000_0000
Channel Interrupt Mask Register (CIMR2)	02C	0x0000_0000
Watchdog Threshold Interrupt Status Register (WTISR)	030	0x0000_0000
Watchdog Threshold Interrupt Mask Register (WTIMR)	034	0x0000_0000
Reserved	038–03C	—
DMA Enable Register (DMAE)	040	0x0000_0000
Reserved	044	—
DMA Channel Select Register 1 (DMAR1)	048	0x0000_0000
DMA Channel Select Register 2 (DMAR2)	04C	0x0000_0000
Threshold Control Register 0 (TRC0)	050	0x0000_0000
Threshold Control Register 1 (TRC1)	054	0x0000_0000
Threshold Control Register 2 (TRC2)	058	0x0000_0000
Threshold Control Register 3 (TRC3)	05C	0x0000_0000
Threshold Register 0 (THRHLR0)	060	0x03FF_0000
Threshold Register 1 (THRHLR1)	064	0x03FF_0000
Threshold Register 2 (THRHLR2)	068	0x03FF_0000
Threshold Register 3 (THRHLR3)	06C	0x03FF_0000
Reserved	070–094	—
Conversion Timing Register 1 (CTR1)	098	0x0000_0203

Table 5-5. ADC digital registers (continued)

Register name	Address offset (hex)	Reset value
Conversion Timing Register 2 (CTR2)	09C	0x0000_0203
Reserved	0A0–0A4	—
Normal Conversion Mask Register 1 (NCMR1)	0A8	0x0000_0000
Normal Conversion Mask Register 2 (NCMR2)	0AC	0x0000_0000
Reserved	0B0–0B4	—
Injected Conversion Mask Register 1 (JCMR1)	0B8	0x0000_0000
Injected Conversion Mask Register 2 (JCMR2)	0BC	0x0000_0000
Reserved	0C0	—
Decode Signals Delay Register (DSDR)	0C4	0x0000_0000
Power-down Exit Delay Register (PDEDR)	0C8	0x0000_0000
Reserved	0CC–17C	—
Channel 32 Data Register (CDR32)	180	0x0000_0000
Channel 33 Data Register (CDR33)	184	0x0000_0000
Channel 34 Data Register (CDR34)	188	0x0000_0000
Channel 35 Data Register (CDR35)	18C	0x0000_0000
Channel 36 Data Register (CDR36)	190	0x0000_0000
Channel 37 Data Register (CDR37)	194	0x0000_0000
Channel 38 Data Register (CDR38)	198	0x0000_0000
Channel 39 Data Register (CDR39)	19C	0x0000_0000
Channel 40 Data Register (CDR40)	1A0	0x0000_0000
Channel 41 Data Register (CDR41)	1A4	0x0000_0000
Channel 42 Data Register (CDR42)[1]	1A8	0x0000_0000
Channel 43 Data Register (CDR43)[1]	1AC	0x0000_0000
Channel 44 Data Register (CDR44)[1]	1B0	0x0000_0000
Channel 45 Data Register (CDR45)[1]	1B4	0x0000_0000
Channel 46 Data Register (CDR46)[1]	1B8	0x0000_0000
Channel 47 Data Register (CDR47)[1]	1BC	0x0000_0000
Reserved	1C0–1FC	—
Channel 64 Data Register (CDR64)[1]	200	0x0000_0000
Channel 65 Data Register (CDR65)[1]	204	0x0000_0000
Channel 66 Data Register (CDR66)[1]	208	0x0000_0000
Channel 67 Data Register (CDR67)[1]	20C	0x0000_0000
Channel 68 Data Register (CDR68)[1]	210	0x0000_0000

Table 5-5. ADC digital registers (continued)

Register name	Address offset (hex)	Reset value
Channel 69 Data Register (CDR69)[1]	214	0x0000_0000
Channel 70 Data Register (CDR70)[1]	218	0x0000_0000
Channel 71 Data Register (CDR71)[1]	21C	0x0000_0000
Reserved	220–2FC	—

1 The MPC5606S supports either 16 dedicated ADC channels (ch32 – ch47), or 15 dedicated channels (ch32 – ch41 plus ch43 – ch47) and 8 extended channels (ch64 – ch71).

When any of the extended channels is used, ch42 is irrelevant because the single pin PC[10] shares the functionality of either dedicated ch42 or all 8 extended channels of analog input coming from the output of the external analog multiplexer, as shown in Figure 5-1.

Table 5-6. Bit access descriptions

Access type	Description
read/write (rw)	Software can read and write to these bits.
read-only (r)	Software can only read these bits.
write-only (w)	Software can only write to these bits.
write 1 to clear (w1c)	Software can clear bits by writing 1.

5.4.2 Control logic registers

5.4.2.1 Main Configuration Register (MCR)

The Main Configuration Register (MCR) provides configuration settings for the ADC.

Address: Base + 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWREN	WLSIDE	MODE	EDGLEV	TRGEN	EDGE	0	NSTART	0	JTRGEN	JEDGE	JSTART	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	ADCLK SEL	ABORT CHAIN	ABORT	ACKO	0	0	0	0	PWDN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 5-5. Main Configuration Register (MCR)

Table 5-7. MCR field descriptions

Field	Description
0 OWREN	Overwrite enable This bit enables or disables the functionality to overwrite unread converted data. 0 Prevents overwrite of unread converted data; new result is discarded. 1 Enables converted data to be overwritten by a new conversion.
1 WLSIDE	Write left/right-aligned 0 The conversion data is written right-aligned. 1 Data is left-aligned (from 15 to (15 – resolution + 1)).
2 MODE	One Shot/Scan 0 One Shot mode—Configures the normal conversion of one chain. 1 Scan mode—Configures continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.
3 EDGLEV	Edge or level selection for external start trigger 0 Edge configuration for external trigger usage. 1 Level configuration for external trigger usage.
4 TRGEN	External trigger enable. This bit must be set to use external triggering to start a conversion. 0 An external trigger cannot be used to start a conversion. 1 An external trigger can start a conversion.

Table 5-7. MCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>5
EDGE</td><td>Start trigger edge/ level detection. The following table shows the interaction between the EDGE bit and the TRGEN and EDGLEV bits.
<table>
<tr><th>TRGEN</th><th>EDGLEV</th><th>EDGE</th><th>Trigger Detection</th></tr>
<tr><td>0</td><td>n</td><td>n</td><td>External triggering disabled</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>External trigger on falling edge of trigger</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>External trigger on rising edge of trigger</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>External trigger on low edge of trigger</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>External trigger on high edge of trigger</td></tr>
</table></td></tr>
<tr><td>6</td><td>Reserved
Must be kept at 0.</td></tr>
<tr><td>7
NSTART</td><td>Normal Start conversion
Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation.
This bit stays high while the conversion is ongoing (or pending during injection mode).
0 Causes the current chain conversion to finish and stops the operation.
1 Starts the chain or scan conversion.</td></tr>
<tr><td>8</td><td>Reserved
A write of any value has no effect. The read value is always 0.</td></tr>
<tr><td>9
JTRGEN</td><td>Injection external trigger enable
0 External trigger disabled for channel injection (injected conversion cannot be started using an external signal).
1 External trigger enabled for channel injection.</td></tr>
<tr><td>10
JEDGE</td><td>Injection trigger edge selection
Edge selection for external trigger, if JTRGEN = 1.
0 Selects falling edge for the external trigger.
1 Selects rising edge for the external trigger.</td></tr>
<tr><td>11
JSTART</td><td>Injection start
Setting this bit will start the configured injected analog channels to be converted by software.
Resetting this bit has no effect, as the injected chain conversion cannot be interrupted.</td></tr>
<tr><td>12–13</td><td>Reserved
A write of any value has no effect. The read value is always 0.</td></tr>
<tr><td>14</td><td>Reserved
Must be kept at 0.</td></tr>
<tr><td>15–22</td><td>Reserved
A write of any value has no effect. The read value is always 0.</td></tr>
<tr><td>23
ADCLKSEL</td><td>Analog clock frequency selector
If this bit is set the AD_clk frequency is equal to ipg_clk frequency. Otherwise, it is half of ipg_clk frequency. This bit can be written in power-down only.</td></tr>
</table>

Table 5-7. MCR field descriptions (continued)

Field	Description
24 ABORTCHAIN	Abort Chain When this bit is set, the ongoing Chain Conversion is aborted. This bit is reset by hardware as soon as a new conversion is requested. 0 Conversion is not affected. 1 Aborts the ongoing chain conversion.
25 ABORT	Abort Conversion When this bit is set, the ongoing conversion is aborted and a new conversion is invoked. This bit is reset by hardware as soon as a new conversion is invoked. 0 Conversion is not affected. 1 Aborts the ongoing conversion. **Note:** If the abort pulse is valid in the last cycle of the SAMPLE phase, the current channel is correctly aborted but the data register (CDR[0..15]) of the next channel conversion shows an invalid value.
26 ACKO	Auto-clock-off enable If set, this bit enables the Auto clock off feature. 0 Auto clock off disabled. 1 Auto clock off enabled.
27–28	Reserved Must be kept at 0.
29–30	Reserved A write of any value has no effect. The read value is always 0.
31 PWDN	Power-down enable When this bit is set, the analog module is requested to enter Power Down mode. When ADC status is PWDN, resetting this bit starts ADC transition to Idle mode. 0 ADC is in normal mode. 1 ADC has been requested to power down.

5.4.2.2 Main Status Register (MSR)

The Main Status Register (MSR) provides status bits for the ADC.

Address: Base + 0x0004 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	N START	J ABORT	0	0	J START	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR[0:6]							0	0	0	ACK0	0	0	ADCSTATUS[0:2]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 5-6. Main Status Register (MSR)

Table 5-8. MSR field descriptions

Field	Description
0–6	Reserved A write of any value has no effect. The read value is always 0.
7 NSTART	This status bit signals that a normal conversion is ongoing. This bit stays high while the conversion is ongoing (or pending during injection mode).
8 JABORT	This status bit signals that an injected conversion has been aborted. This bit is reset when a new injected conversion starts.
9–10	Reserved A write of any value has no effect. The read value is always 0.
11 JSTART	This status bit signals that an injected conversion is ongoing.
12–14	Reserved A write of any value has no effect. The read value is always 0.
15	Reserved A write of any value has no effect. The read value is always 0.
16–22 CHADDR[0:6]	Channel under measure address This status bit signals which channel is under measure.
23–25	Reserved A write of any value has no effect. The read value is always 0.
26 ACKO	Auto-clock-off enable This status bit signals if the Auto-clock-off feature is on.
27–28	Reserved A write of any value has no effect. The read value is always 0.

Table 5-8. MSR field descriptions (continued)

Field	Description
29–31 ADCSTATUS[0:2]	The value of this parameter depends on ADC status: 000 Idle 001 Power-down 010 Wait state 011 — 100 Sample 101 — 110 Conversion 111 —

5.4.3 Interrupt registers

5.4.3.1 Interrupt Status Register (ISR)

The Interrupt Status Register (ISR) contains interrupt status bits for the ADC.

Address: Base + 0x0010 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	EOC	ECH
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-7. Interrupt Status Register (ISR)

Table 5-9. ISR field descriptions

Field	Description
0–24	Reserved A write of any value has no effect. The read value is always 0.
25–26	Reserved A write of any value has no effect. The read value is always 0.
27	Reserved A write of any value has no effect. The read value is always 0.
28 JEOC	End of Injected Channel Conversion interrupt (JEOC) flag It is the interrupt of the digital end of conversion for the injected channel; active when set. When this bit is set, a JEOC interrupt has occurred.

Table 5-9. ISR field descriptions (continued)

Field	Description
29 JECH	End of Injected Chain Conversion interrupt (JECH) flag It is the interrupt of the digital end of chain conversion for the injected channel; active when set. When this bit is set, a JECH interrupt has occurred.
30 EOC	End of Channel Conversion interrupt (EOC) flag It is the interrupt of the digital end of conversion. When this bit is set, an EOC interrupt has occurred.
31 ECH	End of Chain Conversion interrupt (ECH) flag It is the interrupt of the digital end of chain conversion. When this bit is set, an ECH interrupt has occurred.

5.4.3.2 Channel Pending Registers (CEOCFR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-5.

CEOCFR1 = End of conversion pending interrupt for channel 32 to 63 (extended internal channels)

CEOCFR2 = End of conversion pending interrupt for channel 64 to 95 (external channels)

Address: Base + 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH63	EOC_CH62	EOC_CH61	EOC_CH60	EOC_CH59	EOC_CH58	EOC_CH57	EOC_CH56	EOC_CH55	EOC_CH54	EOC_CH53	EOC_CH52	EOC_CH51	EOC_CH50	EOC_CH49	EOC_CH48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH47	EOC_CH46	EOC_CH43	EOC_CH44	EOC_CH43	EOC_CH42	EOC_CH41	EOC_CH40	EOC_CH39	EOC_CH38	EOC_CH37	EOC_CH36	EOC_CH35	EOC_CH34	EOC_CH33	EOC_CH32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-8. Channel Pending Register 1 (CEOCFR1)

Address: Base + 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH95	EOC_CH94	EOC_CH93	EOC_CH92	EOC_CH91	EOC_CH90	EOC_CH89	EOC_CH88	EOC_CH87	EOC_CH86	EOC_CH85	EOC_CH84	EOC_CH83	EOC_CH82	EOC_CH81	EOC_CH80
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH79	EOC_CH78	EOC_CH77	EOC_CH76	EOC_CH75	EOC_CH74	EOC_CH73	EOC_CH72	EOC_CH71	EOC_CH70	EOC_CH69	EOC_CH68	EOC_CH67	EOC_CH66	EOC_CH65	EOC_CH64
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-9. Channel Pending Register 2 (CEOCFR2)

Table 5-10. CEOFR*n* field descriptions

Field	Description
EOC_CH *n*	This field indicates the end of conversion. 0 The measure of channel *n* is not complete. 1 The measure of channel *n* is complete.

5.4.3.3 Interrupt Mask Register (IMR)

The Interrupt Mask Register (IMR) contains the interrupt enable bits for the ADC.

Address: Base + 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	MSK JEOC	MSK JECH	MSK EOC	MSK ECH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-10. Interrupt Mask Register (IMR)

Table 5-11. IMR field descriptions

Field	Description
0:24	Reserved A write of any value has no effect. The read value is always 0.
25–26	Reserved Must be kept at 0.
27	Reserved Must be kept at 0.
28 MSKJEOC	Mask bit for JEOC When set, the JEOC interrupt is enabled.
29 MSKJECH	Mask bit for JECH When set, the JECH interrupt is enabled.
30 MSKEOC	Mask bit for EOC When set, the EOC interrupt is enabled.
31 MSKECH	Mask bit for ECH When set, the ECH interrupt is enabled.

5.4.3.4 Channel Interrupt Mask Register (CIMR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-5.

CIMR1 = Enable bits for channels 32 to 63 (extended internal channels)

CIMR2 = Enable bits for channels 64 to 95 (external channels)

Address: Base + 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIM 63	CIM 62	CIM 61	CIM 60	CIM 59	CIM 58	CIM 57	CIM 56	CIM 55	CIM 54	CIM 53	CIM 52	CIM 51	CIM 50	CIM 49	CIM 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM 47	CIM 46	CIM 43	CIM 44	CIM 43	CIM 42	CIM 41	CIM 40	CIM 39	CIM 38	CIM 37	CIM 36	CIM 35	CIM 34	CIM 33	CIM 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-11. Channel Interrupt Mask Register 1 (CIMR1)

Address: Base + 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIM 95	CIM 94	CIM 93	CIM 92	CIM 91	CIM 90	CIM 89	CIM 88	CIM 87	CIM 86	CIM 85	CIM 84	CIM 83	CIM 82	CIM 81	CIM 80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM 79	CIM 78	CIM 77	CIM 76	CIM 75	CIM 74	CIM 73	CIM 72	CIM 71	CIM 70	CIM 69	CIM 68	CIM 67	CIM 66	CIM 65	CIM 64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-12. Channel Interrupt Mask Register 2 (CIMR2)

Table 5-12. CIMR*n* field descriptions

Field	Description
CIM*n*	This field enables the interrupt for channel *n*. 0 Interrupt for channel *n* is disabled. 1 Interrupt for channel *n* is enabled.

5.4.3.5 Watchdog Threshold Interrupt Status Register (WTISR)

Address: Base + 0x0030 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	WDG 3H	WDG 2H	WDG 1H	WDG 0H	WDG 3L	WDG 2L	WDG 1L	WDG 0L
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-13. Watchdog Threshold Interrupt Status Register (WTISR)

Table 5-13. WTISR field descriptions

Field	Description
0–23	Reserved A write of any value has no effect. The read value is always 0.
24–27 WDG*x*H	This field indicates whether an interrupt has been generated because the converted value is higher than the programmed higher threshold. 0 Converted value is lower or equal to the programmed higher threshold (no interrupt generated). 1 Converted value is higher than the programmed higher threshold (interrupt is generated).
28–31 WDG*x*L	This field indicates whether an interrupt has been generated because the converted value is lower than the programmed lower threshold. 0 Converted value is higher or equal to the programmed lower threshold (no interrupt generated). 1 Converted value is lower than the programmed lower threshold (interrupt is generated).

5.4.3.6 Watchdog Threshold Interrupt Mask Register (WTIMR)

Reset value: 0x0000_0000

Address: Base + 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MSK WDG 3H	MSK WDG 2H	MSK WDG 1H	MSK WDG 0H	MSK WDG 3L	MSK WDG 2L	MSK WDG 1L	MSK WDG 0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-14. Watchdog Threshold Interrupt Mask Register (WTIMR)

Table 5-14. WTIMR field descriptions

Field	Description
0–23	Reserved A write of any value has no effect. The read value is always 0.
24–27 MSKWDG*x*H	Mask bit for the interrupt generated because the converted value is higher than the programmed higher threshold. 0 Interrupt is not enabled. 1 Interrupt is enabled.
28–31 MSKWDG*x*L	Mask bit for the interrupt generated because the converted value is lower than the programmed lower threshold. 0 Interrupt is not enabled. 1 Interrupt is enabled.

5.4.4 DMA registers

5.4.4.1 DMA Enable Register (DMAE)

The DMA Enable (DMAE) register sets up the DMA for use with the ADC.

Address: Base + 0x0040 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-15. DMA Enable Register (DMAE)

Table 5-15. DMA Enable Register (DMAE) field descriptions

Field	Description
0–29	Reserved A write of any value has no effect. The read value is always 0.
30 DCLR	DMA clear sequence enable 0 DMA request cleared by Acknowledge from DMA controller. 1 DMA request cleared automatically before DMA occurs In this mode the DMA will not be performed.
31 DMAEN	DMA global enable 0 DMA feature disabled. 1 DMA feature enabled.

5.4.4.2 DMA Channel Select Register (DMAR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-5.

31DMAR1 = Enable bits for channels 32 to 63 (extended internal channels)

DMAR2 = Enable bits for channels 64 to 95 (external channels)

Reset value: 0x0000_0000

Address: Base + 0x0048 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA 63	DMA 62	DMA 61	DMA 60	DMA 59	DMA 58	DMA 57	DMA 56	DMA 55	DMA 54	DMA 53	DMA 52	DMA 51	DMA 50	DMA 49	DMA 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA 47	DMA 46	DMA 43	DMA 44	DMA 43	DMA 42	DMA 41	DMA 40	DMA 39	DMA 38	DMA 37	DMA 36	DMA 35	DMA 34	DMA 33	DMA 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-16. DMA Channel Select Register 1 (DMAR1)

Address: Base + 0x004C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA 95	DMA 94	DMA 93	DMA 92	DMA 91	DMA 90	DMA 89	DMA 88	DMA 87	DMA 86	DMA 85	DMA 84	DMA 83	DMA 82	DMA 81	DMA 80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA 79	DMA 78	DMA 77	DMA 76	DMA 75	DMA 74	DMA 73	DMA 72	DMA 71	DMA 70	DMA 69	DMA 68	DMA 67	DMA 66	DMA 65	DMA 64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-17. DMA Channel Select Register 2 (DMAR2)

Table 5-16. DMAR*n* field descriptions

Field	Description
DMA*n*	DMA enable 0 DMA transfer for channel *n* is disabled. 1 Channel *n* is enabled to transfer data in DMA mode.

5.4.5 Threshold registers

5.4.5.1 Introduction

These four registers are used to store the user-programmable lower and upper threshold values. The inverter bit and the mask bit to mask the interrupt are stored in the TRC registers.

5.4.5.2 Threshold Control Register (TRC*x*, *x* = [0..3])

Reset value: 0x0000_0000

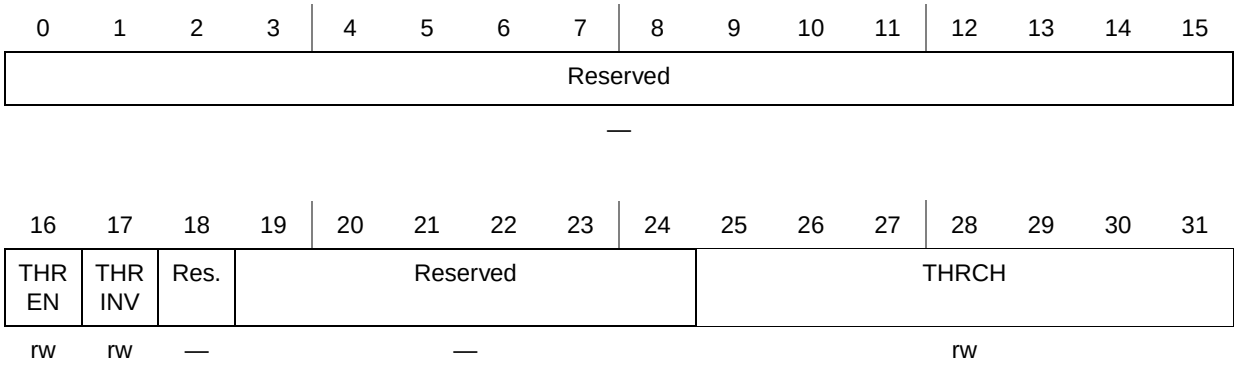


Figure 5-18. Threshold Control Register (TRCx, x = [0..3])

Table 5-17. Threshold Control Register (TRCx, x = [0..3]) field descriptions

Field	Description
0–15	Reserved A write of any value has no effect. The read value is always 0.
16 THREN	Threshold enable When set, this bit enables the threshold detection feature for the selected channel.
17 THRINV	Invert the output pin Setting this bit inverts the behavior of the threshold output pin.
18	Reserved Must be kept at 0.
19–24	Reserved A write of any value has no effect. The read value is always 0.
25–31 THRCH	Choose the channel for threshold comparison.

5.4.5.3 Threshold Register (THRHLR[0:3])

The four THRHLR*n* registers are used to store the user-programmable thresholds' 10-bit values.

Address: Base + 0x0060 (THRHLR0)
Base + 0x0064 (THRHLR1)
Base + 0x0068 (THRHLR2)
Base + 0x006C (THRHLR3)

Access: User read/write

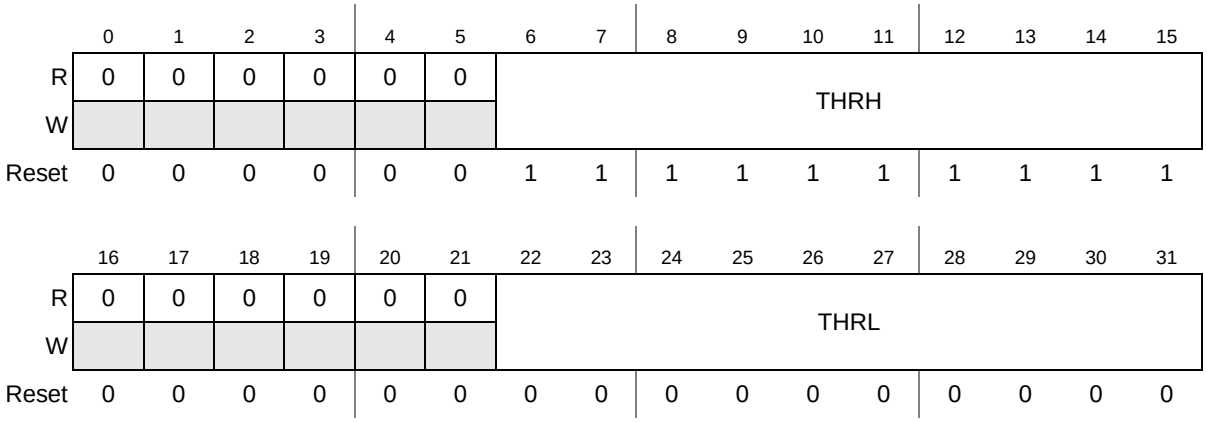


Figure 5-19. Threshold Register (THRHLR[0:3])

Table 5-18. Threshold Register (THRHLR[0:3]) field descriptions

Field	Description
0–5	Reserved A write of any value has no effect. The read value is always 0.
6–15 THRH	High threshold value for channel *n*.
16–21	Reserved A write of any value has no effect. The read value is always 0.
22–31 THRL	Low threshold value for channel *n*.

5.4.6 Conversion timing registers CTR[1..2]

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-5.

CTR1 = associated to extended internal channels (from 32 to 63)

CTR2 = associated to external channels (from 64 to 95)

Address: Base + 0x0098 (CTR1)
Base + 0x009C (CTR2)

Access: User read/write

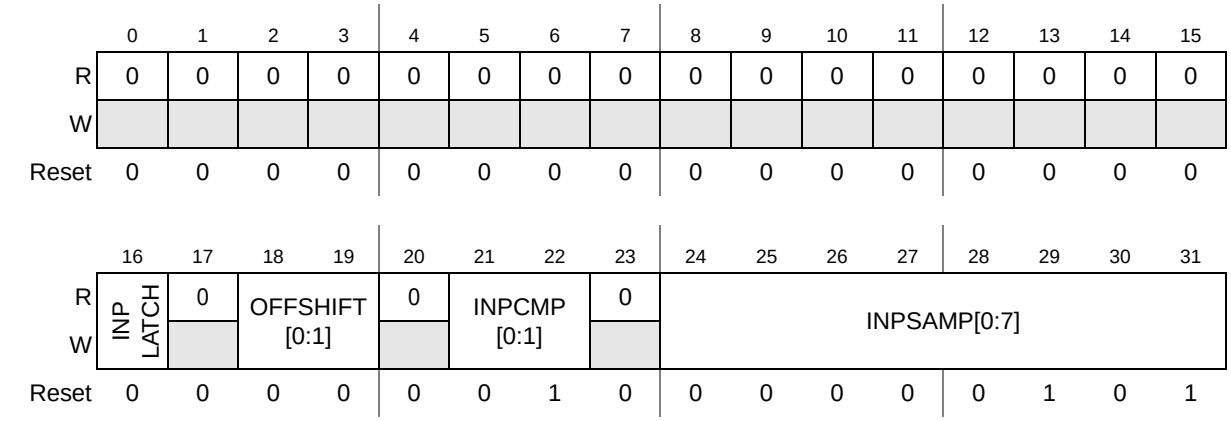


Figure 5-20. Conversion timing registers CTR[1..2]

Table 5-19. Conversion timing registers CTR[1..2] field descriptions

Field	Description
0–15	Reserved A write of any value has no effect. The read value is always 0.
16 INPLATCH	Configuration bit for latching phase duration
17	Reserved A write of any value has no effect. The read value is always 0.
18–19 OFFSHIFT[0:1]	Configuration for offset shift characteristic 00 No shift (that is the transition between codes 000h and 001h) is reached when the A_{VIN} (analog input voltage) is equal to 1 LSB. 01 Transition between code 000h and 001h is reached when the A_{VIN} is equal to1/2 LSB. 10 Transition between code 00h and 001h is reached when the A_{VIN} is equal to 0. 11 Not used.
20	Reserved A write of any value has no effect. The read value is always 0.
21–22 INPCMP[0:1]	Configuration bits for comparison phase duration.
23	Reserved A write of any value has no effect. The read value is always 0.
24–31 INPSAMP[0:7]	Configuration bits for sampling phase duration.

5.4.7 Mask registers

5.4.7.1 Introduction

These registers are used to program which of the 96 input channels must be converted during normal and injected conversion.

5.4.7.2 Normal Conversion Mask Registers (NCMR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-5.

NCMR1 = Enable bits of normal sampling for channel 32 to 63 (extended internal channels)

Reset value: 0x0000_0000

Address: Base + 0x00A8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH63	CH62	CH61	CH60	CH59	CH58	CH57	CH56	CH55	CH54	CH53	CH52	CH51	CH50	CH49	CH48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-21. Normal Conversion Mask Register 1 (NCMR1)

Table 5-20. Normal Conversion Mask Registers (NCMR[1..2]) field descriptions

Field	Description
CH*n*	Sampling enable When set, sampling is enabled for channel *n*.

5.4.7.3 Injected Conversion Mask Registers (JCMR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-5.

JCMR1 = Enable bits of injected sampling for channel 32 to 63 (extended internal channels)

JCMR2 = Enable bits of injected sampling for channel 64 to 95 (external channels)

Reset value: 0x0000_0000

Address: Base + 0x00B8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH63	CH62	CH61	CH60	CH59	CH58	CH57	CH56	CH55	CH54	CH53	CH52	CH51	CH50	CH49	CH48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-22. Injected Conversion Mask Register 1 (JCMR1)

Address: Base + 0x00BC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-23. Injected Conversion Mask Register 2 (JCMR2)

Table 5-21. Injected Conversion Mask Registers (JCMR[1..2]) field descriptions

Field	Description
CH*n*	Sampling enable When set, sampling is enabled for channel *n*.

5.4.8 Delay registers

5.4.8.1 Decode Signals Delay Register (DSDR)

Reset value: 0x0000_0000

Address: Base + 0x00C4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DSD[0:7]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-24. Decode Signals Delay Register (DSDR)

Table 5-22. Decode Signals Delay Register (DSDR) field descriptions

Field	Description
0–23	Reserved A write of any value has no effect. The read value is always 0.
24–31 DSD[0:7]	Delay between the external decode signals and the start of the sampling phase It is used to take into account the settling time of the external multiplexer. The decode signal delay is calculated as: DSD × 1/frequency of system clock. For the case when ADC clock = Peripheral Clock/2, the DSD bit field has to be incremented by 2 to see an additional ADC clock cycle delay on the decode signal. For example: 0000 0 ADC clock cycle delay. 0010 1 ADC clock cycle delay. 0100 2 ADC clock cycle delay. 0110 3 ADC clock cycle delay.

5.4.8.2 Power-down Exit Delay Register (PDEDR)

Reset value: 0x0000_0000

Address: Base + 0x00C8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PDED[0:7]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-25. Power-down Exit Delay Register (PDEDR)

Table 5-23. Power-down Exit Delay Register (PDEDR) field descriptions

Field	Description
0–23	Reserved A write of any value has no effect. The read value is always 0.
24–31 PDED[0:7]	Delay between the power-down bit reset and the start of conversion The power down delay is calculated as: PDED × 1/frequency of ADC clock.

5.4.9 Data registers

5.4.9.1 Introduction

ADC conversion results are stored in data registers. There is one register per channel.

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-5.

CDR[32..63] = Extended internal channels

CDR[64..95] = External channels

Each data register also gives information regarding the corresponding result as described below.

5.4.9.2 Channel Data Register (CDR[0..95])

Address: See Table 5-5 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CDATA[0:9]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-26. Channel Data Register (CDR[0..95])

Table 5-24. Channel Data Register (CDR[0..95]) field descriptions

Field	Description
0:11	Reserved A write of any value has no effect. The read value is always 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.
13 OVERW	Overwrite data This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, then OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, then OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten. 1 Previous converted data has been overwritten before having been read.

Table 5-24. Channel Data Register (CDR[0..95]) field descriptions (continued)

Field	Description
14–15 RESULT[0:1]	This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of normal conversion mode. 01 Data is a result of injected conversion mode. 10 Reserved. 11 Reserved.
16–21	Reserved A write of any value has no effect. The read value is always 0.
22–31 CDATA[0:9]	Channel 0-95 converted data.

Chapter 6
Boot Assist Module (BAM)

This chapter describes the Boot Assist Module (BAM).

6.1 Overview

The Boot Assist Module is a block of read-only memory containing VLE code which is executed according to the boot mode of the device.

The BAM allows you to download code into internal SRAM through the following serial protocol and execute it afterwards:

- FlexCAN (without autobaud)
- LINFlex (without autobaud)

6.2 Features

The BAM provides the following features:

- Locate and detect application boot code
- MPC5606S in static mode if internal flash is not initialized or invalid
- System can recover from Static mode only by Reset
- Programmable 64-bit password protection for serial boot mode
- Serial boot loads the application boot code from a FlexCAN or LINFlex bus into internal SRAM
- Censorship protection for internal flash module

6.3 Boot modes

The MPC5606S supports the following boot modes:

- Single-chip (SC) — The device boots from the first bootable section of the flash memory main array.
- Serial Boot (SBL) — The device downloads boot code from either LINFlex or FlexCAN interface and then executes it.

If booting is not possible with the selected configuration (e.g., if no Boot ID is found in the selected boot location) then the device enters the static mode.

6.4 Memory map

The BAM code resides in a reserved 8 KB ROM mapped from address 0xFFFF_C000.

Table 6-1. BAM memory organization

Parameter	Address
BAM entry point	0xFFFF_C000
Downloaded code base address	0x4000_0100

The address space and memory used by BAM application is shown in Table 6-1.

The RAM location where to download the code can be any 4-byte aligned location starting from the address 0x4000_0100.

6.5 Functional description

6.5.1 Entering boot modes

The MPC5606S detects the boot mode based on external pins and device status (see Figure 6-1).

To boot either from FlexCAN or LINFlex, the device must be forced into an Alternate Boot Loader mode via the FAB (Force Alternate Boot mode) pin which must be asserted before initiating the reset sequence. The type of alternate boot mode is selected according to the ABS (Alternate Boot Selector) pin (see Table 6-2).

NOTE

The watchdog (SWT) is disabled at the start of BAM execution. In the case of an unexpected issue during BAM execution, the CPU may be stalled and it will be necessary to generate an external reset to recover.

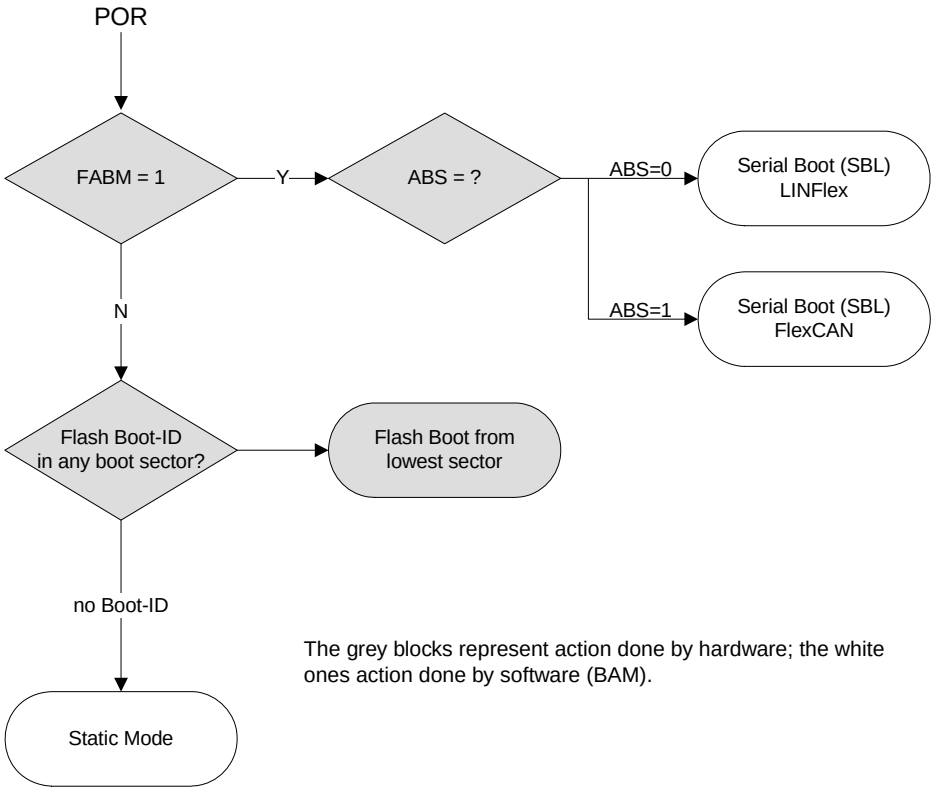


Figure 6-1. Boot mode selection

Table 6-2. Hardware configuration to select boot mode

FAB	ABS	Standby-RAM boot flag	Boot ID	Boot mode
1	0	0	—	LINFlex
1	1	0	—	FlexCAN
0	—	0	valid	SC (Single-chip)
0	—	0	not found	Static mode

6.5.2 Reset Configuration Half Word Source (RCHW)

MPC5606S flash memory is partitioned into boot sectors shown in Table 6-4.

Each boot sector contains at offset 0x00 the Reset Configuration Half-Word (RCHW).

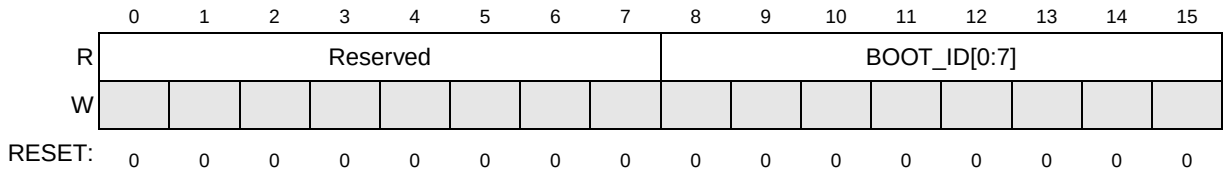


Figure 6-2. Reset Configuration Half Word (RCHW)

Table 6-3. RCHW field descriptions

Field	Description
0-7	Reserved
8-15 BOOT_ID[0:7]	Is valid if its value is 0x5A, then the sector is considered bootable

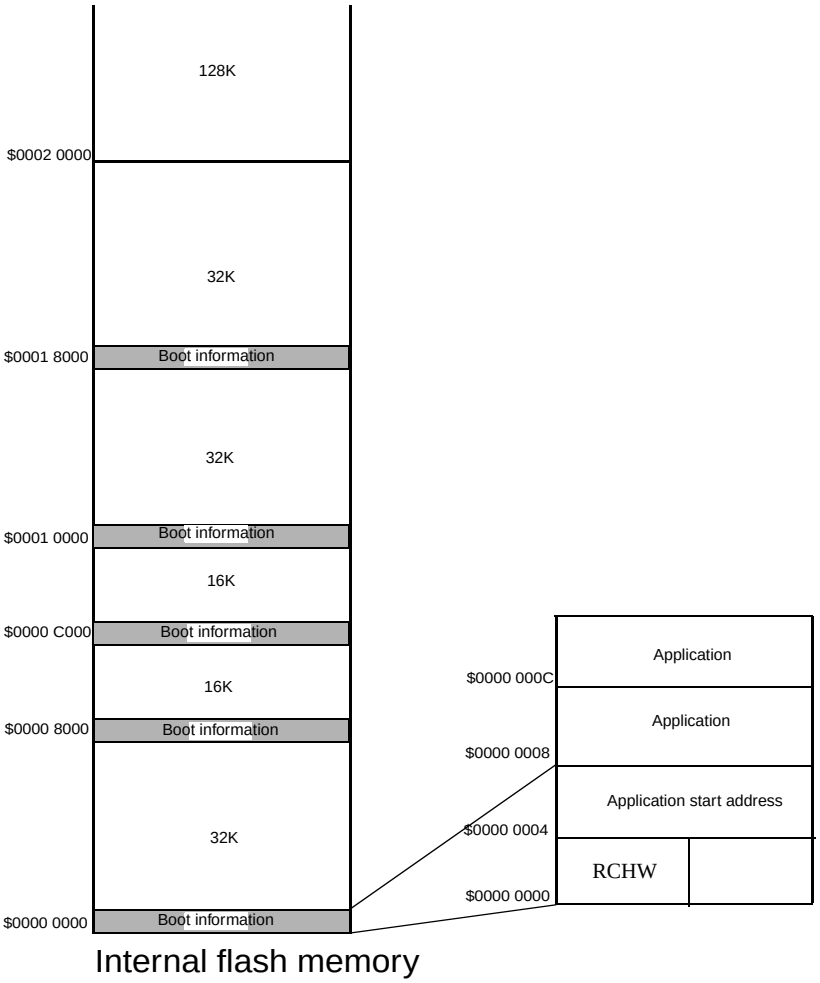


Figure 6-3. MPC5606S flash memory partitioning and RCHW search

Table 6-4. Flash boot sector

Block	Address
0	0x0000_0000
1	0x0000_8000
2	0x0000_C000
3	0x0001_0000
4	0x0001_8000

6.5.3 Single-chip boot mode

In Single-chip mode, the hardware searches flash boot sectors for a valid boot ID. As soon the device detects a bootable sector, it jumps within this sector and reads the 32-bit word at offset 0x4. This word is the address where the startup code is located (reset boot vector).

Then the device executes this startup code. A user application should have a valid instruction at the reset boot vector address.

If a valid RCHW is not found, the BAM code is executed. In this case BAM moves the MPC5606S into static mode.

6.5.3.1 Boot and alternate boot

Some applications require an alternate boot sector so that the main boot can be erased and reprogrammed in the field.

When an alternate boot is needed, the user can create two bootable sectors. The lowest sector is the main boot sector, and the highest is the alternate boot sector. The alternate boot sector does not need to be consecutive to the main boot sector.

This scheme ensures that there is always one active boot sector by erasing one of the boot sectors only:

- Sector is activated (that is, program a valid BOOT_ID instead of 0xFF as initially programmed).
- Sector is deactivated writing to 0 some of the bits BOOT-ID bit field (bit 1 and/or bit 3, and/or bit 4, and/or bit 6).

6.5.4 Boot through BAM

6.5.4.1 Executing BAM

Single-chip mode is managed by hardware and BAM does not participate.

BAM is executed only in the following two cases:

- Serial boot mode has been selected by FAB pin
- Hardware has not found a valid Boot-ID in any flash boot locations

If one of these conditions is true, the device fetches code at location 0xFFFF_C000 and BAM application starts.

6.5.4.2 BAM software flow

Figure 6-4 shows the BAM logic flow.

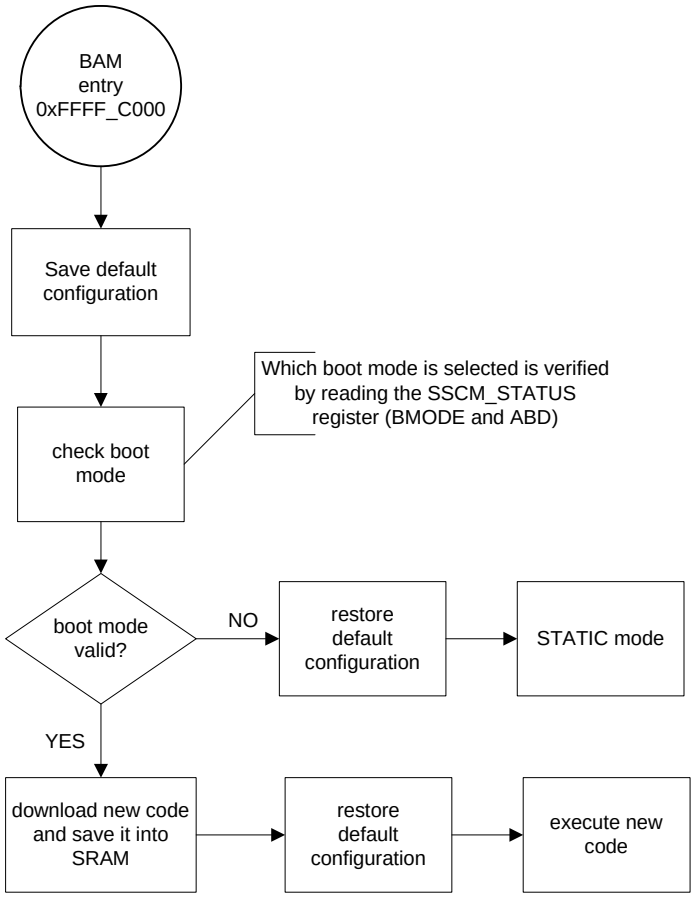


Figure 6-4. BAM logic flow

The first action is to save the initial device configuration. In this way, it is possible to restore the initial configuration after downloading the new code but before executing it. This allows the new code to execute as the device is just coming out of reset.

The BMODE and ABD fields of the SSCM STATUS register (see Section 38.2.2.1, System Status Register (STATUS)) indicate which boot has to be executed (see Table 6-5).

If BMODE field shows either a single-chip value (011) or the reserved values, the boot mode is not considered valid and the BAM pushes the device into static mode.

In all other cases the code of the relative boot is called. Data is downloaded and saved into the proper SRAM location.

Table 6-5. Fields of SSCM STATUS Register Used by BAM

Field	Description
BMODE	BMODE Device Boot mode. 000 FlexRay Boot Serial Boot Loader (future use) 001 CAN Serial Boot Loader 010 SCI Serial Boot Loader 011 Single-chip 1xx Reserved

Then, the initial device configuration is restored and the code jumps to the address of downloaded code. At this point BAM has just finished its task.

If there is any error (that is, communication error, wrong boot selected, etc.), BAM restores the default configuration and puts the device into static mode. Static mode means the device enters the low-power mode Safe and the processor executes a wait instruction. It is needed if the device cannot boot in the mode which was selected. During BAM execution and after, the mode reported by the field S_CURRENT_MODE of the register ME_GS in the module MC_ME Module is the DRUN mode.

6.5.4.3 BAM resources

BAM uses/initializes the following MCU resources:

- MC_ME and MC_CGM to initialize mode and clock sources
- FlexCAN 0, LINFlex 0, and their pads when performing serial boot mode
- SSCM to check the boot mode and during password check (see Table 6-5 and Figure 6-5)
- External oscillator
- SWT (the BAM disables it)

The following hardware resources are used only when autobaud feature is selected:

- STM to measure the baud rate
- CMU to measure the external clock frequency related to the internal RC clock source
- FMPLL to work with system clock near the maximum allowed frequency (to have higher resolution during baud rate measurement).

As already mentioned, the initial configuration is restored before executing the downloaded code.

The system clock is selected directly from the external oscillator. Thus the oscillator frequency defines baud rates for serial interfaces used to download the user application (see Table 6-6).

Table 6-6. Serial Boot mode without autobaud—baud rates

Crystal frequency (MHz)	LINFlex baud rate (baud)	CAN bit rate (bit/s)
f_{extal}	f_{extal} / 833	f_{extal} / 40
8	9600	200K
12	14400	300K

Table 6-6. Serial Boot mode without autobaud—baud rates (continued)

Crystal frequency (MHz)	LINFlex baud rate (baud)	CAN bit rate (bit/s)
16	19200	400K
20	24000	500K
40	48000	1M

6.5.4.4 Download and execute the new code

From a high-level perspective, the download protocol follows steps:

0. (optional step) Send message and receive acknowledge message for autobaud or autobit rate selection.
1. Send 64-bit password.
2. Send start address, size of downloaded code in bytes, and VLE bit[1].
3. Download data.
4. Execute code from start address.

Each step must be completed before the next step starts.

The communication is done in half-duplex manner. Any transmission from host is followed by the MCU transmission:

1. Host sends data to MCU and starts waiting.
2. MCU echoes to host the data received.
3. MCU verifies if echo is correct.
   - If data is correct, the host can continue to send data.
   - If data is not correct, the host stops transmitting and the MCU needs to be reset.

All multi-byte data structures are sent with MSB first.

A more detailed description of these steps follows.

6.5.4.5 Download 64-bit password and password check

The first 64 bits received represent the password. This password is sent to the Password Check procedure, which verifies if it is correct.

Password check data flow is shown in Figure 6-5 where:

- SSCM_STATUS.SEC = 1 means flash secured
- SSCM_STATUS.PUB = 1 means flash with public access

In case of flash memory with public access, the received password is compared with the public password 0xFEED_FACE_CAFE_BEEF.

1. Since this device supports only VLE code and it does not support Book E code, this flag is used only for backward compatibility.

If public access is not allowed but the flash is not secured, the received password is compared with the value saved on NVPWD0 and NVPWD1 registers.

In both of the previous cases, comparison is done by the BAM application. If password validation fails, the BAM pushes the MCU into static mode.

In case of public access not allowed and flash secured, the password is written into the SSCM.PWCMPH-L registers.

After a fixed waiting time, a comparison is done by hardware. Then the BAM again verifies SSCM_STATUS's SEC flag:

- SEC = 0, flash is now unsecured and BAM continues its task
- SEC = 1, flash is still secured because password was wrong; BAM puts MCU into static mode

This fixed time depends on the external crystal oscillator frequency (FXOSC). With FXOSC of 12 MHz, the fixed time is 350 ms.

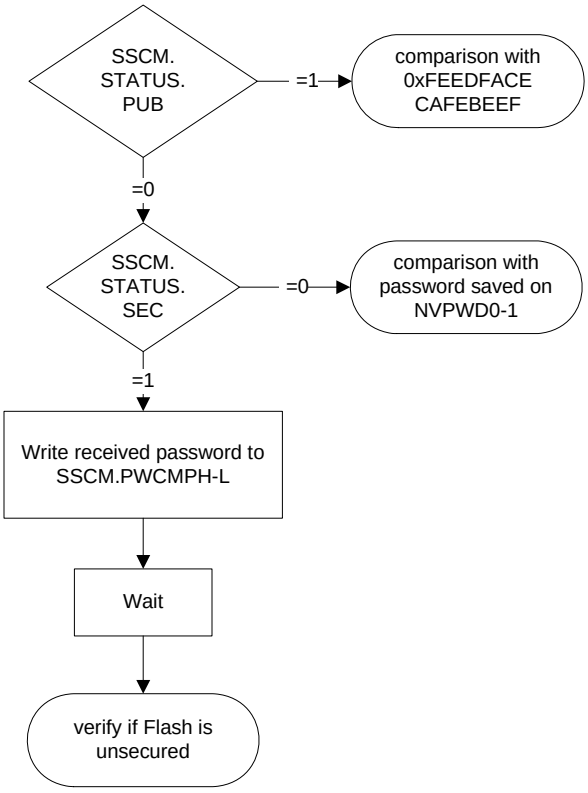


Figure 6-5. Password check flow

6.5.4.6 Download start address, VLE bit and code size

The next 8 bytes received by the MCU contain a 32-bit start address, the VLE mode bit, and a 31-bit code length, as shown in Figure 6-6.

The VLE bit (Variable Length Instruction) is used to indicate for which instruction set the code has been compiled. This device family supports only VLE = 1. This bit is used for backward compatibility.

START_ADDRESS defines where the received data will be stored and where the MCU will branch after the download is finished. The two LSB bits of START_ADDRESS are ignored by the BAM program, such that the loaded code should be 32-bit word aligned.

CODE_LENGTH defines how many data bytes have to be loaded.

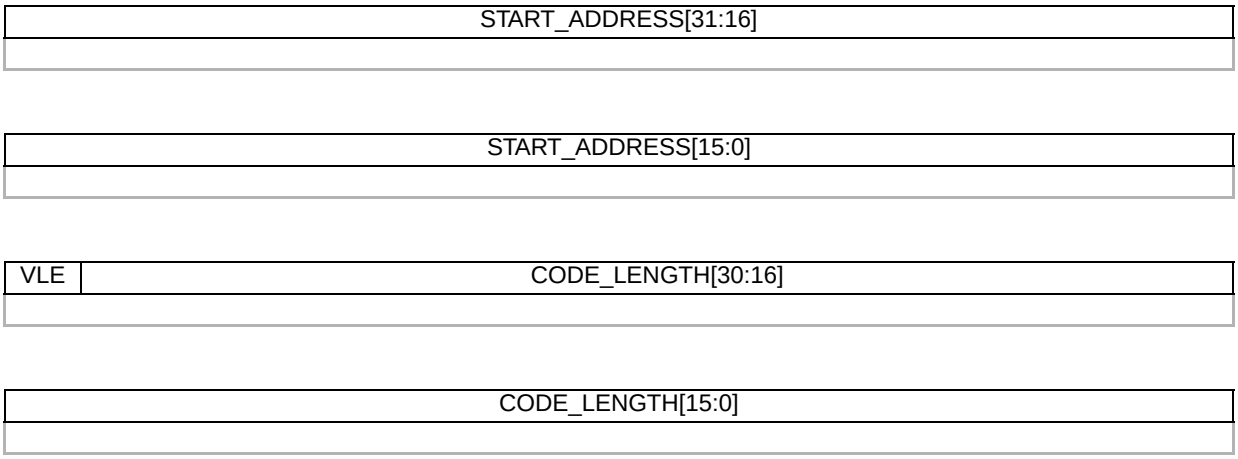


Figure 6-6. Start address, VLE bit, and download size in bytes

6.5.4.7 Download data

Each byte of data received is stored into the device's SRAM, starting from the address specified in the previous protocol step.

The address increments until the number of bytes of data received matches the number of bytes specified in the previous protocol step.

Since the SRAM is protected by a 32-bit wide Error Correction Code (ECC), BAM always writes bytes into SRAM grouped into 32-bit words. If the last byte received does not fall onto a 32-bit boundary, BAM fills it with 0 bytes.

Then a dummy word (0x0000_0000) is written to avoid ECC error during core prefetch.

6.5.4.8 Execute code

The BAM program waits for the last echo message transmission to complete.

Then it restores the initial MCU configuration and jumps to the loaded code at the start address received in step 2 of the protocol.

At this point, the BAM has finished its tasks and the MCU is controlled by new code executing from SRAM.

6.5.5 Boot from UART

6.5.5.1 Configuration

Boot from UART protocol is implemented by the LINFlex 0 module. The pins used are:

- LINFlex_TX—corresponds to pin PB[2]
- LINFlex_RX—corresponds to pin PB[3]

The system clock is driven by an external oscillator.

The LINFlex controller is configured to operate at a baud rate = system clock frequency/833 (see Table 6-6 for baud rate example), using an 8-bit data frame without parity bit and 1 stop bit.

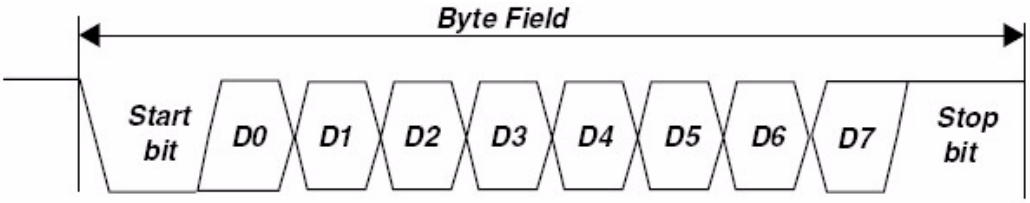


Figure 6-7. LINFlex bit timing in UART mode

6.5.5.2 Protocol

Table 6-7 summarizes the protocol and BAM action during this boot mode.

Table 6-7. UART Boot mode download protocol (autobaud disabled)

Protocol step	Host sent message	BAM response message	Action
1	64-bit password (MSB first)	64-bit password	Password checked for validity and compared against stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + 31-bit number of bytes (MSB first)	VLE bit + 31-bit number of bytes (MSB first)	Size of download is stored for future use. Verify if VLE bit is set to 1.
4	8 bits of raw binary data	8 bits of raw binary data	8-bit data are packed into 32-bit words. This word is saved in SRAM starting from the “Load address”. “Load address” increments until the amount of data received and stored matches the size as specified in the previous step.
5	None	None	Branch to downloaded code.

6.5.6 Bootstrap with CAN

6.5.6.1 Configuration

Boot from FlexCAN protocol is implemented by the FlexCAN_0 module. The pins used are:

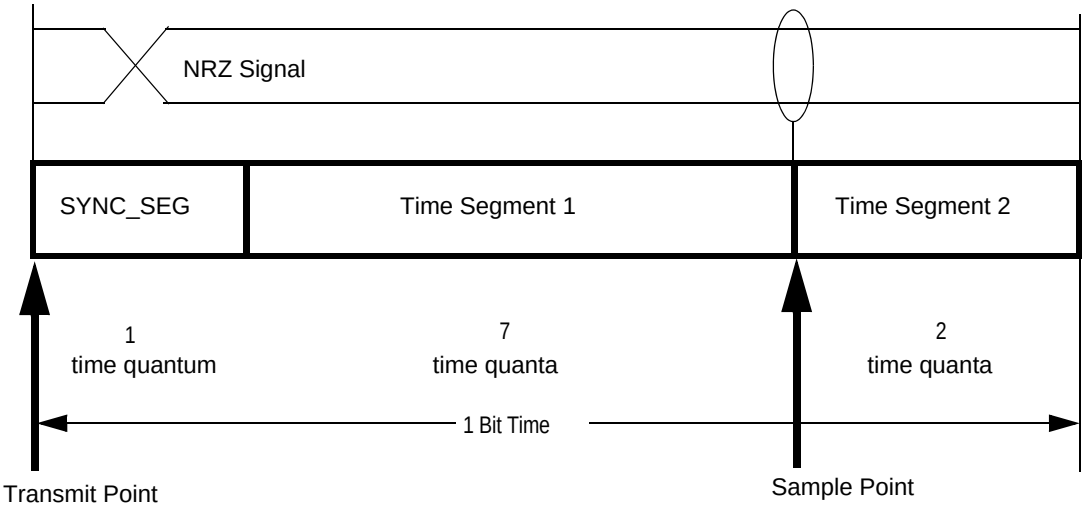
- CAN_TX—corresponds to pin PB[0]
- CAN_RX—corresponds to pin PB[1]

Boot from FlexCAN uses the system clock driven by an external oscillator.

The FlexCAN controller is configured to operate at a baud rate = system clock frequency/40 (see Table 6-6 for examples of baud rate).

It uses the standard 11-bit identifier format detailed in FlexCAN 2.0A specification.

FlexCAN controller bit timing is programmed with 10 time quanta, and the sample point is 2 time quanta before the end, as shown in Figure 6-8.



1 time quantum = 4 system clock periods

Figure 6-8. FlexCAN Bit Timing

6.6 Protocol

Table 6-8 summarizes the protocol and BAM action during this boot mode. All data are transmitted byte wise.

Table 6-8. FlexCAN Boot mode download protocol (autobaud disabled)

Protocol step	Host sent message	BAM response message	Action
1	CAN ID 0x011+ 64-bit password	CAN ID 0x001+ 64-bit password	Password checked for validity and compared against stored password.
2	CAN ID 0x012+ 32-bit store address+ VLE bit+ 31-bit number of bytes	CAN ID 0x002+ 32-bit store address+ VLE bit+ 31-bit number of bytes	Load address is stored for future use. Size of download is stored for future use. Verify if VLE bit is set to 1.
3	CAN ID 0x013+ 8 to 64 bits of raw binary data	CAN ID 0x003+ 8 to 64 bits of raw binary data	8-bit data are packed into 32-bit words. These words are saved in SRAM starting from the “Load address”. “Load address” increments until the number of data received and stored matches the size as specified in the previous step.
5	None	None	Branch to downloaded code. **Note:** Once the BAM has downloaded the code, it attempts to disable the FlexCAN module. If there is continuing traffic on the CAN bus from the boot master or other nodes, then the FlexCAN module cannot be disabled and the BAM will stall. This will prevent the downloaded code from executing and a reset will be required to recover the MCU.

Table 6-9. System clock frequency related to external clock frequency

f_{osc} [MHz]	f_{rc}/f_{osc} [1]	f_{sys} [MHz]
4–8	4–2	16–32
8–12	2–4/3	32–48
12–16	4/3–1	36–48
16–24	1–2/3	32–48
> 24	< 2/3	> 24

1 These values and consequently the f_{sys} suffer from the precision of the RC internal oscillator used to measure f_{osc} through the CMU module.

6.6.1 Flash memory password swapping

When the chip uses the BAM to boot using FlexCAN or UART, the required flash memory password is different depending on whether the flash memory is secured or unsecured. This difference affects how you must program the NVPWD0, NVPWD1, NVSCI0, and NVSCI1 registers.

When the flash memory is secured, the registers are programmed as follows:

- NVPWD0 = 0x87654321
- NVPWD1 = 0x12345678
- NVSCI0 = 0x55AA1111
- NVSCI1 = 0x55AA1111

To download the code via serial boot, the provided password is 0x1234_5678_8765_4321 (NVPWD1 followed by NVPWD0).

When the flash memory is unsecured, the registers are programmed as follows:

- NVPWD0 = 0x87654321
- NVPWD1 = 0x12345678
- NVSCI0 = 0x55AA55AA
- NVSCI1 = 0x55AA55AA

To download the code via serial boot, the provided password is 0x8765_4321_1234_5678 (NVPWD0 followed by NVPWD1).

6.6.2 Interrupts

No interrupts are generated by or are enabled by the BAM.

Chapter 7
CAN Sampler

7.1 Introduction

The CAN Sampler peripheral has been designed to store the first identifier of CAN message detected on the CAN bus while no precise clock (crystal) is running at that time on the device, typically in low-power modes (Stop, Halt or Standby) or in Run mode with the crystal switched off.

Depending on both the CAN baud rate and which low-power mode is used, it is possible to catch either the first or the second CAN frame by sampling one of two CAN Rx ports and storing all samples in internal registers.

After selection of the mode (first or second frame), the CAN Sampler stores samples of the 48 bits or skips the first frame and stores samples of the 48 bits of the second frame using the 16 MHz IRC oscillator and the 5-bit clock prescaler.

After completion, software has to process the sampled data in order to rebuild the 48 minimal bits.

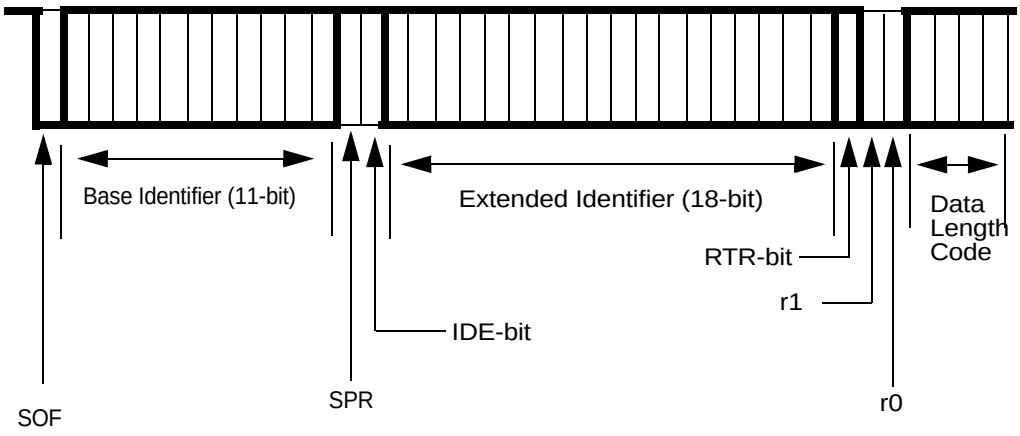


Figure 7-1. Extended CAN data frame

7.2 Main features

- Store 384 samples, equivalent to 48 CAN bits at 8 samples/bit
- Sample frequency from 500 kHz up to 16 MHz, equivalent at 8 samples/bit to CAN baud rates of 62.5 kbit/s to 2 Mbit/s
- User selectable CAN Rx sample port, CAN0RX or CAN1RX
- 16 MHz IRC clock
- 5-bit clock prescaler
- Configurable trigger mode (immediate, next frame)
- Flexible sample processing by software
- Very low-power consumption

7.3 Register description

The CAN registers are listed in Table 7-1.

Table 7-1. CAN registers

Register Name	Address Offset	Reset Value	Location
Control Register (CR)	00h	0000 0000h	on page 182
Sample registers 0	04h	xxxx xxxxh[1]	on page 183
Sample register 1	08h	xxxx xxxxh[1]	on page 183
.....	....	....	
.....	....	....	
Sample register 11	30h	xxxx xxxxh[1]	on page 183

[1] The initialization data is unknown. They will be filled only after first CAN sampling.

7.3.1 CAN Sampler Control Register

Address Offset: 0x00 Reset value: 0000 0000h

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX_COMPLETE	BUSY	Active_CK	0	0	0	MODE	CAN_RX_SEL			BRP					CAN_SMPLR_EN
W																

Figure 7-2. Control Register (CR)

Table 7-2. Control Register (CR) field description

0-15	Reserved
16 RX_COMPLETE	0 CAN frame has not been stored in the sample registers. 1 CAN frame is stored in the sample registers.
17 BUSY	This bit indicates the status of sampling. 0 Sampling is complete or has not started. 1 Sampling is ongoing.
18 Active_CK	This bit indicates which is current clock for sample registers, xmem_ck. 0 ipg_clk_s is currently xmem_ck 1 RC_CLK is currently xmem_ck. .
19-21	Reserved These are reserved bits. These bits are always read as 0.
22 MODE	0:Skip the first frame and sample and store the second frame (SF_MODE). 1:Sample and store the first frame (FF_MODE).
23-25 CAN_RX_SEL	These bits determine which RX bit is sampled. 000 Rx0 is selected 001 Rx1 is selected 010 Rx2 is selected 011 Rx3 is selected 100 Rx4 is selected 101 Rx5 is selected (not valid on this device) 110 Rx6 is selected (not valid on this device) 111 Rx7 is selected (not valid on this device)
26-30 BRP	Baud Rate Prescaler These bits are used to set the baud rate before going into standby mode 00000 Prescaler has 1. 11111 Prescaler has 32.
31 CAN_SMPLR_EN	CAN Sampler Enable This bit enables the CAN Sampler before going into standby or stop mode. 0 CAN Sampler is disabled. 1 CAN Sampler is enabled. **Note:** When the CAN Sampler is enabled (CAN_SMPLER_EN = 1) and the peripheral is stopped by a mode transition through the Mode Entry (MC_ME) block, it remains in the stopped state even after exiting the mode. The peripheral is unavailable for further sampling until and unless either a device reset occurs, or the CAN Sampler is disabled and enabled again. If the CAN Sampler has to be used again for sampling, it needs to be disabled and enabled again by programming CAN_SMPLER_EN = 0, followed by CAN_SMPLER_EN = 1.

7.3.2 CAN Sampler Sample Registers 0–11

The CAN Sampler sample registers 0–11 have the same structure; Figure 7-3 and Figure 7-4 show this structure for registers 0 and 11, respectively.

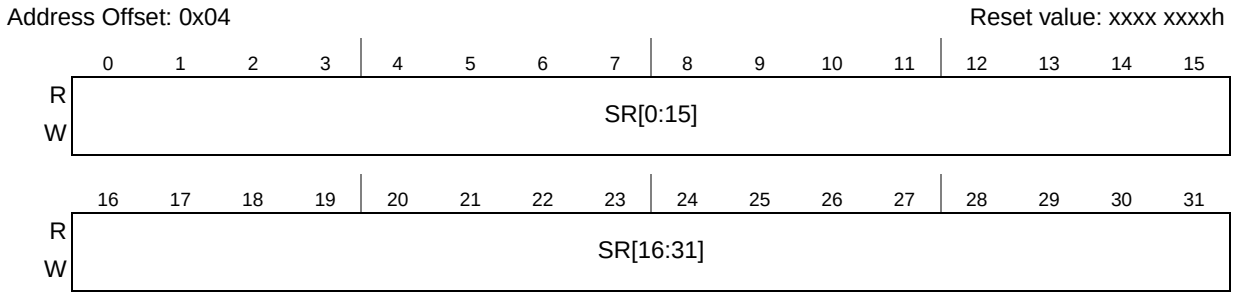


Figure 7-3. Sample Register 0

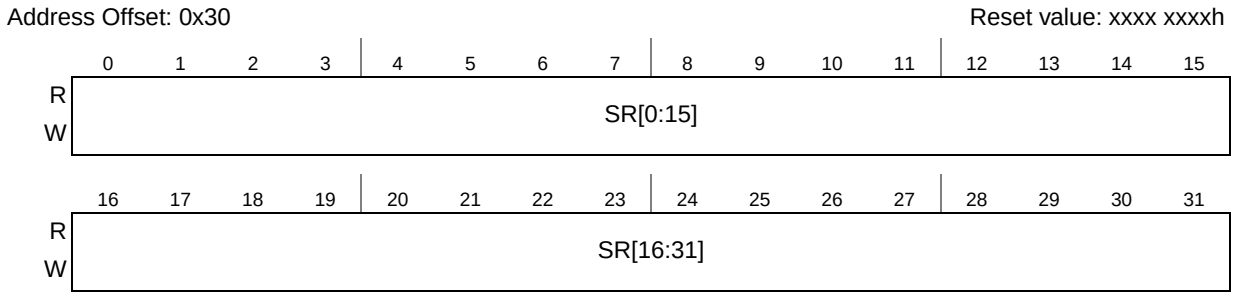


Figure 7-4. Sample Register 11

7.4 Functional description

As the CAN Sampler is driven by the 16 MHz IRC to properly sample the CAN identifier, two modes are possible, depending on both the CAN baud rate and the low-power mode used:

- Immediate sampling on falling edge detection (first CAN frame): this mode is used when the IRC 16 MHz is available in low-power mode, for example Stop or Halt.
- Sampling on next frame (second CAN frame): this mode is used when the IRC 16 MHz is switched off in low-power mode, for example Standby. Due to the startup times of both the voltage regulator and the IRC 16 MHz (~10 µs), the CAN Sampler would miss the first bits of a CAN identifier sent at 500 kbit/s. Therefore the first identifier is ignored and the sampling is performed on the first falling edge of after interframe space.

The CAN Sampler performs sampling on a user-selected CAN Rx port, normally when the device is in Standby or Stop mode storing the samples in internal registers. The user is required to configure the baud rate to achieve 8 samples per CAN nominal bit. It does not perform any sort of filtering on input samples. Thereafter, the software must enable the sampler by setting the CAN_SMPLR_EN bit in the CR register. It then becomes the master controller for accessing the internal registers implemented for storing samples.

When enabled, the CAN Sampler waits for a low pulse on the selected Rx line, taking it as a valid bit of the first CAN frame. It generates the RC wakeup request, which can be used to start the RC oscillator. Depending on the mode, it stores the first 8 samples of the 48 bits on the selected Rx line or skips the first frame and stores 8 bits for the first 48 bits of the second frame. In FF_MODE, it samples the CAN Rx line on the RC clock and stores the 8 samples of the first 48 bits (384 samples). In SF_MODE, it samples the Rx line and waits for 11 consecutive dominant bits (11 × 8 samples), taking it as the end of the first frame. It then waits for the first low pulse on the Rx line, taking it as a valid Start of Frame (SOF) of the second frame. The sampler takes 384 samples (48 bytes × 8) using the RC clock (configuring 8 samples per

nominal bit) of the second frame, including the SOF bit. These samples are stored in consecutive addresses of the (12 × 32) internal registers. RX_COMPLETE bit is set to 1, indicating that sampling is complete.

Software should now process the sampled data by first becoming master for accessing samples internal registers by resetting the CAN_SMPLR_EN bit. The sampler will need to be enabled again to start waiting for a new sampling routine.

7.4.1 Enabling/disabling the CAN Sampler

The CAN Sampler is disabled on reset and the CPU is able to access the 12 registers used for storing samples. The CAN Sampler must be enabled before entering Standby or Stop mode by setting CR[CAN_SMPLR_EN].

When the CAN Sampler is enabled, the A, D, WEN, CSN, and CK to the (12 × 32) block of registers are switched to those generated by the kernel of the sampler. You can monitor CR[Active_CK] to check which is the active clock to the registers.

If there is any activity on the selected Rx line, the sampler enables the 16 MHz RC oscillator. When CAN_SMPLR_EN is reset to 0, the sampler should receive at least three RC clock pulses to reset itself, after which the RC can be switched off.

When the software wishes to access the contents of the sample registers, it must first reset the CAN_SMPLR_EN bit by writing a 0. Before accessing the register contents it must monitor Active_CK bit for 0. When this bit is reset, it can safely access the (12 × 32) sample registers. While shifting between Normal to Sample mode, the sample register signals must be static and inactive to ensure the data is not corrupted.

7.4.2 Selecting the Rx port

One Rx port can be selected per sampling routine; the port to be sampled is selected by CAN_RX_SEL.

Table 7-3. Internal multiplexer correspondence

CAN_RX_SEL	Rx selected
000	CANRX_0 PB[1]
001	CANRX_1 PB[10]
010	CANRX_2 PF[13]
011	CANRX_3 PJ[4]
100	CANRX_4 PJ[6]
101	Reserved
110	Reserved
111	Reserved

7.4.3 Baud rate generation

Sampling is performed at a baud rate that is set by the software as a multiple of the RC oscillator frequency of 62.5 ns (assuming RC is configured for high frequency mode of 16 MHz). The user must set the baud rate prescaler (BRP) such that 8 samples per bit are achieved.

The baud rate must be set by software before going into Standby or Stop mode. This is done by setting BRP bits 5:1 in the Control register. The reset value of BRP is 00000 and can be set to a maximum of 11111, which gives a prescale value of BRP + 1, thus providing a BRP range of 1 to 32.

- Maximum bit rate supported for sampling is 2 Mbit/s using BRP of 1
- Minimum bit rate supported for sampling is 62.5 kbit/s using BRP of 32

For example, suppose the system is transmitting at 125 kbit/s. In this case, the nominal bit period is:

$$T = 1/(125 \times 10^3)s = 8 \times 10^{-3} \times 10^{-3}\ s = 8\ \mu s \qquad \textit{Eqn. 7-1}$$

To achieve 8 samples per bit:

Sample period= 8/8 µs = 1 µs

BRP = 1 µs/62.5 ns = 16. Thus in this case, BRP = 01111.

Chapter 8
Clock Description

This chapter describes the clock architectural implementation for MPC5606S.

8.1 Clock architecture

System clocks are generated from three sources:

- External oscillator FXOSC (4–16 MHz)
- High speed internal RC (16 MHz)
- FMPLL0 clocked by FXOSC, still one of system clock sources

Additionally, there are two low-power oscillators:

- Low-speed internal RC (128 kHz)
- Low-power external oscillator SXOSC (32 KHz)

Additionally, there is a secondary FMPLL:

- FMPLL1 clocked by FXOSC is available as a clock source, only for eMIOS_0, eMIOS_1, QuadSPI, and DCU modules

The clock architecture is shown in Figure 8-1.

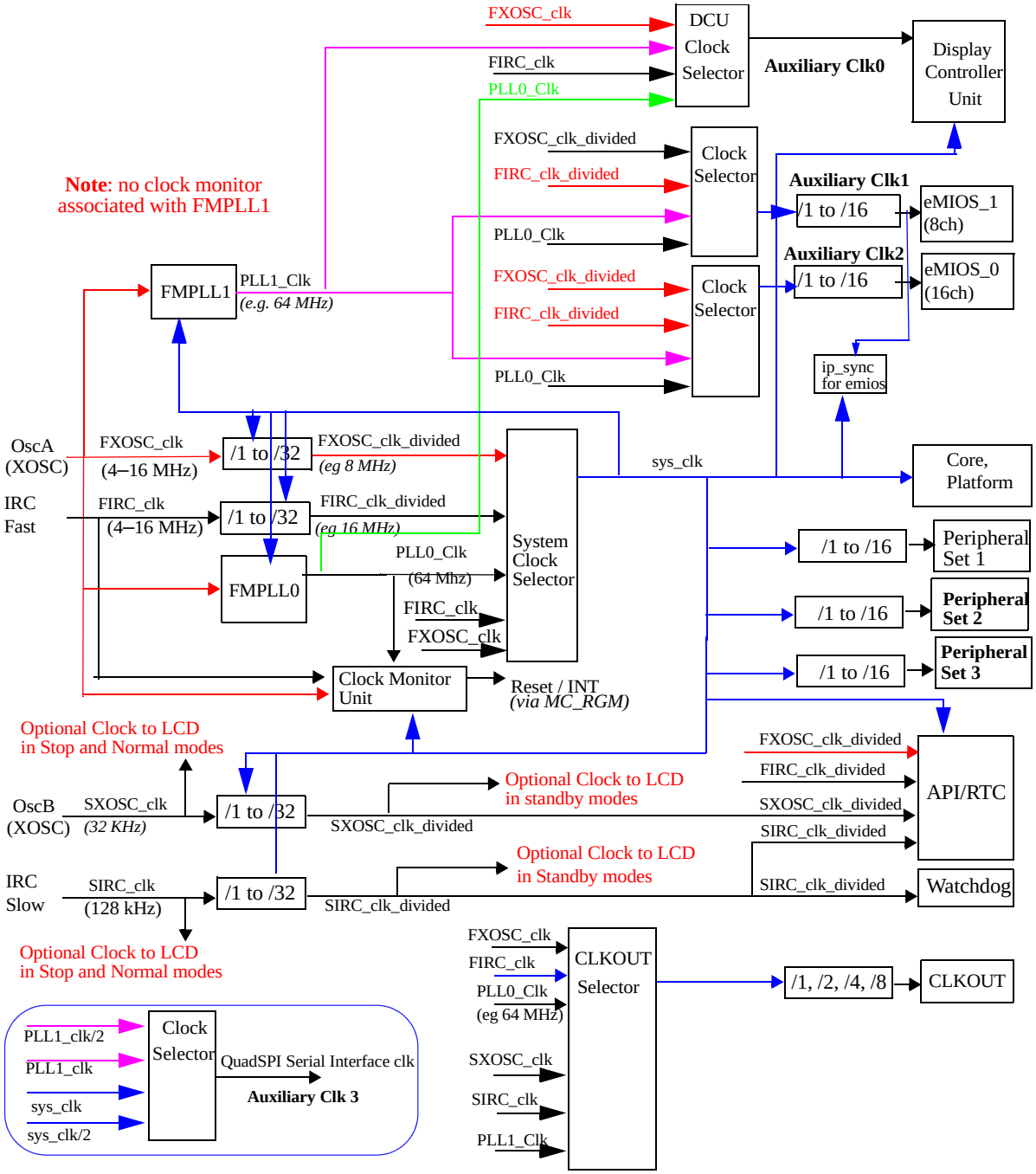


Figure 8-1. MPC5606S system clock generation

8.2 Auxiliary clocks

This device has four auxiliary clocks configurable using the MC_CGM registers. These auxiliary clocks allow the associated peripherals to operate at clock speeds independent of the system clock (sys_clk). The peripherals also use the undivided system clock to synchronously interface with the rest of the device. The auxiliary clock configuration is as follows:

- Auxiliary Clock 0: Display Control Unit (DCU)
- Auxiliary Clock 1: eMIOS_1
- Auxiliary Clock 2: eMIOS_0
- Auxiliary Clock 3: QuadSPI (uses undivided system clock when in DSPI mode)

8.3 Clock gating

The MPC5606S provides the user with the possibility of gating the clock to certain peripherals. See Section 25.4.6, Peripheral clock gating, for details.

Peripherals sets 1, 2, and 3, and the DCU, eMIOS, and QuadSPI peripherals can be configured to use specific clocks. In the case of peripheral sets 1, 2, and 3, the choice of clock is limited to the system clock optionally divided by up to 16. See Section 8.4.3.1.4, System Clock Divider Configuration Registers (CGM_SC_DC0…2). In the case of the DCU, eMIOS_0, eMIOS_1, and QuadSPI peripherals there is a choice of source clocks. For the eMIOS0 and eMIOS1 peripherals there is the option to further divide the chosen clock. See Section 8.4.4.2, Auxiliary clock generation.

Table 8-1 shows the peripheral sets, their peripherals, and the associated registers to enable and generate clocks to these peripherals. Peripherals not explicitly listed in a peripheral set or using an auxiliary clock use the system clock (or where available an alternative chosen within the peripheral) as their reference.

Table 8-1. Peripheral clock generation registers

Peripheral set	Peripherals	Registers to enable and generate clock
1	LINFlex I^2C SMC SSD SGL LCD	CGM_SC_DC0
2	FlexCAN CAN Sampler DSPI	CGM_SC_DC1
3	ADC	CGM_SC_DC2
—	DCU	CGM_AC0_SC
—	eMIOS_0	CGM_AC1_SC CGM_AC1_DC0
—	eMIOS_1	CGM_AC2_SC CGM_AC2_DC0
—	QuadSPI	CGM_AC3_SC

8.4 Clock Generation Module (MC_CGM)

8.4.1 Introduction

8.4.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all device blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). Peripheral clock selection is controlled by MC_CGM control registers. A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources that have addressable memory spaces are accessed through the MC_CGM memory space. The MC_CGM also selects and generates an output clock.

Figure 8-2 depicts the MC_CGM block diagram.

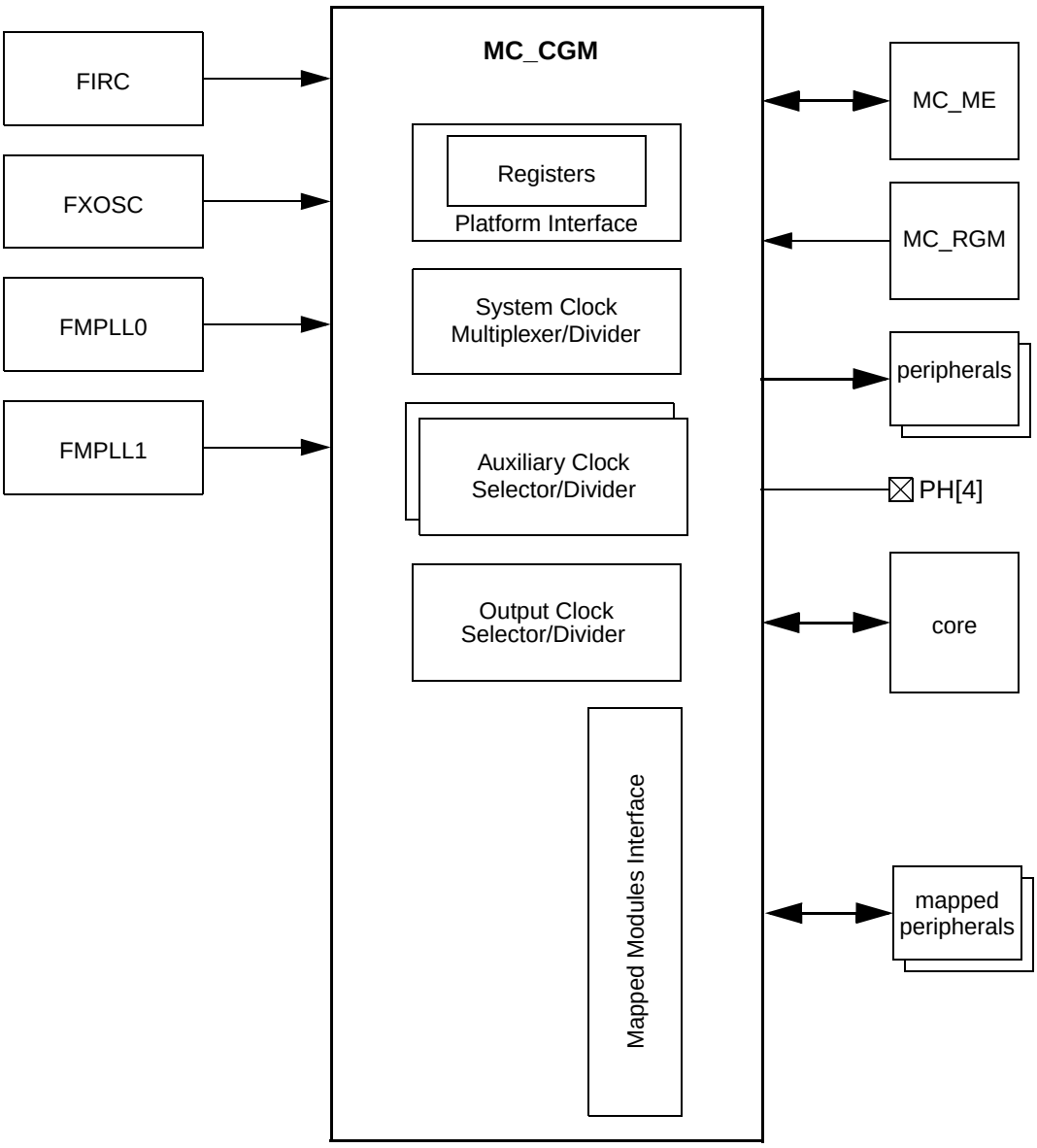


Figure 8-2. MC_CGM block diagram

8.4.1.2 Features

The MC_CGM includes the following features:

- Generates system and peripheral clocks
- Selects and enables/disables the system clock supply from system clock sources according to MC_ME control
- Contains a set of registers to control clock dividers for divided clock generation
- Contains a set of registers to control peripheral clock selection
- Supports multiple clock sources and maps their address spaces to its memory map
- Generates an output clock

- Guarantees glitchless clock transitions when changing the system clock selection
- Supports 8-, 16-, and 32-bit wide read/write accesses

8.4.1.3 Modes of operation

This section describes the basic functional modes of the MC_CGM.

8.4.1.3.1 Normal and reset modes of operation

During normal and reset modes of operation, the clock selection for the system clock is controlled by the MC_ME.

8.4.2 External signal description

The MC_CGM delivers an output clock to the PH[4] pin for off-chip use and/or observation.

8.4.3 Memory map and register definition

Table 8-2. MC_CGM register description

Address	Name	Description	Size	Access
0xC3FE_0370	CGM_OC_EN	Output Clock Enable	word	read/write
0xC3FE_0374	CGM_OCDS_SC	Output Clock Division Select	byte	read/write
0xC3FE_0378	CGM_SC_SS	System Clock Select Status	byte	read
0xC3FE_037C	CGM_SC_DC0	System Clock Divider Configuration 0	byte	read/write
0xC3FE_037D	CGM_SC_DC1	System Clock Divider Configuration 1	byte	read/write
0xC3FE_037E	CGM_SC_DC2	System Clock Divider Configuration 2	byte	read/write
0xC3FE_0380	CGM_AC0_SC	Aux Clock 0 Select Control	word	read/write
0xC3FE_0388	CGM_AC1_SC	Aux Clock 1 Select Control	word	read/write
0xC3FE_038C	CGM_AC1_DC0	Aux Clock 1 Divider Configuration 0	byte	read/write
0xC3FE_0398	CGM_AC2_SC	Aux Clock 2 Select Control	word	read/write
0xC3FE_0394	CGM_AC2_DC0	Aux Clock 2 Divider Configuration 0	byte	read/write
0xC3FE_0398	CGM_AC3_SC	Aux Clock 3 Select Control	word	read/write

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- Not change register content
- Cause a transfer error

Table 8-3. MC_CGM Memory map

Address	Name	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FE_0000 … 0xC3FE_001C	FXOSC registers																
0xC3FE_0020 … 0xC3FE_003C	Reserved																
0xC3FE_0040 … 0xC3FE_005C	SXOSC registers																
0xC3FE_0060 … 0xC3FE_007C	FIRC registers																
0xC3FE_0080 … 0xC3FE_009C	Reserved																
0xC3FE_00A0 … 0xC3FE_00BC	FMPLL0 registers																
0xC3FE_00C0 … 0xC3FE_00DC	FMPLL1 registers																
0xC3FE_00E0 … 0xC3FE_00FC	Reserved																
0xC3FE_0100 … 0xC3FE_011C	CMU0 registers																
0xC3FE_0120 … 0xC3FE_013C	Reserved																
0xC3FE_0140 … 0xC3FE_015C	Reserved																
0xC3FE_0160 … 0xC3FE_017C	Reserved																
0xC3FE_0180 … 0xC3FE_019C	Reserved																
0xC3FE_01A0 … 0xC3FE_01BC	Reserved																

Table 8-3. MC_CGM Memory map (continued)

Address	Name	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FE_01C0 … 0xC3FE_01DC	Reserved																
0xC3FE_01E0 … 0xC3FE_01FC	Reserved																
0xC3FE_0200 … 0xC3FE_021C	Reserved																
0xC3FE_0220 … 0xC3FE_023C	Reserved																
0xC3FE_0240 … 0xC3FE_025C	Reserved																
0xC3FE_0260 … 0xC3FD_C27C	Reserved																
0xC3FE_0280 … 0xC3FE_029C	Reserved																
0xC3FE_02A0 … 0xC3FE_02BC	Reserved																
0xC3FE_02C0 … 0xC3FE_02DC	Reserved																
0xC3FE_02E0 … 0xC3FE_02FC	Reserved																
0xC3FE_0300 … 0xC3FE_031C	Reserved																
0xC3FE_0320 … 0xC3FE_033C	Reserved																
0xC3FE_0340 … 0xC3FE_035C	Reserved																
0xC3FE_0360 … 0xC3FE_036C	Reserved																

Table 8-3. MC_CGM Memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FE_0370	CGM_OC_EN	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																
0xC3FE_0374	CGM_OCDS_SC	R	0	0	SELDIV			SELCTL			0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0xC3FE_0378	CGM_SC_SS	R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0xC3FE_037C	CGM_SC_DC0…2	R	DE0	0	0	0	DIV0				DE1	0	0	0	DIV1			
		W																
		R	DE2	0	0	0	DIV2				0	0	0	0	0	0	0	0
		W																
0xC3FE_0380	CGM_AC0_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0xC3FE_0384	Reserved																	
0xC3FE_0388	CGM_AC1_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0xC3FE_038C	CGM_AC1_DC0	R	DE0	0	0	0	DIV0				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																

Table 8-3. MC_CGM Memory map (continued)

Address	Name		0 16	1 17	2 18	3 19	27 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
0xC3FE_0390	CGM_AC2_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0xC3FE_0394	CGM_AC2_DC0	R	DE0	0	0	0	DIV0				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0xC3FE_0398	CGM_AC3_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0xC3FE_039C	Reserved																	
0xC3FE_0400 … 0xC3FE_3FFC	Reserved																	

8.4.3.1 Register descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered as big-endian. For example, the CGM_OC_EN register may be accessed as a word at address 0xC3FE_0370, as a half-word at address 0xC3FE_0372, or as a byte at address 0xC3FE_0373.

8.4.3.1.1 Output Clock Enable Register (CGM_OC_EN)

Address 0xC3FE_0370 Access: Supervisor read/write, User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-3. Output Clock Enable Register (CGM_OC_EN)

This register is used to enable and disable the output clock.

8.4.3.1.2 Output Clock Division Select Register (CGM_OCDS_SC)

Table 8-4. Output Clock Enable Register (CGM_OC_EN) field descriptions

Field	Description
EN	Output Clock Enable control 0 Output clock is disabled. 1 Output clock is enabled.

Address 0xC3FE_0374 Access: Supervisor read/write, User read-only

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

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

Figure 8-4. Output Clock Division Select Register (CGM_OCDS_SC)

This register is used to select the current output clock source and by which factor it is divided before being delivered at the output clock.

Table 8-5. Output Clock Division Select Register (CGM_OCDS_SC) field descriptions

Field	Description
SELDIV	Output Clock Division Select 00 Output selected: output clock without division. 01 Output selected: output clock divided by 2. 10 Output selected: output clock divided by 4. 11 Output selected: output clock divided by 8.
SELCTL	**Output Clock Source Selection Control** — This value selects the current source for the output clock. 000 16 MHz internal RC oscillator 001 4–16 MHz external oscillator 010 Primary FMPLL 011 Secondary FMPLL 100 128 kHz internal RC oscillator 101 32 kHz external oscillator 110 reserved 111 reserved

8.4.3.1.3 System Clock Select Status Register (CGM_SC_SS)

Address 0xC3FE_0378 Access: Supervisor read-only, User read-only

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

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

Figure 8-5. System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Table 8-6. System Clock Select Status Register (CGM_SC_SS) field descriptions

Field	Description
SELSTAT	**System Clock Source Selection Status** — This value indicates the clock source for the system clock. 0000 16 MHz internal RC oscillator 0001 Divided 16 MHz internal RC oscillator 0010 4–16 MHz external oscillator 0011 Divided 4–16 MHz external oscillator 0100 Primary FMPLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 System clock is disabled

8.4.3.1.4 System Clock Divider Configuration Registers (CGM_SC_DC0…2)

Address 0xC3FE_037C Access: Supervisor read/write, User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE0	0	0	0	DIV0				DE1	0	0	0	DIV1			
W																
Reset	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DE2	0	0	0	DIV2				0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-6. System Clock Divider Configuration Registers (CGM_SC_DC0…2)

These registers control the system clock dividers. The divided clock is the reference for the associated peripheral set.

Table 8-7. System Clock Divider Configuration Registers (CGM_SC_DC0…2) field descriptions

Field	Description
DE0	Divider 0 Enable 0 Disable system clock divider 0. 1 Enable system clock divider 0.
DIV0	Divider 0 Division Value — The resultant peripheral set 1 clock will have a period DIV0 + 1 times that of the system clock. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV0 field is ignored and the peripheral set 1 clock remains disabled.
DE1	Divider 1 Enable 0 Disable system clock divider 1. 1 Enable system clock divider 1.
DIV1	Divider 1 Division Value — The resultant peripheral set 2 clock will have a period DIV1 + 1 times that of the system clock. If the DE1 is set to 0 (Divider 1 is disabled), any write access to the DIV1 field is ignored and the peripheral set 2 clock remains disabled.
DE2	Divider 2 Enable 0 Disable system clock divider 2. 1 Enable system clock divider 2.
DIV2	Divider 2 Division Value — The resultant peripheral set 3 clock will have a period DIV2 + 1 times that of the system clock. If the DE2 is set to 0 (Divider 2 is disabled), any write access to the DIV2 field is ignored and the peripheral set 3 clock remains disabled.

8.4.3.1.5 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Address 0xC3FE_0380 Access: Supervisor read/write, User read-only

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

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

Figure 8-7. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

This register is used to select the current auxiliary clock 0 sources.

Table 8-8. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC) field descriptions

Field	Description
SELCTL	**Auxiliary Clock 0 Source Selection Control** — This value selects the current source for auxiliary clock 0. 0000 4–16 MHz external oscillator 0001 16 MHz internal RC oscillator 0010 Secondary FMPLL 0011 Primary FMPLL 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

8.4.3.1.6 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Address 0xC3FE_0388 Access: Supervisor read/write, User read-only

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

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

Figure 8-8. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

This register is used to select the current auxiliary clock 1 sources.

Table 8-9. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC) field descriptions

Field	Description
SELCTL	**Auxiliary Clock 1 Source Selection Control** — This value selects the current source for auxiliary clock 1. 0000 div. 4-16 MHz external oscillator 0001 div. 16 MHz int. RC osc. 0010 Secondary FMPLL 0011 Primary FMPLL 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

8.4.3.1.7 Auxiliary Clock 1 Divider Configuration Register (CGM_AC1_DC0)

Address 0xC3FE_038C Access: Supervisor read/write, User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE0	0	0	0	DIV0				0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 8-9. Auxiliary Clock 1 Divider Configuration Register (CGM_AC1_DC0)

This register controls the auxiliary clock 1 divider.

Table 8-10. Auxiliary Clock 1 Divider Configuration Register (CGM_AC1_DC0) field descriptions

Field	Description
DE0	Divider 0 Enable 0 Disable auxiliary clock 1 divider 0. 1 Enable auxiliary clock 1 divider 0.
DIV0	Divider 0 Division Value — The resultant eMIOS0 clock will have a period DIV0 + 1 times that of auxiliary clock 1. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV 0 field is ignored and the eMIOS0 clock remains disabled.

8.4.3.1.8 Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)

Address 0xC3FE_0398 Access: Supervisor read/write, User read-only

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

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

Figure 8-10. Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)

This register is used to select the current auxiliary clock 1 sources.

Table 8-11. Auxiliary Clock 2 Select Control Register (CGM_AC2_SC) field descriptions

Field	Description
SELCTL	**Auxiliary Clock 2 Source Selection Control** — This value selects the current source for auxiliary clock 2. 0000 div. 4–16 MHz external oscillator 0001 div. 16 MHz int. RC osc. 0010 Secondary FMPLL 0011 Primary FMPLL 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

8.4.3.1.9 Auxiliary Clock 2 Divider Configuration Register (CGM_AC2_DC0)

Address 0xC3FE_0394 Access: Supervisor read/write, User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE0	0	0	0	DIV0				0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 8-11. Auxiliary Clock 2 Divider Configuration Register (CGM_AC2_DC0)

This register controls the auxiliary clock 2 divider.

Table 8-12. Auxiliary Clock 2 Divider Configuration Register (CGM_AC2_DC0) field descriptions

Field	Description
DE0	Divider 0 Enable 0 Disable auxiliary clock 2 divider 0. 1 Enable auxiliary clock 2 divider 0.
DIV0	Divider 0 Division Value — The resultant eMIOS1 clock will have a period DIV0 + 1 times that of auxiliary clock 2. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV 0 field is ignored and the eMIOS1 clock remains disabled.

8.4.3.1.10 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Address 0xC3FE_0398 Access: Supervisor read/write, User read-only

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

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

Figure 8-12. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current auxiliary clock 3 sources.

Table 8-13. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC) field descriptions

Field	Description
SELCTL	**Auxiliary Clock 3 Source Selection Control** — This value selects the current source for auxiliary clock 3. 0000 System clock 0001 System clock / 2 0010 Secondary FMPLL 0011 Secondary FMPLL / 2 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

8.4.4 Functional description

8.4.4.1 System clock generation

Figure 8-13 shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see Chapter 25, Mode Entry Module (MC_ME), for more details), and the MC_RGM provides the safe clock request (see Chapter 31, Reset Generation Module (MC_RGM), for more details). The safe clock request forces the selector to select the 16 MHz internal RC oscillator as the system clock and to ignore the system clock select.

8.4.4.1.1 System clock source selection

During normal operation, the system clock selection is controlled by:

- MC_RGM on a Safe mode event
- MC_ME in all other cases

8.4.4.1.2 System clock disable

During normal operation, the system clock can be disabled by the MC_ME.

8.4.4.1.3 System clock dividers

The MC_CGM generates three derived clocks from the system clock that are used as the reference clocks for their associated peripherals.

8.4.4.2 Auxiliary clock generation

Figure 8-14 (and those following) shows the block diagram of the auxiliary clock generation logic. See Section 8.4.3.1.5, Auxiliary Clock 0 Select Control Register (CGM_AC0_SC), Section 8.4.3.1.6, Auxiliary Clock 1 Select Control Register (CGM_AC1_SC), Section 8.4.3.1.8, Auxiliary Clock 2 Select Control Register (CGM_AC2_SC), and Section 8.4.3.1.10, Auxiliary Clock 3 Select Control Register (CGM_AC3_SC), for auxiliary clock selection control.

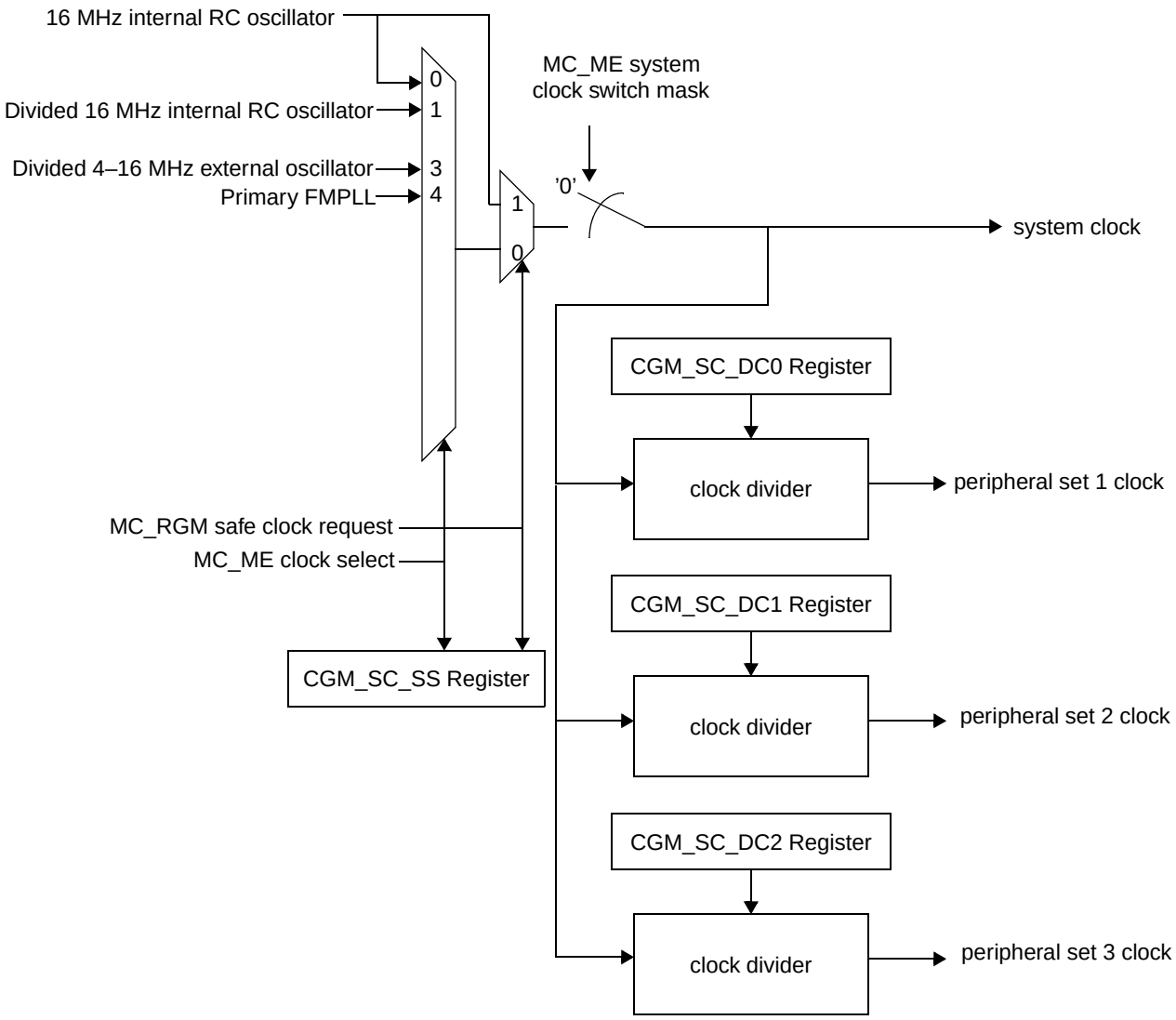


Figure 8-13. MC_CGM System Clock Generation Overview

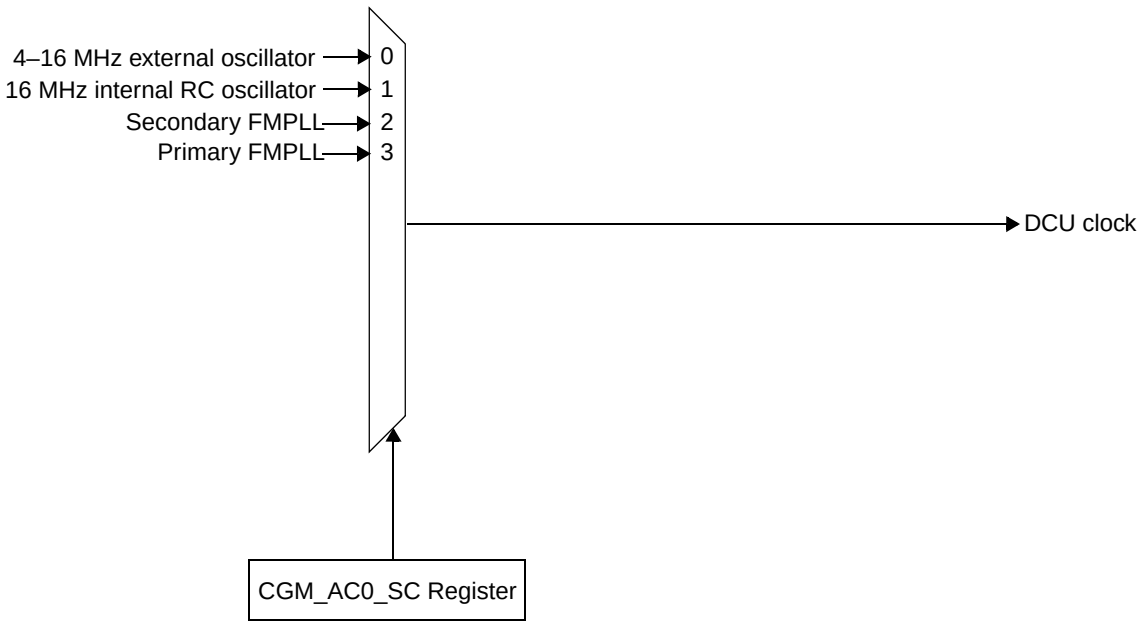


Figure 8-14. MC_CGM Auxiliary Clock 0 Generation Overview

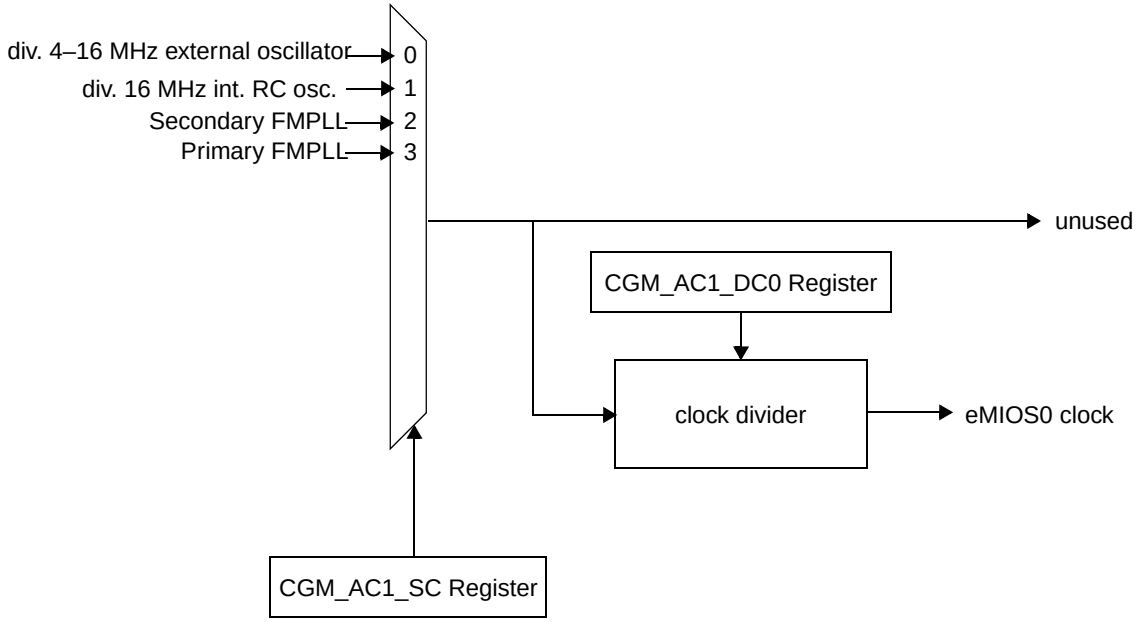


Figure 8-15. MC_CGM Auxiliary Clock 1 Generation Overview

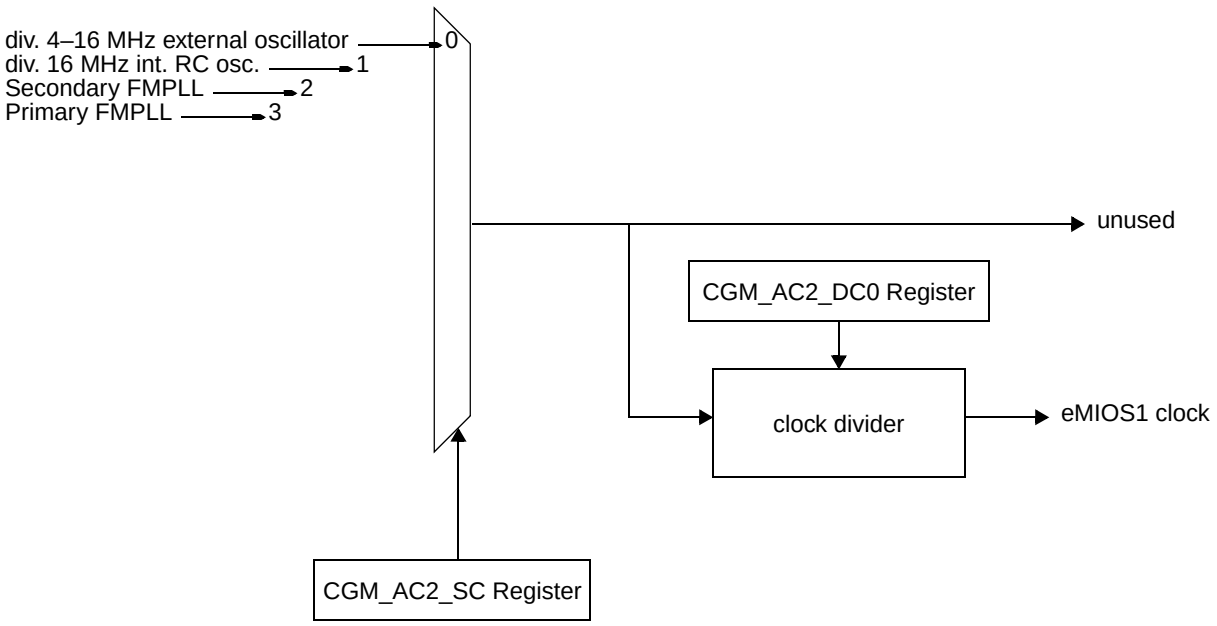


Figure 8-16. MC_CGM Auxiliary Clock 2 Generation Overview

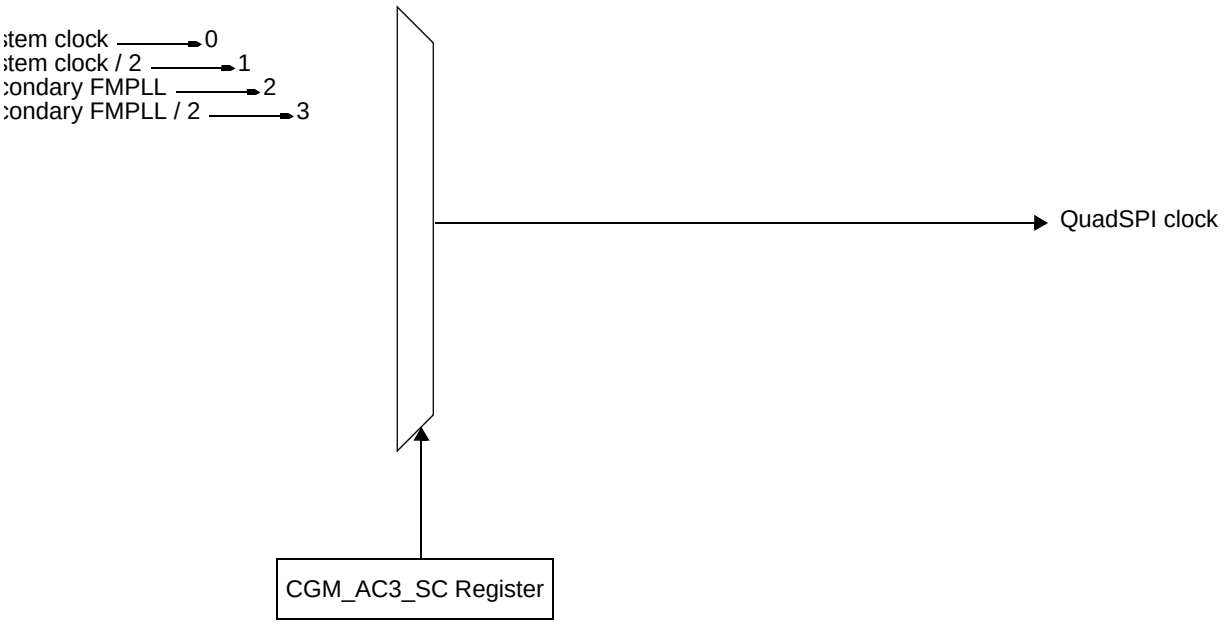


Figure 8-17. MC_CGM Auxiliary Clock 3 Generation Overview

8.4.4.2.1 Auxiliary Clock Source Selection

During normal operation, the auxiliary clock selection is done via the CGM_AC0…3_SC registers. If software selects an unavailable source, the old selection remains, and the register content does not change.

8.4.4.2.2 Auxiliary clock dividers

The selected auxiliary clock can be optionally divided before use.

8.4.4.2.3 Dividers functional description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- Section 8.4.3.1.4, System Clock Divider Configuration Registers (CGM_SC_DC0…2)
- Section 8.4.3.1.7, Auxiliary Clock 1 Divider Configuration Register (CGM_AC1_DC0)
- Section 8.4.3.1.9, Auxiliary Clock 2 Divider Configuration Register (CGM_AC2_DC0)

The reset value of all counters is 1. If a divider has its DE bit in the respective configuration register set to 0 (the divider is disabled), any value in its DIV*n* field is ignored.

8.4.4.3 Output clock multiplexing

The MC_CGM contains a multiplexing function for a number of clock sources which can then be used as output clock sources. The selection is done via the CGM_OCDS_SC register.

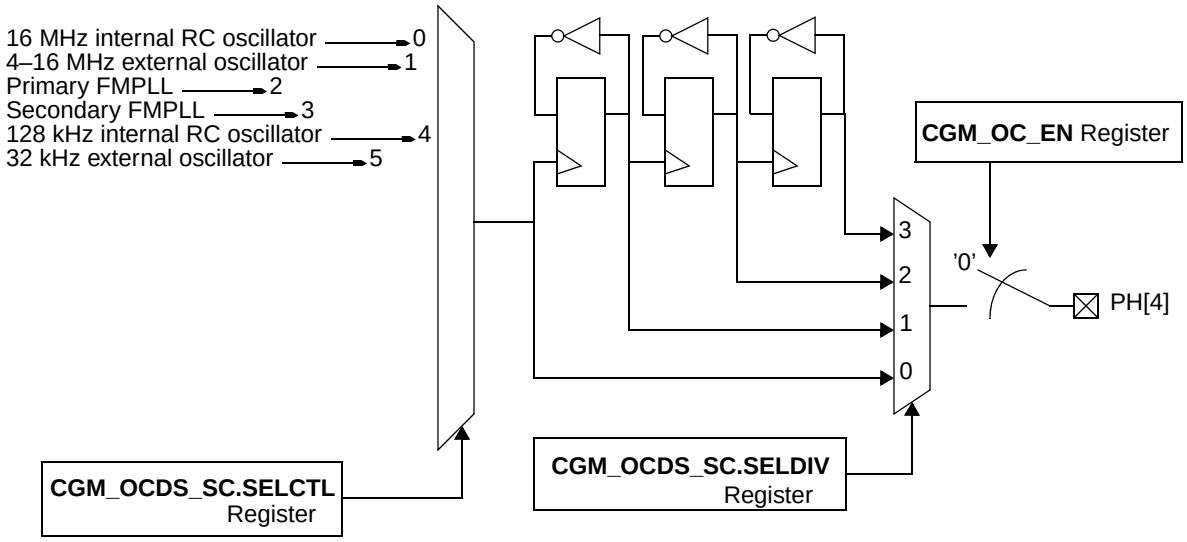


Figure 8-18. MC_CGM Output Clock Multiplexer and PH[4] Generation

8.4.4.4 Output Clock Division Selection

The MC_CGM provides the following output signals for the output clock generation:

- PH[4] (see Figure 8-18). This signal is generated by using one of the 3-stage ripple counter outputs or the selected signal without division. The non-divided signal is not guaranteed to be 50% duty cycle by the MC_CGM.
- The MC_CGM also has an output clock enable register (see Section 8.4.3.1.1, Output Clock Enable Register (CGM_OC_EN)), which contains the output clock enable/disable control bit.

8.5 FXOSC external oscillator

The FXOSC digital interface controls the external crystal oscillator (FXOSC). It holds control and status registers that are accessible by software.

8.5.1 Main features

- External crystal oscillator (FXOSC) digital interface
- Oscillator clock available interrupt
- Oscillator bypass mode
- Output clock division factors ranging from 1,2,3....32

8.5.2 Functional description

The crystal oscillator circuit includes an internal oscillator driver and external crystal circuitry. It provides an output clock that can be provided to PLL or used as a reference clock to specific modules depending on system needs.

The crystal oscillator is controlled by the MC_ME module. The OSCON bit of ME_<*xxx*>_MCR registers controls the power-down of the oscillator based on the current device mode, while S_OSC of ME_GS register provides the oscillator clock available status.

After system reset, the oscillator is powered down, and software has to restart it when required. Whenever the crystal oscillator is switched on from the off state, the OSCCNT counter starts. When it reaches the value of EOCV[7:0] × 512, an oscillator clock is made available to the system. Also, an interrupt pending bit, I_OSC of the OSC_CTL register, is set. An interrupt is generated if the interrupt mask bit M_OSC is set.

The oscillator circuit can be bypassed by setting OSC_CTL[OSCBYP]. This bit can only be set by the software. System reset is needed to reset this bit. In this bypass mode, the output clock has the same polarity as the external clock applied on the EXTAL pin, and the oscillator status is forced to 1. The bypass configuration is independent of the power-down mode of the oscillator.

Table 8-14 shows the truth table for different configurations of the oscillator.

Table 8-14. Truth table of crystal oscillator

ENABLE	BYP	XTAL	EXTAL	CK_OSCM	OSC MODE
0	0	No crystal, high impedance	No crystal, high impedance	0	Power down, IDDQ
x	1	x	Ext clock	EXTAL	Bypass, OSC disabled
1	0	Crystal	Crystal	EXTAL	Normal, OSC enabled
		Gnd	Ext clock	EXTAL	Normal, OSC enabled

The crystal oscillator clock can be further divided by a configurable factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the OSCDIV[4:0] bits of the OSC_CTL register.

8.5.3 Register description

Base Address: 0xC3FE_0000 Offset: 0x0000 Access: Supervisor: read/write, User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC	0	0	OSCDIV					I_OSC	0	0	0	0	0	0	0
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-19. Crystal Oscillator Control Register (OSC_CTL)

Table 8-15. Crystal Oscillator Control Register (OSC_CTL) field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Only software can set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.
8–15 EOCV[7:0]	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset, or whenever it is switched on from the off state. This counting period ensures that the external oscillator clock signal is stable before it can be selected by the system. When the oscillator counter reaches the value EOCV[7:0] × 512, an oscillator available interrupt request is generated. The reset value of this field depends on the device specification. The OSCCNT counter will be kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
19–23 OSCDIV[4:0]	Crystal oscillator clock division factor These bits specify the crystal oscillator output clock division factor. The output clock is divided by the factor OSCDIV + 1.
24 I_OSC	Crystal oscillator clock interrupt This bit is set by hardware when OSCCNT counter reaches the count value EOCV[7:0] × 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.

Note: OSC_CTL register is writable only in supervisor mode.

8.6 32 KHz OSC digital interface

8.6.1 Introduction

The OSC digital interface controls the external crystal oscillator (SXOSC). It holds control and status registers that are accessible by software.

8.6.2 Main features

- External crystal oscillator (SXOSC) digital interface
- Oscillator power-down control and status
- Oscillator clock available interrupt
- Oscillator bypass mode
- Output clock division factors ranging from 1,2,3....32

8.6.3 Functional description

The crystal oscillator circuit includes an internal oscillator driver and external crystal circuitry. It can be used as a reference clock to specific modules depending on system needs.

The crystal oscillator is controlled by the OSC_CTL register. The OSCON bit controls the power-down while the S_OSC bit provides the oscillator clock available status.

After system reset, the oscillator is put into power-down state and software has to start it up when required. Whenever the crystal oscillator is switched on from an off state, the OSCCNT counter starts. When it reaches the value of EOCV[7:0] × 512, the oscillator clock is made available to the system. Also, an interrupt pending bit, I_OSC of the OSC_CTL register, is set. An interrupt is generated if the interrupt mask bit M_OSC is set.

The oscillator circuit can be bypassed by setting the OSCBYP bit in the OSC_CTL register to 1. This bit can only be set by the software. System reset is needed to reset this bit. In this bypass mode, the output clock has the same polarity as external clock applied on EXTAL32 pin and the oscillator status is forced to 1. The bypass configuration is independent of the power-down mode of the oscillator.

The table below shows the truth table of different configurations of the oscillator.

Table 8-16. Truth table of crystal oscillator

OSCON	OSCBYP	XTAL32	EXTAL32	CK_OSCM	OSC MODE
0	0	No crystal, high Z	No crystal, high Z	0	Power down, IDDQ
x	1	Ext clock	x	EXTAL32	Bypass, OSC disabled
1	0	Crystal	Crystal	EXTAL32	Normal, OSC enabled
		Gnd	Ext clock	EXTAL32	Normal, OSC enabled

The crystal oscillator clock can be further divided by a configurable factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the OSCDIV[4:0] bits of the OSC_CTL register.

8.6.4 Register description

Base Address: 0xC3FE_0040 Offset: 0x0000 Access: Supervisor: read/write, User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_ OSC	0	0	OSCDIV					I_ OSC	0	0	0	0	0	S_O SC	OSC ON
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-20. Crystal Oscillator Control Register (OSC_CTL)

Note: OSC32, after it is enabled, is always on, but can be turned off in Standby mode by writing the OSCON bit.

Table 8-17. Crystal Oscillator Control Register (OSC_CTL) field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Only software can set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL32 is used as root clock.
8–15 EOCV[7:0]:\	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset, or whenever it is switched on from the off state. This counting period ensures that the external oscillator clock signal is stable before it can be selected by the system. When the oscillator counter reaches the value of EOCV[7:0] × 512, the oscillator available interrupt request is generated. The reset value of this field depends on the device specification. The OSCCNT counter will be kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
19–23 OSCDIV[4:0]	Crystal oscillator clock division factor These bits specify the crystal oscillator output clock division factor. The output clock is divided by the factor OSCDIV + 1.
24 I_OSC	Crystal oscillator clock interrupt This bit is set by hardware when the OSCCNT counter reaches the count value of EOCV[7:0] × 512. It is cleared by writing a 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.

Table 8-17. Crystal Oscillator Control Register (OSC_CTL) field descriptions (continued)

Field	Description
30 S_OSC	Crystal oscillator status 0 Crystal oscillator output clock is not stable. 1 Crystal oscillator is providing a stable clock.
31 OSCON	Crystal oscillator powerdown control 0 Crystal oscillator is switched off. 1 Crystal oscillator is switched on.

Note: OSC_CTL register is writable only in supervisor mode.

8.7 SIRC digital interface

8.7.1 Introduction

The SIRC digital interface controls the internal low-power 128 kHz RC oscillator (SIRC). It holds control and status registers that are accessible by software.

8.7.2 Low-Power RC Oscillator (128 kHz)

The low-power RC oscillator provides a low frequency (f_{LPRC}) clock in the range of tens of kHz, requiring less current consumption. This clock can be used as a reference clock when a fixed base time is required for specific modules.

The low-power RC oscillator is always on in all device modes.

The SIRC clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the LPRCDIV[4:0] bits of the LPRC_CTL register.

The SIRC oscillator output frequency can be trimmed by the LPRCTRIM[4:0] bits of the LPRC_CTL register. These bits can be programmed to modify an internal capacitor/resistor. After power-on reset, the trimming bits are provided by the flash options. After the first write access, only the value specified by the LPRCTRIM[4:0] bits will control the trimming.

8.7.3 Register description

Base Address: 0xC3FE_0080 Offset: 0x0000 Access: Supervisor: read/write, User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	LPRCDIV					0	0	0	S_LPRC	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-21. Low-Power RC Control Register (LPRC_CTL)

Table 8-18. Low-Power RC Control Register (LPRC_CTL) field descriptions

Field	Description
11–15 LPRCTRIM[4:0]	Low-power RC trimming bits **Note:** Not all configurations can be used. Please refer to the MPC5606S .
19–23 LPRCDIV[4:0]	Low-power RC clock division factor These bits specify the low-power RC oscillator output clock division factor. The output clock is divided by the factor LPRCDIV + 1.
27 S_LPRC	Low-power RC clock status 0 LPRC is not providing a stable clock. 1 LPRC is providing a stable clock.

Note: The LPRC_CTL register is writable only in supervisor mode.

8.8 FIRC digital interface

8.8.1 Introduction

The FIRC digital interface controls the main internal 16 MHz RC oscillator (FIRC). It holds control and status registers that are accessible by software.

8.8.2 Functional description (16 MHz)

The main RC oscillator provides a high-frequency (f_{MRC}) clock. This clock can be used to fasten the exit from reset and wakeup sequence from low-power modes of the system. It is controlled by the MC_ME module based on the current device mode. The clock source status is updated in the S_RC bit of the ME_GS register. Please refer to Chapter 25, Mode Entry Module (MC_ME), for further details.

The MRC clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the RCDIV[4:0] bits of the RC_CTL register.

The main RC oscillator output frequency can be trimmed by the RCTRIM[5:0] bits of the RC_CTL register. These bits can be programmed to modify internal capacitor/resistor values. After power-on reset, the trimming bits are provided by the flash options. After the first write access, only the value specified by the RCTRIM[5:0] bits will control the trimming.

During standby mode entry process, the RC oscillator is controlled based on the RCON bit of the ME_STANDBY_MC register. The is the last step in the standby entry sequence. On any system wakeup event, the device exits Standby mode and switches on the RC oscillator.

8.8.3 Register description

Base Address: 0xC3FE_0060 Offset: 0x0000 Access: Supervisor: read/write, User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	RCDIV												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-22. RC Oscillator Control Register (RC_CTL)

Table 8-19. RC Oscillator Control Register (RC_CTL) field descriptions

Field	Description
10–15 RCTRIM[5:0]	Low-power RC trimming bits **Note:** Not all configurations can be used. Please refer to the MPC5606S .
19–23 RCDIV[4:0]	Low-power RC clock division factor These bits specify the low-power RC oscillator output clock division factor. The output clock is divided by the factor LPRCDIV + 1.

Note: The RC_CTL register is writable only in supervisor mode.

8.9 Frequency-modulated phase locked loops and system clocks (FMPLL0 and FMPLL1)

8.9.1 Introduction

This section describes the features and functions of the two independent FMPLL modules implemented in MPC5606S.

8.9.2 Overview

The FMPLLs enable the user to generate high-speed system clocks from a common 4 MHz to 16 MHz input clock. Further, the FMPLLs support programmable frequency modulation of the system clock. The PLL multiplication factor and the output clock divider ratio are both software-configurable.

NOTE

The user must take care not to program the device with a frequency higher than allowed (no hardware check is provided).

The FMPLL block diagram is shown in Figure 8-23.

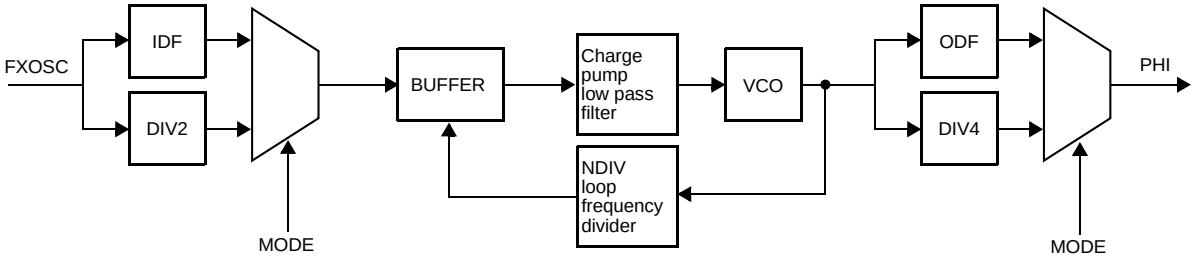


Figure 8-23. FMPLL block diagram

8.9.3 Features

Each FMPLL has the following major features:

- Input clock frequency from 4 MHz to 16 MHz
- Voltage controlled oscillator (VCO) range from 256 MHz to 512 MHz
- Reduced frequency divider (RFD) for reduced frequency operation without forcing the PLL to relock
- Frequency-modulated PLL
  - Modulation enabled/disabled through software
  - Triangle wave modulation
- Programmable modulation depth
  - ±0.25% to ±4% deviation from center-spread frequency
  - –0.5% to +8% deviation from down-spread frequency
  - Programmable modulation frequency dependent on reference frequency
- Self-clocked mode (SCM) operation
- Five available modes
  - Normal mode
  - Progressive clock switching
  - Normal mode with FM
  - Power-down mode
  - 1:1 mode (FMPLL0 only)

8.9.4 Memory map

Table 8-20 shows the memory map locations. Addresses are given as offsets of the module base address.

NOTE

FMPLL_*x* registers are mapped through the MC_CGM register slot.

Table 8-20. FMPLL Memory map

Address	Register	Access	Location
Base: 0xC3FE00A0 (FMPLL0) 0xC3FE00C0 (FMPLL1)			
0x0000	Control Register (CR)	R/W	on page 217
0x0004	Modulation Register (MR)	Special	on page 220

8.9.5 Register description

The PLL operation is controlled by two registers. Those registers can only be written in supervisor mode.

8.9.5.1 Control Register (CR)

Offset 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	en_pll_sw	mode	unlock_once	0	i_lock	s_lock	pll_fail_mask	pll_fail_flag	0
W												w1c			w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Reset value is determined by the SoC integration.

Figure 8-24. Control Register (CR)

Table 8-21. CR field descriptions

Field	Description
2–5 IDF	The value of this field sets the PLL Input division factor as described in Table 8-22. The reset value is set during integration.
6–7 ODF	The value of this field sets the PLL Output division factor as described in Table 8-23. The reset value is set during integration.
9–15 NDIV	The value of this field sets the PLL Loop division factor as described in Table 8-24. The reset value is set during integration.
23 en_pll_sw	This bit is used to enable progressive clock switching. After the PLL locks, the PLL output initially is divided by 8 then progressively divides down until divide by 1. 0 Progressive clock switching disabled 1 Progressive clock switching enabled **Note:** The PLL output should not be used if a non-changing clock is needed (such as for serial communications) until the division has finished.
24 mode	This bit activates the 1:1 mode.
25 unlock_once	This bit is a sticky indication of PLL loss-of-lock condition. The unlock_once bit is set when the PLL loses lock. Whenever the PLL reacquires lock, unlock_once remains set. Only a power-on reset can clear this bit. **Note:** If the FMPLL is locked and a functional reset occurs, FMPLL_CR[UNLOCK_ONCE] is automatically set even when the FMPLL has not lost lock.
27 i_lock	This bit is set by hardware whenever there is a lock/unlock event.It is cleared by software, writing 1.
28 s_lock	This bit indicates whether the PLL has acquired lock. 0 PLL unlocked 1 PLL locked
29 pll_fail_mask	This bit is used to mask the pll_fail output. 0 pll_fail not masked 1 pll_fail masked
30 pll_fail_flag	This bit is asynchronously set by hardware whenever a loss of lock event occurs while PLL is switched on. It is cleared by software, writing 1.

Table 8-22. Input divide ratios

IDF[3:0]	Input divide ratios
0000	Divide by 1
0001	Divide by 2
0010	Divide by 3
0011	Divide by 4
0100	Divide by 5
0101	Divide by 6
0110	Divide by 7
0111	Divide by 8

Table 8-22. Input divide ratios (continued)

IDF[3:0]	Input divide ratios
1000	Divide by 9
1001	Divide by 10
1010	Divide by 11
1011	Divide by 12
1100	Divide by 13
1101	Divide by 14
1110	Divide by 15
1111	Clock Inhibit

Table 8-23. Output divide ratios

ODF[1:0]	Output divide ratios
00	Divide by 2
01	Divide by 4
10	Divide by 8
11	Divide by 16

Table 8-24. Loop divide ratios

NDIV[6:0]	loop divide ratios
0000000–0011111	N/A
0100000	Divide by 32
0100001	Divide by 33
0100010	Divide by 34
...	...
1011111	Divide by 95
1100000	Divide by 96
1100001–1111111	N/A

8.9.5.2 Modulation Register (MR)

Offset 0x0004 Access: User read/write

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

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

Figure 8-25. Modulation Register (MR)

Table 8-25. MR field descriptions

Field	Description
0 STRB_BYPASS	Strobe bypass The STRB_BYPASS signal is used to bypass the STRB signal used inside PLL to latch the correct values for control bits (INC_STEP, MOD_PERIOD and SPRD_SEL). 0 STRB is used to latch PLL modulation control bits. 1 STRB is bypassed. In this case control bits need to be static. The control bits must be changed only when PLL is in power-down mode.
2 SPRD_SEL	Spread type selection The SPRD_SEL controls the spread type in Frequency Modulation mode. 0 Center spread 1 Down spread
3–15 MOD_PERIOD	Modulation period The MOD_PERIOD field is the binary equivalent of the value modperiod derived from following formula: $$\text{modperiod} = \frac{f_{ref}}{4 \times f_{mod}}$$ where: f_{ref}: represents the frequency of the feedback divider f_{mod}: represents the modulation frequency The maximum value of MOD_PERIOD is 0x1000.

Table 8-25. MR field descriptions (continued)

Field	Description
16 FM_EN	Frequency Modulation Enable The FM_EN enables the frequency modulation. 0 = Frequency Modulation disabled 1= Frequency Modulation enabled
17–31 INC_STEP	Increment step The INC_STEP field is the binary equivalent of the value *incstep* derived from the following formula: $$\text{incstep} = \text{round}\left(\frac{(2^{15}-1)\times md \times MDF}{100 \times 5 \times MODPERIOD}\right)$$ where: *md*: represents the peak modulation depth in percentage (Center spread — pk-pk = ±md, Downspread — pk-pk = -2 × md) *MDF*: represents the nominal value of loop divider (NDIV in PLL Control Register)

8.9.6 Functional description

8.9.6.1 Normal mode

In Normal mode, the PLL inputs are driven by the CR (see Section 8.9.5.1, Control Register (CR)). This means that when the PLL is in lock state, the PLL output clock (PHI) is derived from the reference clock (CLKIN) through Equation 8-1:

$$phi = \frac{clkin \cdot LDF}{IDF \cdot ODF}$$

Eqn. 8-1

where the value of IDF, LDF, and ODF are set in CR and can be derived from Table 8-22, Table 8-23, and Table 8-24.

8.9.6.2 Progressive clock switching

Progressive clock switching allows switching the system clock to PLL output clock by stepping through different division factors. This means that the current consumption gradually increases, so the voltage regulator has a better response.

This feature can be enabled by programming the en_pll_sw bit in CR. Then, when the input pin pll_select goes high, the output clock ck_pll_div will progressively increase its frequency as described in Table 8-26 and Figure 8-26.

Table 8-26. Progressive clock switching on pll_select rising edge

Number of PLL output clock cycles	ck_pll_frequency (PLL output clock frequency)
8	(ck_pll_out frequency)/8
16	(ck_pll_out frequency)/4
32	(ck_pll_out frequency)/2
onward	(ck_pll_out frequency)

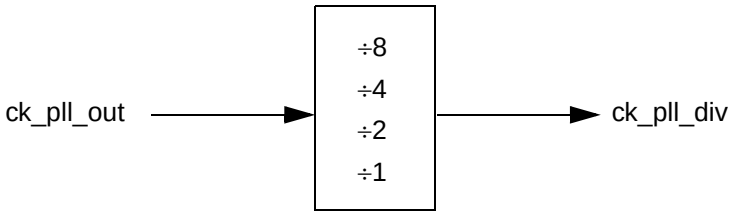


Figure 8-26. Diagram of progressive clock switching

8.9.6.3 Normal mode with frequency modulation

The FMPLL default mode is without frequency modulation enabled. When frequency modulation is enabled, however, two parameters must be set to generate the desired level of modulation: the period and the step. The modulation waveform is always a triangle wave and its shape is not programmable.

FM modulation is activated in two steps:

1. Configure the FM modulation characteristics: MOD_PERIOD, INC_STEP.
2. Enable the FM modulation by programming the FM_EN bit of MR register to 1. FM modulated mode can be enabled only when the PLL is in lock state.

To latch these values inside the PLL, two ways are used, depending on the value of the STRB_BYPASS register bit in MR.

If STRB_BYPASS is low, the modulation parameters are latched in the PLL only when the STRB signal goes high for at least two cycles of the INFIN clock. The STRB signal is automatically generated in the PLLD when the modulation is enabled (FM_EN goes high) if the PLL is locked (s_lock = 1) or when the modulation has been enabled (FM_EN = 1) and PLL enters in lock state (s_lock goes high).

If STRB_BYPASS is high, the STRB signal is bypassed. In this case, control bits (MOD_PERIOD[12:0], INC_STEP[14:0], SPREAD_CONTROL) need to be static or hardwired to constant values. The control bits must be changed only when the PLL is in power down mode.

The modulation depth in % is

$$\text{ModulationDepth} = \left(\frac{100 \times 5 \times \text{INCSTEPxMODPERIOD}}{(2^{15} - 1) \times \text{MDF}}\right)$$

NOTE

You must ensure that the value of MODPERIOD does not exceed 0x1000 and that the product of INCTEP and MODPERIOD is less than $(2^{15} - 1)$.

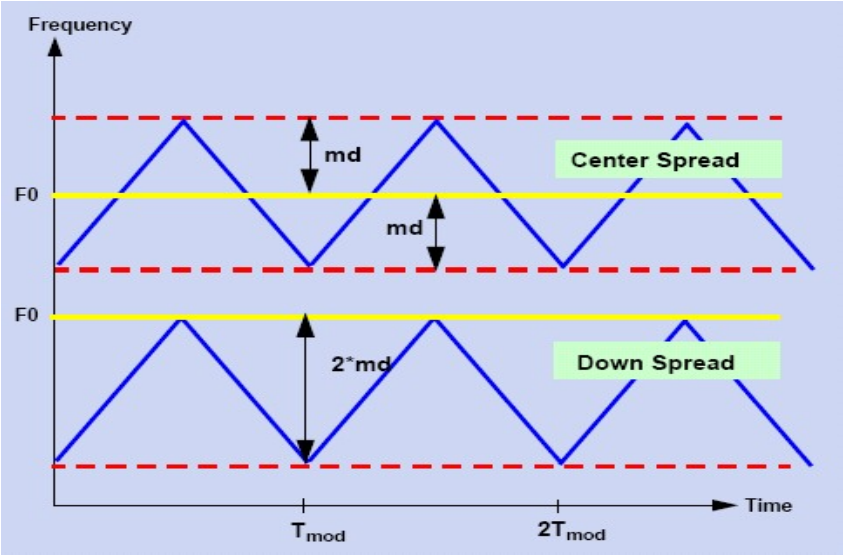


Figure 8-27. PLL frequency modulation modes

8.9.6.4 Powerdown mode

The PLL can be switched off when not required to achieve lower consumption by programming the registers ME_*x*_MC register on MC_ME module.

8.9.6.5 1:1 mode (FMPLL0 only)

1:1 mode is set by asserting the mode bit in CR (see Section 8.9.5.1, Control Register (CR)). An external input signal (mode_en) has been provided to disable this feature. If mode_en is tied to 0, the mode bit in CR is disabled and there is no way to activate 1:1 mode.

In 1:1 mode the inputs of the PLL are driven by CR and MR, but the division factors and the modulation parameters have no influence on the output clock. In fact the dividers and the FM control are bypassed inside the PLL. The PLL output clock (phi) frequency is determined by the following relation:

$$\text{phi} = \frac{\text{clkin}}{2}$$

8.9.7 Recommendations

To avoid any unpredictable behavior of the PLL clock, it is recommended to respect the following guidelines:

- The PLL VCO frequency should reside in the range 256 MHz to 512 MHz. Care is required when programming the multiplication and division factors to respect this requirement.
- The user must change the multiplication, division factors only when the PLL output clock is not selected as system clock. MOD_PERIOD, INC_STEP, SPREAD_SEL bits should be modified before activating the FM modulated mode. Then strobe has to be generated to enable the new settings. If STRB_BYP is set to 1 then MOD_PERIOD, INC_STEP and SPREAD_SEL can be modified only when PLL is in power-down mode.

- Use progressive clock switching.

8.10 Clock Monitor Unit (CMU)

8.10.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves two purposes. The main task is to permanently supervise the integrity of the device's system clock sources, that is, crystal oscillator FXOSC, FIRC, and FMPLL0. If FMPLL0 leaves an upper or lower frequency boundary, or the crystal oscillator fails, it can detect and forward this kind of event to the mode and clock management unit. The clock management unit in turn can then switch to a safe mode where it uses a safe fallback clock source such as an on-chip RC oscillator, resets the device, or generates the interrupt according to the system needs.

It can also monitor an external crystal oscillator clock, the speed of which must be greater than the internal RC clock divided by a division factor given by the RCDIV[1:0] field of the CMU_CSR register, and when enabled, generates a system clock transition request or an interrupt.

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of one clock source as compared to a reference clock. This is useful to allow the calibration of the on-chip RC oscillator(s), as well as being able to correct/calculate the time deviation of a counter which is clocked by the RC oscillator.

NOTE

The CMU does not monitor SXOSC, SIRC, or FMPLL_1.

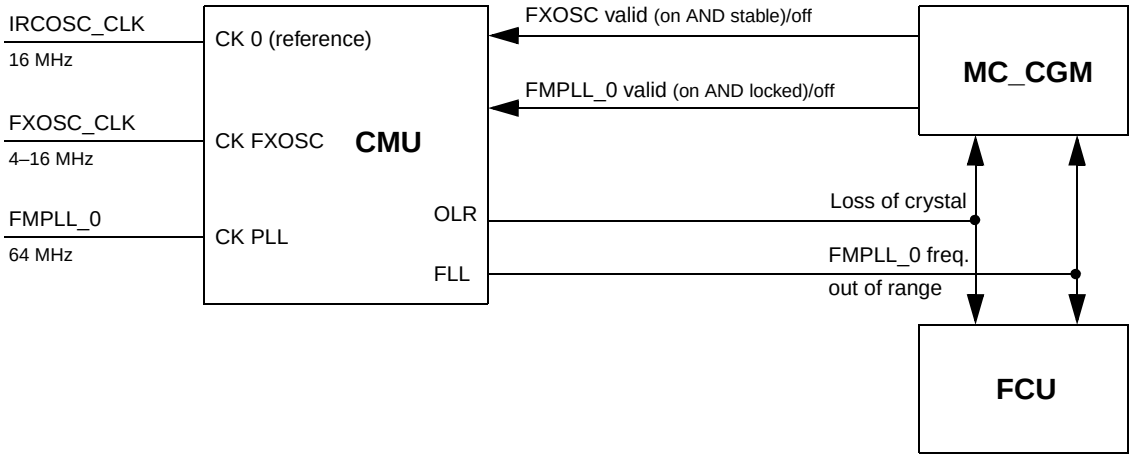


Figure 8-28. CMU component interaction

8.10.2 Main features

- RC oscillator frequency measurement
- External oscillator clock monitoring with respect to CK_FIRC/*n* clock
- PLL clock frequency monitoring with respect to CK_FIRC/4 clock

- Event generation for various failures detected inside monitoring unit

8.10.3 Block diagram

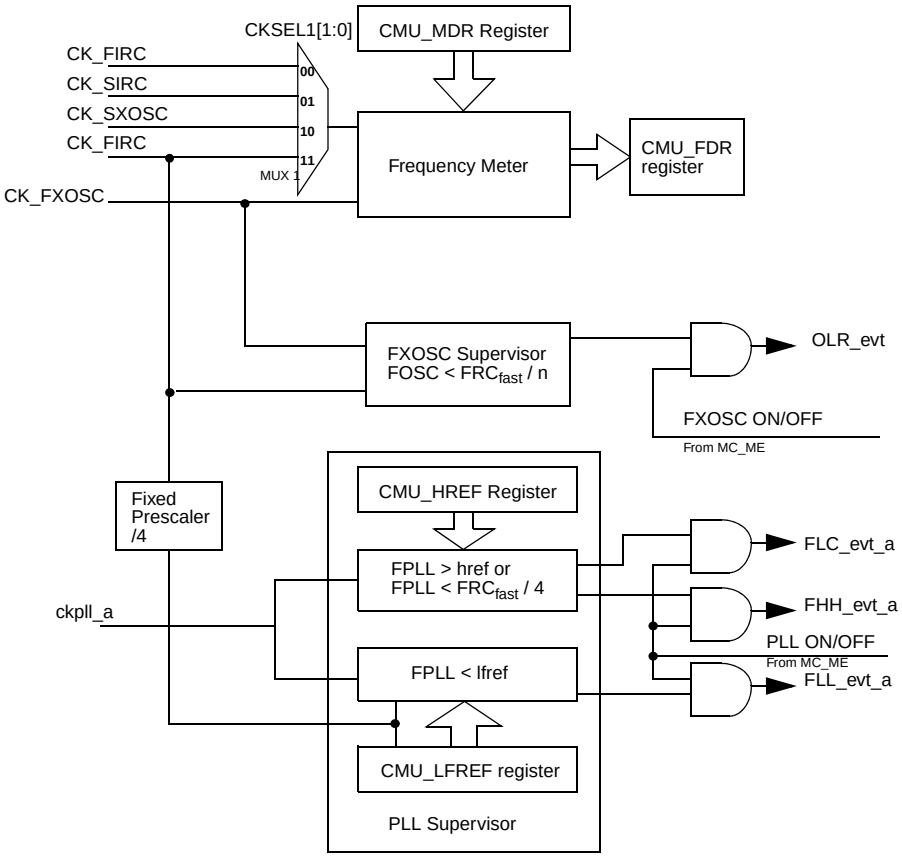


Figure 8-29. Clock Monitor Unit diagram

8.10.4 Functional description

The names of the clocks involved in this block have the following meaning:

- CK_FXOSC: clock from the external crystal oscillator
- CK_SIRC: clock from the low-frequency internal RC oscillator
- CK_FIRC: clock from the high-frequency internal RC oscillator
- CK_PLL: clock from the PLL
- FOSC: frequency of external crystal oscillator clock
- FRCslow: frequency of low-frequency internal RC oscillator
- FRCfast: frequency of high-frequency internal RC oscillator
- FPLL: frequency of FMPLL clock

8.10.4.1 Crystal clock monitor

If FOSC is smaller than FRCfast divided by 2^{RCDIV} bits of CMU_CSR and the CK_FXOSC is ‘ON’ as signaled by the MC_ME, then:

- An event pending bit, OLRI in CMU_ISR, is set.
- A failure event OLR is signaled to the MC_RGM, which in turn can automatically switch to a safe fallback clock and generate an interrupt or a reset.

8.10.4.2 PLL clock monitor

The PLL clock CK_PLL frequency can be monitored by programming the CME bit of the CMU_CSR register to 1. CK_PLL monitor starts as soon as CME bit is set. This monitor can be disabled at any time by writing a 0 to the CME bit.

If CK_PLL frequency (FPLL) is greater than a reference value determined by the HFREF[11:0] bits of CMU_HFREFR and the CK_PLL is ‘ON’ as signaled by the MC_ME, then:

- An event pending bit FHHI in CMU_ISR is set.
- A failure event is signaled to the MC_RGM and Fault Collection Unit, which in turn can generate an interrupt or a reset.

If FPLL is less than a reference clock frequency (FRC/4) and the CK_PLL is ‘ON’ as signaled by the MC_ME, then the event pending bit FLCI in CMU_ISR will be set.

If FPLL is less than a reference value determined by the LFREF[11:0] bits of CMU_LFREFR and the CK_PLL is ‘ON’ as signaled by the MC_ME, then:

- An event pending bit FLLI in CMU_ISR is set.
- A failure event FLL is signaled to the MC_RGM, which can generate an interrupt or a reset.

NOTE

The on-chip RC oscillator is used as the reliable reference clock for the clock supervision. In order to avoid false events, proper programming of the dividers is required. These have to take into account the accuracy and frequency deviation of the RC oscillator.

8.10.4.3 Frequency meter

The purpose of the frequency meter is to calibrate the internal RC oscillator (CK_IRC) using a known frequency.

Hint: This value can then be stored in the flash memory so that application software can reuse it later.

The reference clock is always the FXOSC. The frequency meter returns a precise value of CK_32K, CK_FIRC, or CK_SIRC, according to the value of the CKSEL1 bit. The measurement starts when SFM (Start Frequency Measure) bit in CMU_CSR is set to 1. The CMU_MDR register gives the measurement duration in number of clock cycles of the selected clock source, with a width of 20 bits. The SFM bit is reset to 0 by the hardware once the frequency measurement is done and the count is loaded in the

CMU_FDR. The frequency FRC can be derived from the value loaded in the CMU_FDR register as follows:

$$FRC = (FOSC \times MD) / n \qquad \textit{Eqn. 8-2}$$

where *n* is the value in the CMU_FDR register and MD is the value in the CMU_MDR register.

By default, the frequency meter evaluates CK_FIRC, but the user can measure CK_SIRC or CK_SXOSC by programming the CKSEL bits in CMU_CSR register. The CKON bits indicate which clock is at the output of the multiplexer MUX1.

8.10.5 Memory map and register description

The memory map of the CMU is shown in the following table.

Table 8-27. RC Digital Interface Register Set — Base address 0xC3FE_0100

Register Name	Address Offset	Location
Control Status Register (CMU_CSR)	0x00	on page 228
Frequency Display Register (CMU_FDR)	0x04	on page 229
High Frequency Reference Register FMPLL0(CMU_HFREFR_A)	0x08	on page 229
Low Frequency Reference Register FMPLL0(CMU_LFREFR_A)	0x0C	on page 230
Interrupt Status Register (CMU_ISR)	0x10	on page 230
Reserved	0x14	—
Measurement Duration Register (CMU_MDR)	0x18	on page 231

Address Offset	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00	CMU_CSR	Reserved								SFM	Reserved													CKSEL1	Reserved						RCDIV		CME
04	CMU_FDR	Reserved																						FD[12:31]									
08	CMU_HFREFR	Reserved																						HFREF									
0C	CMU_LFREFR	Reserved																						LFREF									
10	CMU_ISR	Reserved																													FHHI_A	FLLI_A	OLRI
14	Reserved	Reserved																															
18	CMU_MDR	Reserved																						MD[12:31]									

Table 8-28. CMU register map

8.10.5.1 Control Status Register (CMU_CSR)

Offset 0x0000 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CKSEL1		0	0	0	0	0	RCDIV		CME_A
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 8-30. Control Status Register (CMU_CSR)

Table 8-29. Control Status Register (CMU_CSR) field descriptions

Field	Description
8 SFM	Start frequency measure The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR register. 0 Frequency measurement is completed or not yet started. 1 Frequency measurement is not completed.
22–23 CKSEL1	RC Oscillator(s) selection bit CKSEL1 selects the clock to be measured by the frequency meter. 00 CK_FIRC is selected. 01 CK_SIRC is selected. 10 CK_SXOSC crystal oscillator clock is selected. 11 CK_FIRC is selected.
29–30 RCDIV[1:0]	RC clock division factor These bits specify the RC clock division factor. The output clock is CK_IRC_{fast} divided by the factor 2^{RCDIV}. This output clock is used to compare with CK_FXOSC for crystal clock monitor feature.The clock division coding is as follows. 00 Clock divided by 1 (No division) 01 Clock divided by 2 10 Clock divided by 4 11 Clock divided by 8
31 CME_A	FMPLL0 clock monitor enable 0 FMPLL0 monitor is disabled. 1 FMPLL0 monitor is enabled.

8.10.5.2 Frequency Display Register (CMU_FDR)

Address: Base + 0x04 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FD[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-31. Frequency display register (CMU_FDR)

Table 8-30. Frequency Display Register (CMU_FDR) field descriptions

Field	Description
12–31 FD	Measured frequency bits This register displays the measured frequency FRC with respect to FOSC. The measured value is given by the following formula: FRC = (FOSC × MD) / *n*, where *n* is the value in the CMU_FDR register.

8.10.5.3 High Frequency Reference Register FMPLL0 (CMU_HFREFR)

Offset: 0x08 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	HFREF[11:0]											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 8-32. High Frequency Reference Register FMPLL0

Table 8-31. High Frequency Reference Register FMPLL0 field descriptions

Field	Description
20–31 HFREF	High Frequency reference value These bits determine the high reference value for the FMPLL0 clock. The reference value is given by: (HFREF[11:0]/16) × (FRC_{fast}/4).

8.10.5.4 Low Frequency Reference Register FMPLL0 (CMU_LFREFR)

Offset: 0x0C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	LFREF[11:0]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-33. Low Frequency Reference Register FMPLL0

Table 8-32. Low Frequency Reference Register FMPLL0 field descriptions

Field	Description
20–31 LFREF	Low Frequency reference value These bits determine the low reference value for the FMPLL0. The reference value is given by: (LFREF[11:0]/16) × (FRC_{fast}/4).

8.10.5.5 Interrupt Status Register (CMU_ISR)

Offset 0x0010 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FHHI	FLLI	OLRI
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-34. Interrupt Status Register (CMU_ISR)

Table 8-33. Interrupt Status Register (CMU_ISR) field descriptions

Field	Description
29 FHHI	FMPLL0 Clock frequency higher than high reference interrupt This bit is set by hardware when CK_FMPLL frequency becomes higher than HFREF value and CK_FMPLL is on as signaled by the MC_ME. It can be cleared by software by writing 1. 0 No FHH event. 1 FHH event is pending.

Table 8-33. Interrupt Status Register (CMU_ISR) field descriptions (continued)

Field	Description
30 FLLI	FMPLL0 Clock frequency less than low reference event This bit is set by hardware when CK_FMPLL frequency becomes lower than LFREF value and CK_FMPLL is on as signaled by the MC_ME. It is cleared by software by writing 1. 0 No FLL event. 1 FLL event is pending.
31 OLRI	Oscillator frequency less than RC frequency event This bit is set by hardware when the frequency of CK_FXOSC is less than CK_FIRC/2^{RCDIV} frequency and CK_FXOSC is on as signaled by the MC_ME. It can be cleared by software by writing 1. 0 No OLR event. 1 OLR event is pending.

8.10.5.6 Measurement Duration Register (CMU_MDR)

Address offset: 0x18 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MD[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-35. Measurement Duration Register (CMU_MDR)

Table 8-34. Measurement Duration Register (CMU_MDR) field descriptions

Field	Description
12–31 MD	Measurement duration bits This register displays the measured duration in terms of IRC clock cycles. This value is loaded in the frequency meter down counter. When SFM bit is set to 1, down counter starts counting.

Chapter 9
Configurable Enhanced Modular IO Subsystem (eMIOS200)

9.1 Device-specific information

This chapter describes the Configurable Enhanced Modular IO Subsystem (eMIOS200).

The eMIOS provides Timer (IC/OC) and PWM functionality. On this device, two eMIOS blocks have channels clocked by either a modulated or a non-modulated clock. eMIOS200_0 is a 16-channel module; eMIOS200_1 is an 8-channel module.

9.1.1 Unsupported features

- Real-time signal bus client
- Wheel speed channels
- eMIOS0 channels 0–7 and channels 24–31
  - Channels 9–15 do not support operations on internal counter
- eMIOS1 channels 0–15 and channels 24–31

9.1.2 Device-specific configuration

- Both eMIOS200_0 and eMIOS200_1 can work on any of the four auxiliary clock sources:
  - IRC
  - FXOSC
  - FMPLL0
  - FMPLL1
- For eMIOS200_0:
  - Counter bus A is driven by Unified Channel #23
  - Counter bus C is driven by Unified Channel #8
  - Counter bus D is driven by Unified Channel #16
  - Unified Channels 9–15 do not have their own time base
- For eMIOS200_1:
  - Counter bus A is driven by Unified Channel #23
  - Counter bus D is driven by Unified Channel #16

9.1.3 eMIOS clocking configuration

The clocking configurations of the eMIOS200_0 and eMIOS200_1 modules on this device are shown in Figure 9-1.

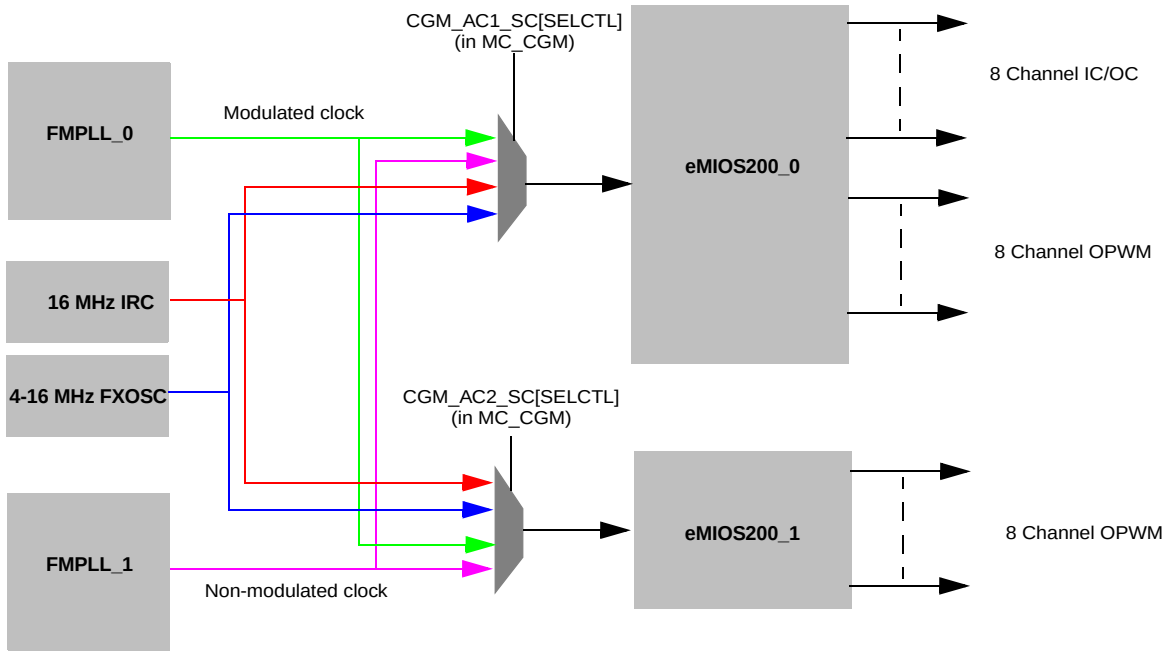


Figure 9-1. eMIOS clocking configuration

9.1.4 MPC5606S family comparison

The following table shows the required split of eMIOS channel functionality across the MPC5606S family as a function of flash memory size.

Table 9-1. eMIOS total channel summary

	256 KB	512 KB	1 MB
16-bit OPWM	16	16	16
16-bit IC/OC	8	8	8

9.1.5 Channel Types

The channels of the eMIOS200_0 and eMIOS200_1 blocks on this device are implemented using a variety of different channel configurations. The available modes of operation for each channel are shown in Table 9-2 and Table 9-3.

Table 9-2. eMIOS200_0 channel configurations

Acronym	Mode	Channels				
		8 IC/OC Counter	9–15 IC/OC	16 PWM Counter	17–22 PWM	23 PWM Counter
GPIO	General Purpose Input/Output	X	X	X	X	X
SAIC	Single Action Input Capture	X	X	X	X	X
SAOC	Single Action Output Compare	X	X	X	X	X
MCB	Modulus Counter Buffered	X		X		X
OPWFMB	Output Pulse Width and Frequency Modulation Buffered			X	X	X
OPWMB	Output Pulse Width Modulation Buffered			X	X	X

Table 9-3. eMIOS200_1 channel configurations

Acronym	Mode	Channels		
		16 PWM Counter	17–22 PWM	23 PWM Counter
GPIO	General Purpose Input/Output	X	X	X
SAIC	Single Action Input Capture	X	X	X
SAOC	Single Action Output Compare	X	X	X
MCB	Modulus Counter Buffered	X		X
OPWFMB	Output Pulse Width and Frequency Modulation Buffered	X	X	X
OPWMB	Output Pulse Width Modulation Buffered	X	X	X

9.1.6 Unified Channel block

Figure 9-2 shows the block diagram of Unified Channel block as it is implemented in this device.

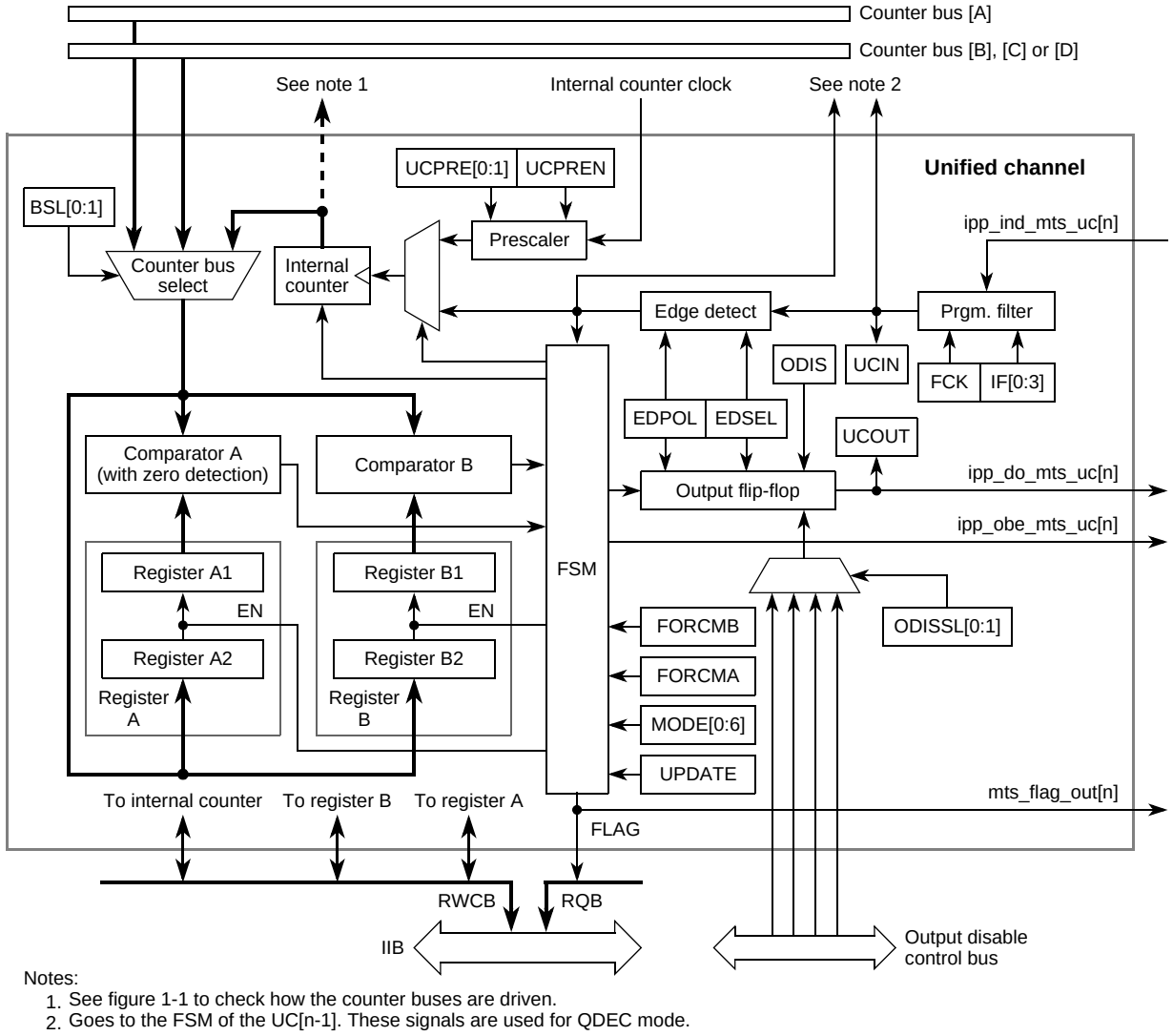


Figure 9-2. Unified Channel block

9.1.6.1 Channel mode selection

The following is a portion of eMIOS200 UC Control Register (EMIOSC[n]). Please see Section 9.4.2.8, eMIOS200 UC Control Register (EMIOSC[n]).

Address Base + 0x0000 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MODE[0:6]						
W																

Figure 9-3. Channel mode selection field

Table 9-4. Channel mode selection field description

Field	Description
bit 25:31 MODE[25:31]	The MODE[0:6] bits select the channel mode operation, as shown in Table 9-5.

Table 9-5. Channel mode selection

MODE[0:6][1]	Mode of operation
0000000	General Purpose Input/Output mode (input)
0000001	General Purpose Input/Output mode (output)
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100 to 1001111	Reserved
101000b	Modulus Counter Buffered (Up counter)
1010010	Reserved
10101bb	Modulus Counter Buffered (Up/Down counter)
10110b0	Output Pulse Width and Frequency Modulation Buffered
10110b1 to 10111b1	Reserved
11000b0	Output Pulse Width Modulation Buffered
1100001 to 1111111	Reserved

1 b = adjust parameters for the mode of operation. Refer to Section 9.5.1.1, UC modes of operation, for details.

9.2 Introduction

Figure 9-4 shows the block diagram of the configurable eMIOS200 block.

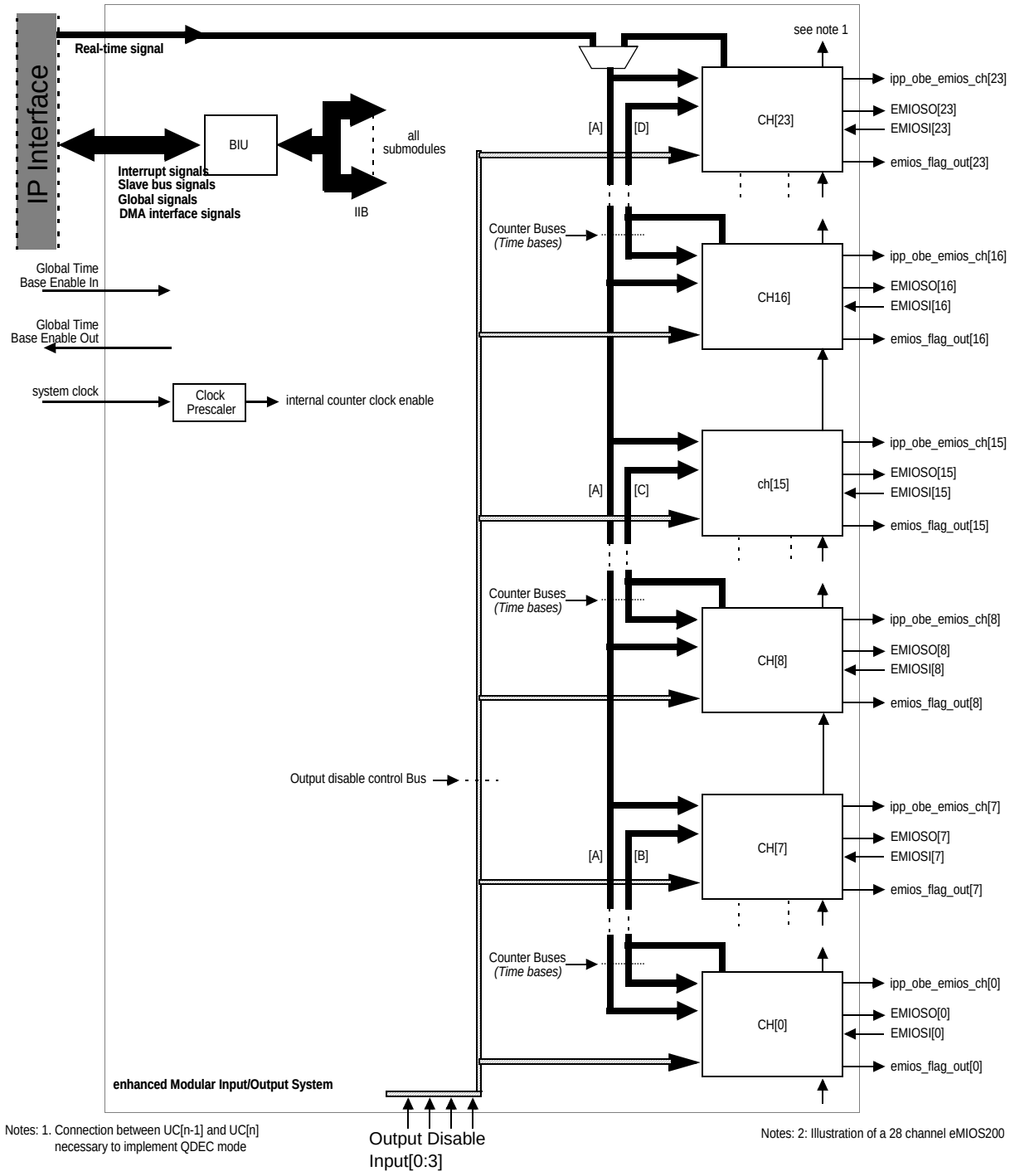


Figure 9-4. eMIOS200 block diagram[1]

1.This diagram shows a 24-channel eMIOS200. On MPC5606S, eMIOS200_0 has 16 channels (8–23) and eMIOS200_1 has 8 channels (16–23). Thus, not all channels shown are available.

9.2.1 Overview

The configurable enhanced Modular Input/Output Subsystem (eMIOS200) provides functionality to generate or measure time events. It is the parameterized version of the eMIOS block, keeping full functional backwards compatibility. Its overall architecture resembles that of the MIOS, shown in Figure 9-4. The predecessor MIOS timer block provides a framework where a set of sublocks with different timer functions are assembled to attend the specific needs of a SoC. The eMIOS200 builds on this concept by using a Unified Channel module that provides a superset of the functionality of all the individual MIOS channels, while providing a consistent user interface. This allows more flexibility as each Unified Channel can be programmed for different functions in different applications of the SoC. Besides that, eMIOS200 architecture allows using Dedicated Channels which perform specific functions not included in MIOS inheritance.

9.2.2 Features

The basic features of the eMIOS200 on this device are the following:

- 24 channels (16 in eMIOS200_0 and 8 in eMIOS200_1) chosen among Unified or Dedicated Channels not necessarily numbered in a continuous sequence
- Data registers of 16-bit width
- Counter buses C and D can be driven by Unified Channel 8 and 16, respectively
- Counter bus A can be driven by the Unified Channel #23 or by the external shared timer bus (Real-Time Signal Bus Client)
- Each channel has its own time base, alternative to the counter buses
- One Global prescaler
- One Prescaler per channel (CP)
- Shared timebases through the counter buses
- Control and status bits grouped in a single register
- Synchronization among timebases
- Global flag register
- State of the UC can be frozen for debug purposes
- Motor control capability

9.2.3 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General purpose input/output (GPIO)
- Single Action Input Capture (SAIC)
- Single Action Output Compare (SAOC)
- Modulus Counter Buffered (MCB)
- Output Pulse Width and Frequency Modulation Buffered (OPWFMB)
- Output Pulse Width Modulation Buffered (OPWMP)

These modes are described in Section 9.5.1.1, UC modes of operation.

Each channel can have a specific set of modes implemented, according to device requirements.

If an unimplemented mode is selected, the results are unpredictable, such as writing a reserved value to MODE[0:6] in Section 9.4.2.8, eMIOS200 UC Control Register (EMIOSC[n]).

9.3 External signal description

9.3.1 Overview

Each channel has one external input and one external output signal, as described in Table 9-6. Depending on the chip integration, the input and output signals can be connected to two separate pins, or to a single bidirectional pin.

9.3.2 Detailed signal descriptions

Table 9-6. External signals

Signal	Direction	Function	Reset State	Pullup
emiosi[*n*]	Input	eMIOS200 Channel n input	—	Chip-dependent
emioso[*n*]	Output	eMIOS200 Channel *n* output	0/ Hi-Z[1]	Chip-dependent
emios_flag_out[*n*]	Output	eMIOS200 Channel *n* flag	0	Chip-dependent

1 Value “0” refers to the reset value of the signal. Hi-Z refers to the state of the external pin if a tristate output buffer is controlled by the corresponding ipp_obe_emios_ch[*n*] signal.

9.3.2.1 emiosi[*n*] — eMIOS200 Channel Input Signal

emiosi[*n*] is synchronized and filtered by the input programmable filter (IPF). The output of the IPF is then used by the channel logic and is available to be read by the MCU through the UCIN bit of the EMIOSS[*n*] register.

9.3.2.2 emioso[*n*] — eMIOS200 Channel Output Signal

emioso[*n*] is a registered output and is available for reading by the MCU through the UCOUT bit of the EMIOSS[*n*] register. Whilst the channel is operating in input modes the signal state is unknown.

9.3.2.3 emios_flag_out[*n*] — eMIOS200 Channel Flag Signal

emios_flag_out[*n*] outputs the state of F[*n*] bit of the EMIOSGFLAG register.

9.4 Memory map and register description

9.4.1 Memory map

The overall address map organization is shown in Table 9-7.

Whenever an access is performed to an absent register, an absent channel, or a reserved address, the eMIOS200 responds asserting Transfer Error signal from the slave bus interface.

Table 9-7. eMIOS200 memory map

eMIOS200[*n*] Base Address	Description	Location
0x000 **0x003**	Module Configuration register (EMIOSMCR)	on page 242
0x004 **0x007**	Global FLAG register (EMIOSGFLAG)	on page 243
0x008 **0x00B**	Output Update Disable (EMIOSOUDIS)	on page 244
0x00C **0x00F**	Disable Channel (EMIOSUCDIS)	on page 245
0x010 **0x11F**	Reserved	—
0x120 **0x21F**	Channel [8] to Channel [15]	
0x220 **0x31F**	Channel [16] to Channel [23]	
0x320 **0xFFF**	Reserved	—

9.4.1.1 Unified Channel memory map

Addresses of Unified Channel registers are specified as offsets from the channel's base address, otherwise the eMIOS200 base address is used as reference.

Table 9-8 describes the Unified Channel memory map.

Table 9-8. Unified Channel memory map

UC[*n*] base address	Description
0x00	A register (EMIOSA[*n*])
0x04	B register (EMIOSB[*n*])
0x08	Counter register (EMIOSCNT[*n*])
0x0C	Control register (EMIOSC[*n*])
0x10	Status register (EMIOSS[*n*])
0x14	Alternate A register (EMIOSALTA[*n*])
0x18–0x1F	Reserved

9.4.2 Register description

All control registers are 32 bits wide. This chapter shows the eMIOS200 with 24 Unified Channels and 16-bit wide data registers.

9.4.2.1 eMIOS200 Module Configuration Register (EMIOSMCR)

The EMIOSMCR contains global control bits for the eMIOS200 block.

Address: eMIOS200 base address +0x00 Access: User read/write

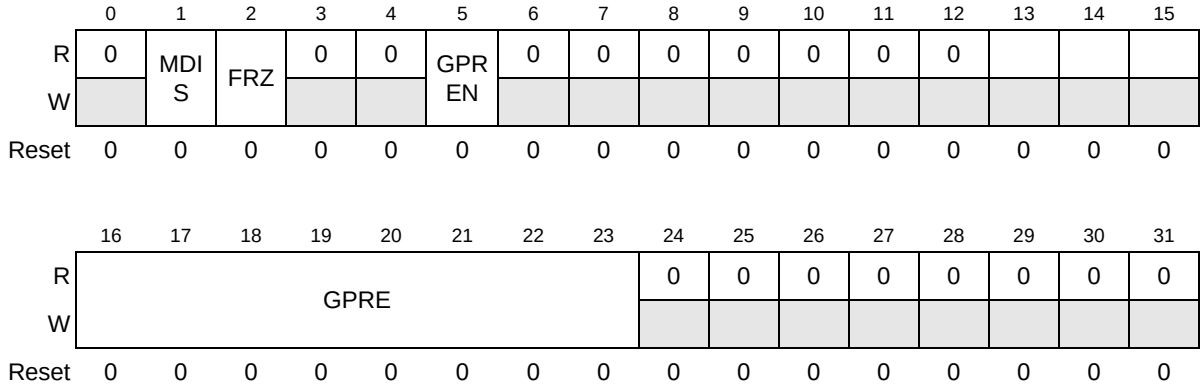


Figure 9-5. eMIOS200 Module Configuration Register (EMIOSMCR)

Table 9-9. EMIOSMCR field descriptions

Field	Description
MDIS	Module Disable Puts the eMIOS200 in low-power mode. The MDIS bit is used to stop the clock of the block, except the access to registers EMIOSMCR, EMIOSOUDIS, and EMIOSUCDIS. 0 Clock is running 1 Enter low-power mode
FRZ	Freeze Enable the eMIOS200 to freeze the registers of the Unified Channels when the MCU is stopped by a debugger. Each Unified Channel should have FREN bit set in order to enter freeze state. While in Freeze state, the eMIOS200 continues to operate to allow the MCU access to the Unified Channel registers. The Unified Channel will remain frozen until the FRZ bit is written to zero, or the MCU exits Debug mode, or the Unified Channel FREN bit is cleared. 0 Exit freeze state 1 Stops Unified Channels operation when in Debug mode and the FREN bit is set in the EMIOSC[*n*] register
GPREN	Global Prescaler Enable The GPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
GPRE	Global Prescaler The GPRE[0:7] bits select the clock divider value for the global prescaler, as shown in Table 9-10.

..

Table 9-10. Global Prescaler clock divider

GPRE[0:7]	Divide ratio
00000000	1
00000001	2
00000010	3
00000011	4
. . . .	. . . .
11111110	255
11111111	256

9.4.2.2 eMIOS200 Global FLAG Register (EMIOSGFLAG)

The EMIOSGFLAG is a read-only register that groups the FLAG bits from all channels. This organization improves interrupt handling on simpler devices. Each bit relates to one channel.

The two modules on this device, EMIOS0 and EMIOS1, have different structures for this register as shown in Figure 9-6 and Figure 9-7.

For Unified Channels these bits are mirrors of the FLAG bits in the EMIOSS[*n*] register.

Address: eMIOS0 base address +0x04 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	F23	F22	F21	F20	F19	F18	F17	F16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F15	F16	F17	F18	F19	F10	F9	F8	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-6. eMIOS200 Global FLAG Register (EMIOSGFLAG) for EMIOS0

Address: eMIOS1 base address +0x04 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	F23	F22	F21	F20	F19	F18	F17	F16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 9-7. eMIOS200 Global FLAG Register (EMIOSGFLAG) for EMIOS1

Table 9-11. EMIOSGFLAG field descriptions

Field	Description
F[*n*]	Channel [*n*] Flag bit

Channels that occupy a pair of slots are referred to by their lower slot number (LSB = 0 standard), therefore the bits corresponding to their higher slot number always read 0.

9.4.2.3 eMIOS200 Output Update Disable (EMIOSOUDIS)

The two modules on this device, EMIOS0 and EMIOS1, have different structures for this register as shown in Figure 9-8 and Figure 9-9.

Address: eMIOS0 base address +0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	OU 23	OU 22	OU 21	OU 20	OU 19	OU 18	OU 17	OU 16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OU 15	OU 14	OU 13	OU 12	OU 11	OU 10	OU 9	OU 8	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-8. eMIOS200 Output Update Disable Register (EMIOSOUDIS) for EMIOS0

Address: eMIOS1 base address +0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	OU 23	OU 22	OU 21	OU 20	OU 19	OU 18	OU 17	OU 16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 9-9. eMIOS200 Output Update Disable Register (EMIOSOUDIS) for EMIOS1

Table 9-12. EMIOSOUDIS field descriptions

Field	Description
OU[*n*]	Channel [*n*] Output Update Disable bit When running MCB or an output mode, values are written to registers A2 and B2. OU[*n*] bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled

9.4.2.4 eMIOS200 Disable Channel (EMIOSUCDIS)

The two modules on this device, EMIOS0 and EMIOS1, have different structures for this register as shown in Figure 9-10 and Figure 9-11.

Address: eMIOS0 base address +0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CHDI S23	CHDI S22	CHDI S21	CHDI S20	CHDI S19	CHDI S18	CHDI S17	CHDI S16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHDI S15	CHDI S14	CHDI S13	CHDI S12	CHDI S11	CHDI S10	CHDI S9	CHDI S8	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-10. eMIOS200 Enable Channel Register (EMIOSUCDIS) for EMIOS200_0

Address: eMIOS1 base address +0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CHDIS23	CHDIS22	CHDIS21	CHDIS20	CHDIS19	CHDIS18	CHDIS17	CHDIS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 9-11. eMIOS200 Enable Channel Register (EMIOSUCDIS) for EMIOS200_1

Table 9-13. EMIOSUCDIS field descriptions

Field	Description
CHDIS[*n*]	Enable Channel [*n*] bit The CHDIS[*n*] bit is used to disable each of the channels by stopping its respective clock. 0 Channel [*n*] enabled 1 Channel [*n*] disabled **Note:** Channels that occupy a pair of slots are referred to by their lower slot number (LSB = 0 standard), therefore the bits corresponding to their higher slot number are reserved and read 0.

9.4.2.5 eMIOS200 UC A Register (EMIOSA[*n*])

Address: UC[*n*] base address + 0x00 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-12. eMIOS200 UC A Register (EMIOSA[*n*])

Depending on the mode of operation, internal registers A1 or A2, used for matches and captures, can be assigned to address EMIOSA[*n*]. Both A1 and A2 are cleared by reset. Table 9-14 summarizes the EMIOSA[*n*] read and write accesses for all operation modes. For more information, see section Section 9.5.1.1, UC modes of operation.

9.4.2.6 eMIOS200 UC B Register (EMIOSB[*n*])

Address: UC[*n*] base address + 0x04 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	B															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-13. eMIOS200 UC B register (EMIOSB[*n*])

Depending on the mode of operation, internal registers B1 or B2 can be assigned to address EMIOSB[*n*]. Both B1 and B2 are cleared by reset. Table 9-14 summarizes the EMIOSB[*n*] read and write accesses for all operation modes. For more information, see section Section 9.5.1.1, UC modes of operation.

Depending on the channel configuration, it may or may not have the EMIOSB register. The EMIOSB register is required for the following modes: OPWMB, OPWFMB, and MCB.

Table 9-14. EMIOSA[*n*], EMIOSB[*n*], and EMIOSALTA[*n*] value assignments

Mode of operation	Register access					
	Write	Read	Write	Read	Alt write	Alt read
GPIO	A1, A2	A1	B1, B2	B1	A2	A2
SAIC[1]	—	A2	B2	B2	—	—
SAOC[1]	A2	A1	B2	B2	—	—
MCB[1]	A2	A1	B2	B2	—	—
OPWFMB	A2	A1	B2	B1	—	—
OPWMB	A2	A1	B2	B1	—	—

1 In these modes, the EMIOSB[*n*] register is not used, but B2 can be accessed.

9.4.2.7 eMIOS200 UC Counter Register (EMIOSCNT[*n*])

Address: UC[*n*] base address + 0x08 Access: User read/write

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

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

Figure 9-14. eMIOS200 UC Counter Register (EMIOSCNT[*n*])

1 In GPIO mode or freeze action, this register is writable.

The EMIOSCNT[*n*] register contains the value of the internal counter. When GPIO mode is selected or the channel is frozen, the EMIOSCNT[*n*] register is read/write. For all others modes, the EMIOSCNT[*n*] is a read-only register. When entering some operation modes, this register is automatically cleared. See Section 9.5.1.1, UC modes of operation, for details.

Depending on the channel configuration, it may or may not have the EMIOSCNT register. The EMIOSCNT register is required for the following modes: OPWFMB and MCB.

It is possible that, for particular reasons, EMIOSCNT may be available on one device even if the respective channel does not feature any mode that requires it. In this case, EMIOSCNT availability should be explicitly described in the device SoC guide.

9.4.2.8 eMIOS200 UC Control Register (EMIOSC[*n*])

The Control register gathers bits reflecting the status of the UC input/output signals and the overflow condition of the internal counter, as well as several read/write control bits.

Address: UC[*n*] base address + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FREN	ODIS	ODISSL[0:1]		UCPRE[0:1]		UCPREN	DMA	0	IF[0:3]				FCK	FEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BSL[0:1]		EDSEL	EDPOL	MODE[0:6]						
W			FORCMA	FORCMB												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-15. eMIOS200 UC Control Register (EMIOSC[*n*])

Table 9-15. EMIOSC[*n*] field descriptions

Field	Description
FREN	Freeze Enable bit The FREN bit, if set and validated by the FRZ bit in EMIOSMCR register, allows the channel to enter freeze state. This freezes all register values when in debug mode and allows the MCU to perform debug functions. 0 Normal operation 1 Freeze UC register values
ODIS	Output Disable bit The ODIS bit allows disabling the output pin when running any of the output modes, with the exception of GPIO mode. 0 The output pin operates normally 1 If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, that is, it continues to produce FLAG and matches; when the selected Output Disable Input signal is negated, the output pin operates normally.
ODISSL[0:1]	Output Disable select bits The ODISSL[0:1] bits select one of the four output disable input signals, as shown in Table 9-16.
UCPRE[0:1]	Prescaler bits The UCPRE[0:1] bits select the clock divider value for the internal prescaler of Unified Channel, as shown in Table 9-17.
UCPREN	Prescaler Enable bit The UCPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock)0:1 1 Prescaler enabled
DMA	Direct Memory Access bit The DMA bit selects if the FLAG generation will be used as an interrupt or as a DMA request. 0 Flag/overrun assigned to interrupt request 1 Flag/overrun assigned to DMA request
IF[0:3]	Input Filter bits The IF[0:3] bits control the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in Table 9-18. For output modes, these bits have no meaning.

Table 9-15. EMIOSC[*n*] field descriptions (continued)

Field	Description
FCK	Filter Clock select bit The FCK bit selects the clock source for the programmable input filter. 0 Prescaled clock 1 Main clock
FEN	FLAG Enable bit The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal (the type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt or DMA request) 1 Enable (FLAG will generate an interrupt or DMA request)
FORCMA	Force Match A bit For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A; otherwise it has no effect. 0 Has no effect 1 Force a match at comparator A **Note:** For input modes, the FORCMA bit is not used and writing to it has no effect.
FORCMB	Force Match B bit For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B; otherwise it has no effect. 0 Has no effect 1 Force a match at comparator B **Note:** For input modes, the FORCMB bit is not used and writing to it has no effect.
BSL[0:1]	Bus Select bits The BSL[0:1] bits are used to select either one of the counter buses or the internal counter to be used by the Unified Channel. Refer to Table 9-19 for details.
EDSEL	Edge Selection bit For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or only by a single edge, as defined by the EDPOL bit. When not shown in the mode of operation description, this bit has no effect. 0 Single-edge triggering defined by the EDPOL bit 1 Both edges triggering For GPIO mode, the EDSEL bit selects whether a FLAG can be generated. 0 A FLAG is generated as defined by the EDPOL bit 1 No FLAG is generated For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match. 0 The EDPOL value is transferred to the output flip-flop 1 The output flip-flop is toggled

Table 9-15. EMIOSC[*n*] field descriptions (continued)

Field	Description
EDPOL	Edge Polarity bit For input modes (except QDEC mode), the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG. When not shown in the mode of operation description, this bit has no effect. 0 Trigger on a falling edge 1 Trigger on a rising edge For QDEC (MODE[6] cleared), the EDPOL bit selects the count direction according to the direction signal (UC[*n*] input). 0 Counts down when UC[*n*] is asserted 1 Counts up when UC[*n*] is asserted **Note:** The UC[*n* – 1] EDPOL bit selects which edge clocks the internal counter of UC[*n*]. 0 Trigger on a falling edge 1 Trigger on a rising edge For QDEC (MODE[6] set), the EDPOL bit selects the count direction according to the phase difference. 0 Internal counter decrements if phase_A is ahead of the phase_B signal 1 Internal counter increments if phase_A is ahead of the phase_B signal **Note:** In order to operate properly, EDPOL bit must contain the same value in UC[*n*] and UC[*n* – 1]. For output modes, the EDPOL bit is used to select the logic level on the output pin. 0 A match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 A match on comparator A sets the output flip-flop, while a match on comparator B clears it
MODE[0:6]	Mode selection bits The MODE[0:6] bits select the mode of operation of the Unified Channel, as shown in Table 9-20. **Note:** If a reserved value is written to MODE, the results are unpredictable.

Table 9-16. UC ODISSL selection

ODISSL[0:1]	eMIOS200_0 channel	eMIOS200_1 channel	Input signal
00	emios_flag_out[8]	emios_flag_out[16]	Output Disable Input 0
01	emios_flag_out[9]	emios_flag_out[17]	Output Disable Input 1
10	emios_flag_out[10]	emios_flag_out[18]	Output Disable Input 2
11	emios_flag_out[11]	emios_flag_out[19]	Output Disable Input 3

Table 9-17. UC internal prescaler clock divider

UCPRE[0:1]	Divide ratio
00	1
01	2
10	3
11	4

Table 9-18. UC input filter bits

IF[0:3][1]	Minimum input pulse width [FLT_CLK periods]
0000	Bypassed[2]
0001	02
0010	04
0100	08
1000	16
all others	Reserved

1 Filter latency is 3 clock edges.

2 The input signal is synchronized before arriving at the digital filter.

Table 9-19. UC BSL bits

BSL[0:1]	Selected bus
00	All channels: counter bus[A]
01	Channels 8 to 15: counter bus[C] Channels 16 to 23: counter bus[D]
10	Reserved
11	All channels: internal counter

Table 9-20. UC MODE bits

MODE[0:6][1]	Mode of operation
0000000	General Purpose Input/Output mode (input)
0000001	General Purpose Input/Output mode (output)
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100 through 1001111	Reserved
101000b	Modulus Counter Buffered (Up counter)
101001b	Reserved
10101bb	Modulus Counter Buffered (Up/Down counter)
10110b0	Output Pulse Width and Frequency Modulation Buffered
10110b1	Reserved
10111b0	Reserved
10111b1	Reserved
11000b0	Output Pulse Width Modulation Buffered
1100001 through 1111111	Reserved

1 b = adjust parameters for the mode of operation. Refer to Section 9.5.1.1, UC modes of operation, for details.

9.4.2.9 eMIOS200 UC Status Register (EMIOSS[*n*])

Address: UC[*n*] base address + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVFL	0	0	0	0	0	0	0	0	0	0	0	0	UCIN	UCOUT	FLAG
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-16. eMIOS200 UC Status Register (EMIOSS[*n*])

Table 9-21. EMIOSS[*n*] field descriptions

Field	Description
OVR	Overrun bit The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set. This bit can be cleared by writing a 1 to it or by writing a 1 to the FLAG bit. 0 Overrun has not occurred 1 Overrun has occurred
OVFL	Overflow bit The OVFL bit indicates that an overflow has occurred in the internal counter. OVFL must be cleared by software writing a 1 to it. 0 No overflow 1 An overflow has occurred
UCIN	Unified Channel Input pin bit The UCIN bit reflects the input pin state after being filtered and synchronized.
UCOUT	Unified Channel Output pin bit The UCOUT bit reflects the output pin state.
FLAG	FLAG bit The FLAG bit is set when an input capture or a match event in the comparators occurred. To clear this bit, write a 1 to it. 0 FLAG cleared 1 FLAG set event has occurred

NOTE

emios_flag_out reflects the FLAG bit value. When DMA bit is set, the FLAG bit can be cleared by the DMA controller.

9.4.2.10 eMIOS200 UC Alternate A Register (EMIOSALTA[*n*])

EMIOSALTA[*n*] address: UC[*n*] base address + 0x14

Address: UC[n] base address + 0x14 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ALTA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-17. eMIOS200 UC Alternate A Register (EMIOSALTA[*n*])

The EMIOSALTA[*n*] register provides an alternate address to access A2 channel registers in restricted modes (GPIO) only. If the EMIOSA[*n*] register is used along with EMIOSALTA[*n*], both A1 and A2 registers can be accessed in these modes. Table 9-14 summarizes the EMIOSALTA[*n*] read and write accesses for all operation modes.

9.5 Functional description

The eMIOS200 provides independent channels (UC) that can be configured and accessed by a host MCU. Up to four time bases can be shared by the channels through four counter buses, and each channel can generate its own time base. Optionally one of the counter buses can be driven by an external time base imported through the real-time signal interface.

The eMIOS200 module is based on a multi-bus timer architecture in which several timer channels are used to drive counter buses that are shared among the channels. There are four counter buses in the module: one global counter bus, shared by all channels and four local counter buses, each one dedicated to a slice of eight channels. Counter bus A is referred to as the global counter bus. Counter buses B, C, and D are the local counter buses.

The eMIOS200 counter buses are driven by channels in specific locations. The global counter bus is driven by the channel in channel slot [23]. Counter buses B, C, and D are driven by channels in slots [0], [8], and [16], respectively. Counter bus A drives all channels. Counter bus B drives channels in slots from [0] through [7]. Counter bus C drives channels in slots from [8] through [15]. Counter bus D drives channels in slots from [16] through [23]. Note that the first channel in an eight-channel slice drives the local counter bus for that slice—therefore, this channel should not be assigned to be driven by the same counter bus, or else a loop will occur. The eMIOS200 interrupt request signal, DMA transfer request signal, and others, are wired to a specific channel—thus the chip integrator should connect those signals having the eMIOS200 channel configuration in mind.

The eMIOS200 block is reset asynchronously. All registers are cleared on reset.

Figure 9-18 describes an eMIOS200 block configured with 32 Unified Channels. Note that the Red Line is also present. Note also that independent of the configuration the channels are fixed in their slots. Thus

for example if channel [2] is not required this location will be empty, meaning that the other channel locations are not affected. In this case the application software should not access any register located in the channel[2] memory. Any attempt to access those registers will return no meaningful data, and a transfer error will be generated.

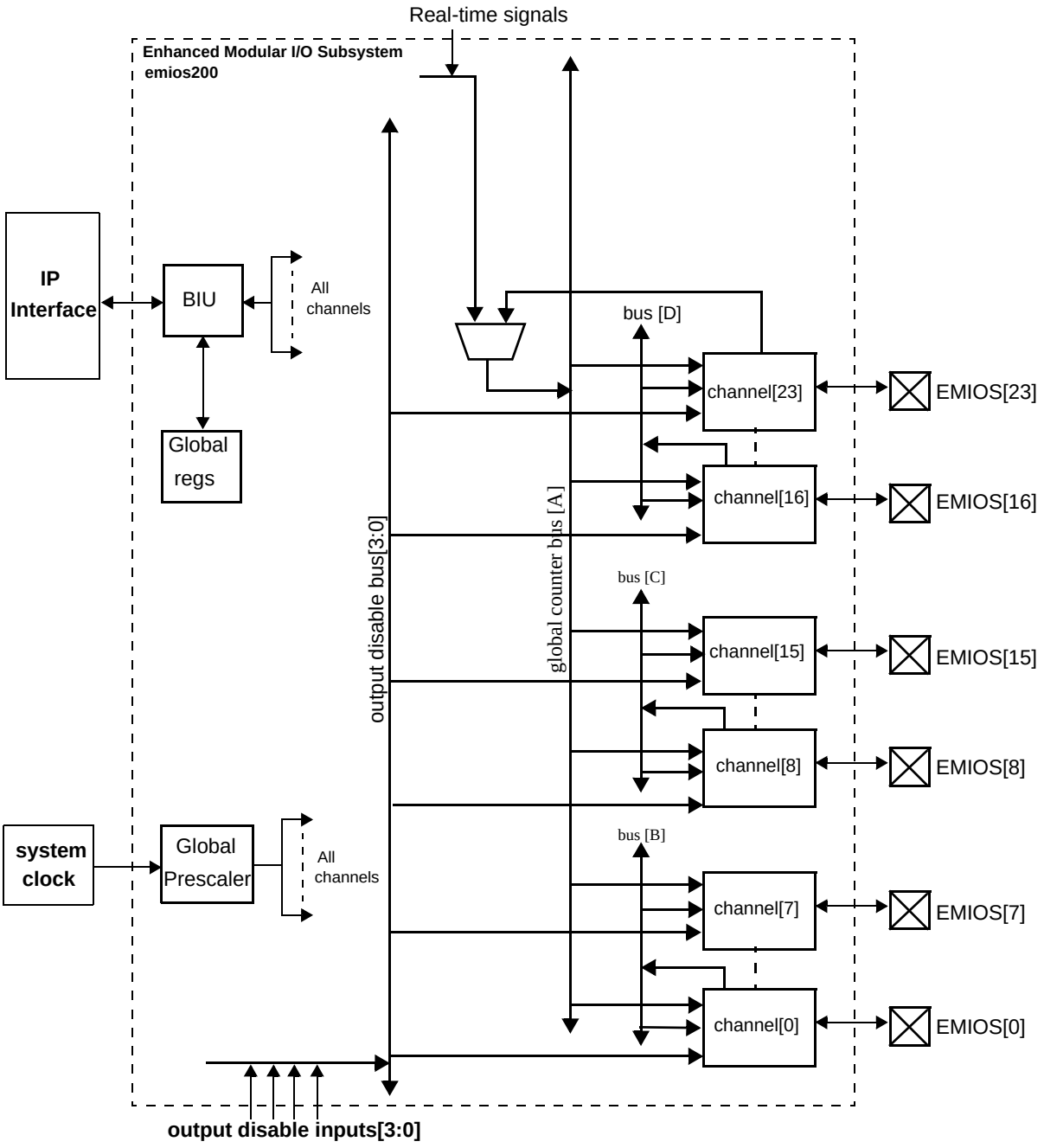


Figure 9-18. eMIOS200 full channel configuration using Unified Channels only

9.5.1 Unified Channel (UC)

Figure 9-19 shows the Unified Channel block diagram. Each Unified Channel consists of:

- Counter bus selector, which selects the time base to be used by the channel for all timing functions
- Programmable clock prescaler
- Two double-buffered data registers A and B that allow up to two input capture and/or output compare events to occur before software intervention is needed
- Two comparators (equal only) A and B, which compare the selected counter bus with the value in the data registers
- Internal counter, which can be used as a local time base or to count input events
- Programmable input filter, which ensures that only valid pin transitions are received by channel
- Programmable input edge detector, which detects the rising, falling, or either edge
- An output flip-flop, which holds the logic level to be applied to the output pin
- eMIOS200 Status and Control register
- An Output Disable Input selector, which selects the Output Disable Input signal that will be used as output disable

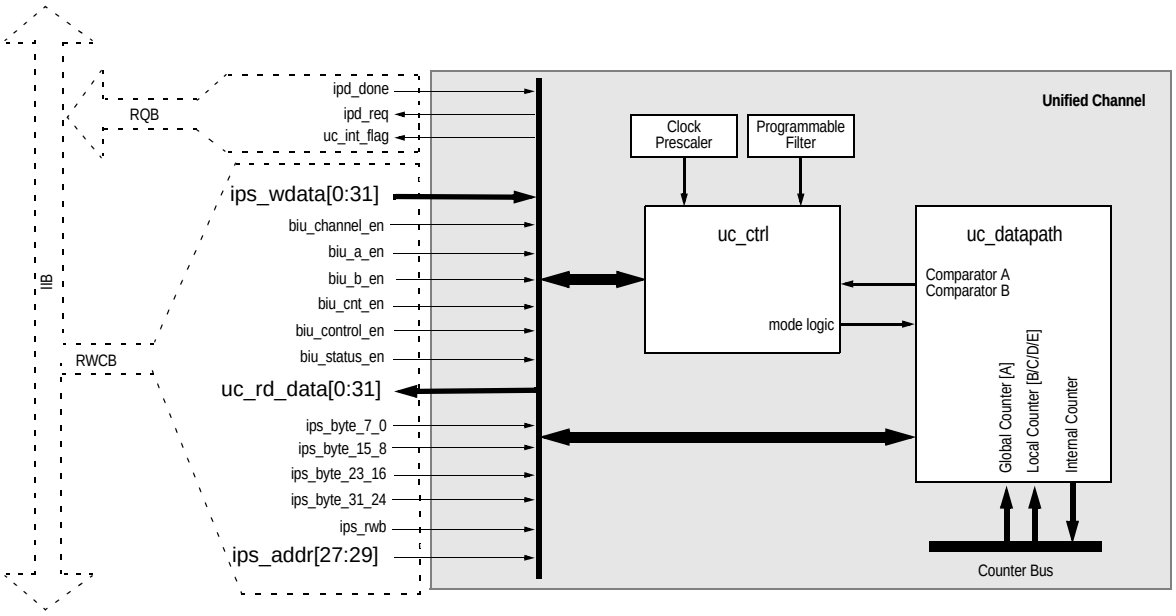


Figure 9-19. Unified Channel block diagram

Figure 9-20 shows both the Unified Channel Control and Datapath block diagram. The Control block is responsible for the generation of signals to control the multiplexes in the Datapath subblock. Each mode is implemented by a dedicated logic independent from other modes, thus allowing optimization the logic by disabling the mode and therefore its associated logic. The unused gates are removed during the synthesis phase. Targeting the logic optimization, a set of registers is shared by the modes, thus providing sequential events to be stored.

The Datapath block provides the channel A and B registers, the internal time base, and comparators. Multiplexors select the input of comparators and data for the register inputs, thus configuring the data path in order to implement the channel modes. The outputs of A and B comparators are connected to the uc_ctrl control block.

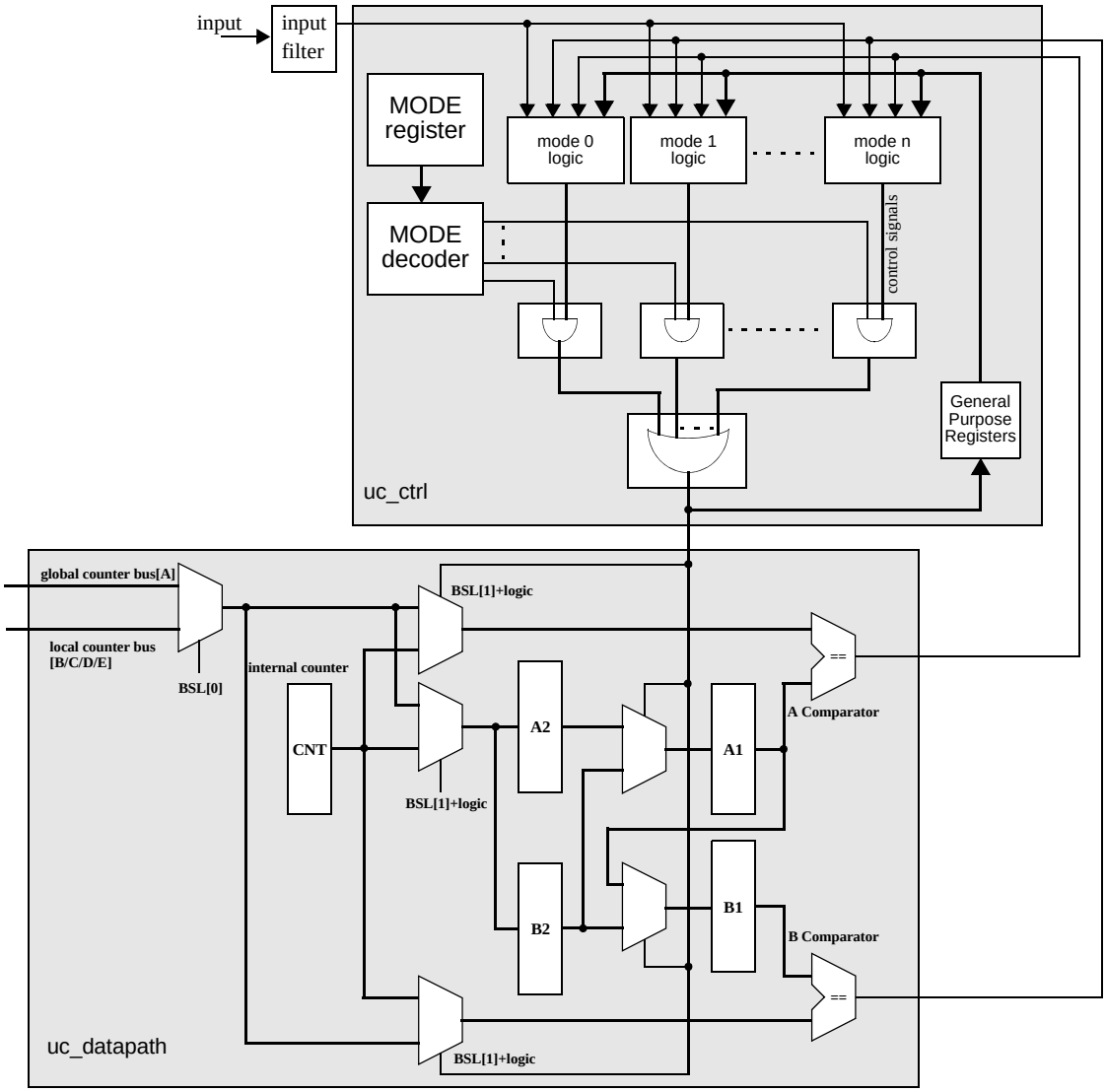


Figure 9-20. Unified Channel control and datapath block diagrams

9.5.1.1 UC modes of operation

The mode of operation of the Unified Channel is determined by the mode select bits MODE[0:6] in the EMIOSC[*n*] register (see Figure 9-20 for details).

When entering an output mode (except for GPIO mode), the output flip-flop is set to the disabled state according to the ODIS bit in the EMIOSC[*n*] register.

As the internal counter EMIOSCNT[*n*] continues to run in all modes (except for GPIO mode), it is possible to use this as a time base if the resource is not used in the current mode.

In order to provide smooth waveform generation even if A and B registers are changed on the fly, the MCB, OPWFMB, and OPWMB modes are available. In these modes A and B registers are double-buffered.

9.5.1.1.1 General Purpose Input/Output mode (GPIO) mode

In GPIO mode, all input capture and output compare functions of the UC are disabled, and the internal counter (EMIOSCNT[*n*] register) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operation mode, writing to registers EMIOSA[*n*] or EMIOSB[*n*] stores the same value in registers A1/A2 or B1/B2, respectively. Writing to register EMIOSALTA[*n*] stores a value only in register A2.

MODE[6] bit selects between input (MODE[6] = 0) and output (MODE[6] = 1) modes.

It is required that when changing MODE[0:6], the application software goes to GPIO mode first in order to reset the UC's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results or the FLAGs being set incorrectly.

In GPIO input mode (MODE[0:6]=0000000), the FLAG generation is determined according to the EDPOL and EDSEL bits, and the input pin status can be determined by reading the UCIN bit.

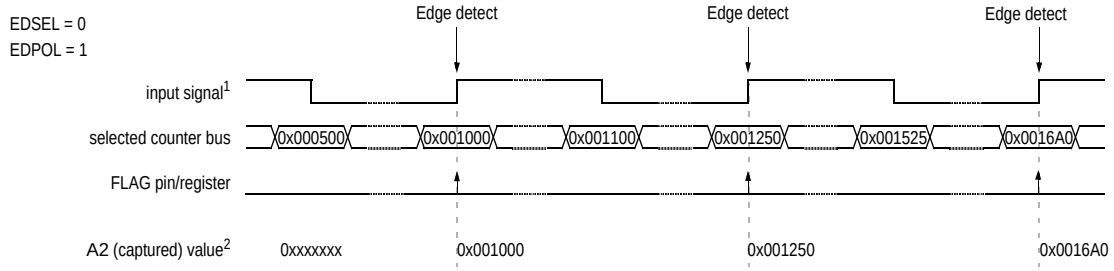
In GPIO output mode (MODE[0:6]=0000001), the Unified Channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

9.5.1.1.2 Single Action Input Capture (SAIC) mode

In SAIC mode (MODE[0:6]=0000010), when a triggering event occurs on the input pin, the value on the selected time base is captured in register A2. The FLAG bit is set along with the capture event to indicate that an input capture has occurred. Register EMIOSA[*n*] returns the value of register A2. As soon as the SAIC mode is entered coming out from GPIO mode, the channel is ready to capture events. The events are captured as soon as they occur—thus reading register A always returns the value of the latest captured event. Subsequent captures are enabled with no need of further reads from EMIOSA[*n*] register. The FLAG is set whenever a new event is captured.

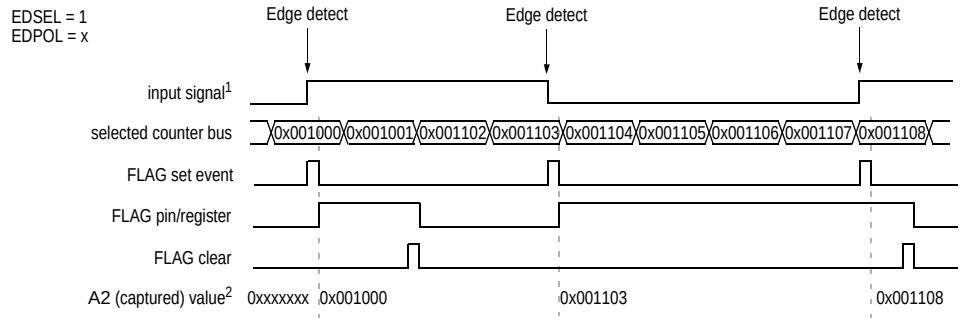
The input capture is triggered by a rising, falling, or either edge on the input pin, as configured by the EDPOL and EDSEL bits in the EMIOSC[*n*] register.

Figure 9-21 and Figure 9-22 show how the Unified Channel can be used for input capture.



Notes: 1. After input filter
2. EMIOSA[*n*] <= A2

Figure 9-21. Single Action Input Capture with rising edge triggering example



Notes: 1. After input filter
2. EMIOSA[*n*] <= A2

Figure 9-22. Single Action Input Capture with both edges triggering example

9.5.1.1.3 Single Action Output Compare (SAOC) mode

In SAOC mode (MODE[0:6]=0000011), a match value is loaded in register A2 and then immediately transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects whether the output flip-flop is toggled or the value in EDPOL is transferred to it. Along with the match the FLAG bit is set to indicate that the output compare match has occurred. Writing to register EMIOSA[*n*] stores the value in register A2, and reading register EMIOSA[*n*] returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in the EMIOSC[*n*] register. In this case, the FLAG bit is not set.

When SAOC mode is entered coming out of GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[*n*] register.

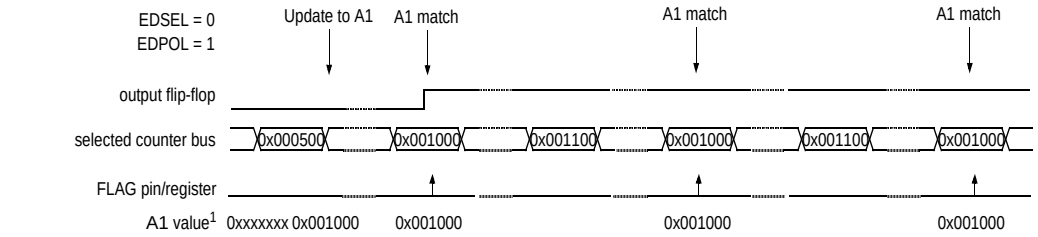
The counter bus can be either internal or external and is selected through BSL[0:1] bits.

Figure 9-23 and Figure 9-24 show how the Unified Channel can be used to perform a single output compare with the EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively. Note that once in SAOC mode the matches are enabled; thus the desired match value on register A1 must be written before the mode is entered. A1 register can be updated at any time, thus modifying the match value which will be reflected in the output signal generated by the channel.

Subsequent matches are enabled with no need of further writes to the EMIOSA[*n*] register. The FLAG is set at the same time a match occurs.

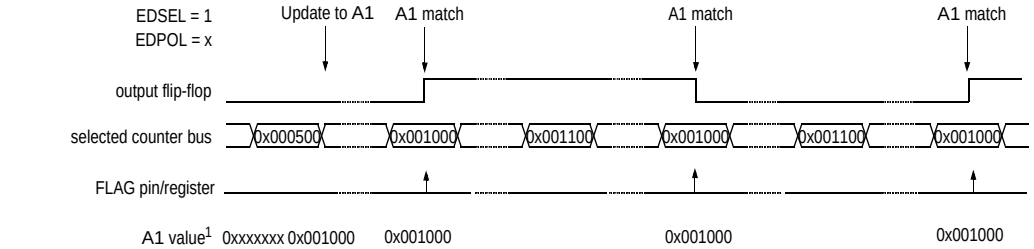
NOTE

The channel internal counter in SAOC mode is free-running. It starts counting as soon as SAOC mode is entered.



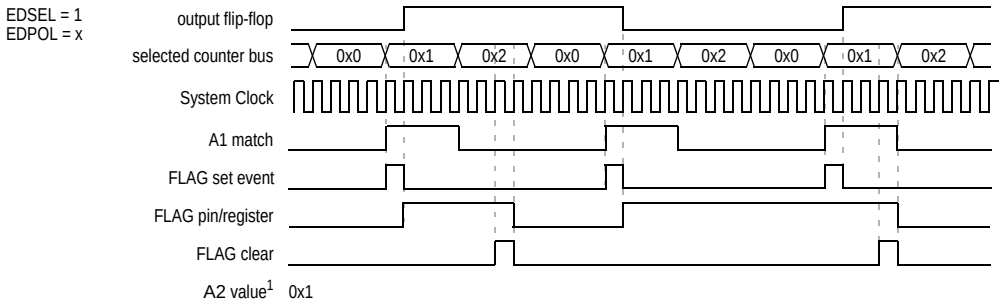
Notes: 1. EMIOSA[*n*] = A2
A2 = A1 according to OU[*n*] bit

Figure 9-23. SAOC example with EDPOL value being transferred to the output flip-flop



Notes: 1. EMIOSA[n] = A2
A2 = A1 according to OU[*n*] bit

Figure 9-24. SAOC example toggling the output flip-flop



Note: 1. EMIOSA[*n*] <= A2

9.5.1.1.4 Modulus Counter Buffered (MCB) mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double-buffered, thus allowing smooth transitions between cycles when changing the value of the A2 register on the fly. The A1 register is updated at the cycle boundary, which is defined as when the internal counter transitions to 0x1.

The internal counter values operate within a range from 0x1 up to the value of the A1 register. If when entering MCB mode coming out of GPIO mode, the internal counter value is not within that range, then the A match will not occur. This will cause the channel internal counter to wrap at the maximum counter value, which is 0xFFFF for a 16-bit counter. After the counter wrap occurs, it returns to 0x1 and resumes normal MCB mode operation. Thus in order to avoid the counter wrap condition, make sure the internal counter value is within the 0x1 to A1 register value range when MCB mode is entered.

The MODE[6] bit selects the internal clock source if cleared, or external if set. When the external clock is selected, the input channel pin is used as the channel clock source. The active edge of this clock is defined by the EDPOL and EDSEL bits in the EMIOSC[*n*] channel register.

When entering MCB mode, if the up counter is selected by MODE[4]=0 (MODE[0:6]=101000b), the internal counter starts counting up from its current value until a match with A1 occurs. The internal counter is set to 0x1 when its value matches the A1 value and a clock tick occurs (either prescaled clock or input pin event).

If the up/down counter is selected by setting MODE[4]=1, the counter changes direction at A1 match and counts down until it reaches the value 0x1. After it has reached 0x1, it is set to count up again. The B1 register is used to generate a match in order to set the internal counter in the up-count direction if up/down mode is selected. Register B1 cannot be changed while this mode is selected.

Note that MCB mode counts between 0x1 and the value in the A1 register. Only values greater than 0x1 may be written to the A1 register. Loading any other values leads to unpredictable results. The counter cycle period is equal to the A1 value in up counter mode. If in up/down counter mode, the period is defined by the expression $(2 \times A1) - 2$.

Figure 9-25 shows the counter cycle for several possible A1 values. Register A1 is loaded with the A2 register value at the cycle boundary. Thus, any value written to the A2 register within cycle *n* will be updated to A1 at the next cycle boundary, and therefore will be used on cycle $n + 1$. The cycle boundary between cycle *n* and cycle $n + 1$ is defined as when the internal counter transitions from the A1 value in cycle *n* to 0x1 in cycle $n + 1$. Note that the FLAG is generated at the cycle boundary and has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

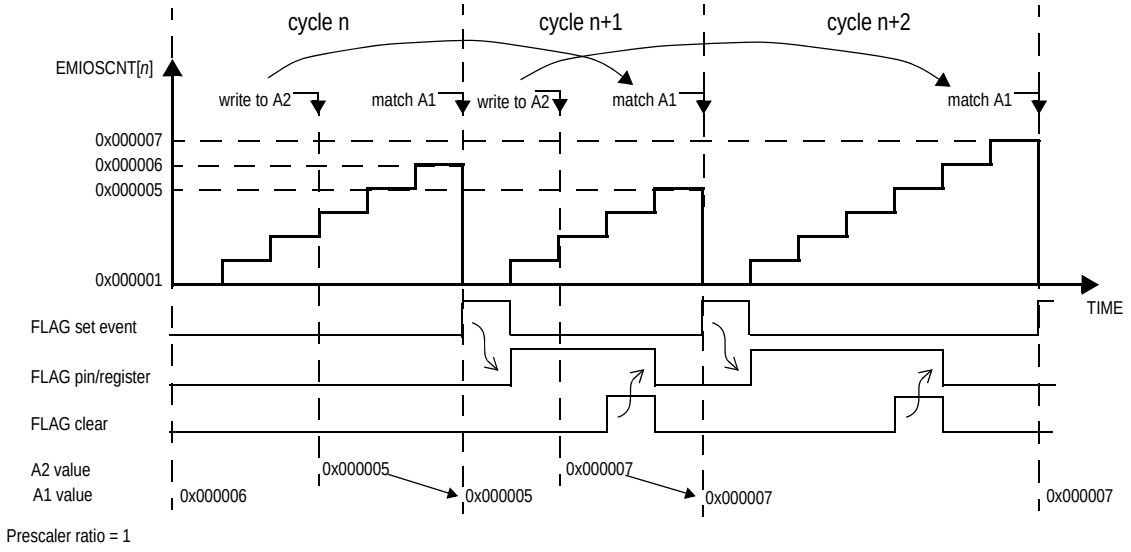


Figure 9-25. Modulus Counter Buffered (MCB) up count mode

Figure 9-26 shows the MCB in up/down counter mode (MODE[0:6]=10101bb). The A1 register is updated at the cycle boundary. If A2 is written in cycle *n*, this new value will be used in cycle *n* + 1 for A1 match. Flags are generated only at A1 match start if MODE[5] is 0. If MODE[5] is set to 1, flags are also generated at the cycle boundary.

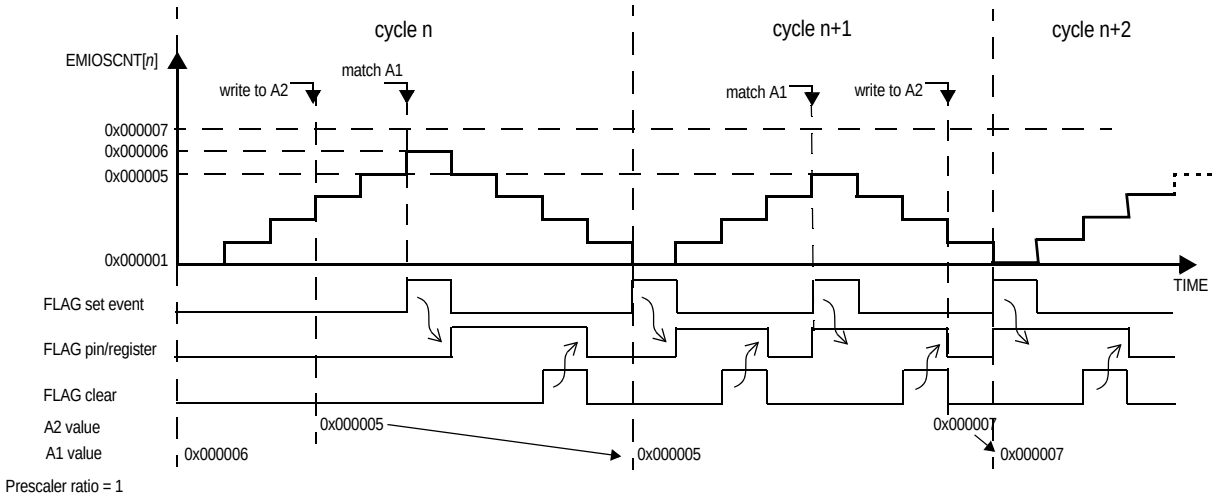


Figure 9-26. Modulus Counter Buffered (MCB) up/down mode

Figure 9-27 describes in more detail the A1 register update process in up counter mode. The A1 load signal is generated at the last system clock period of a counter cycle. Thus, A1 is updated with the A2 value at the same time that the counter (EMIOSCNT[*n*]) is loaded with 0x1. The load signal pulse has the duration of one system clock period. If A2 is written within cycle *n* + 1, its value is available at A1 at the first clock of cycle *n* + 1, and the new value is used for match at cycle *n* + 1. The update disable bits OU[*n*] of the EMIOSOUDIS register can be used to control the update of this register, thus allowing delay of the A1 register update for synchronization purposes.

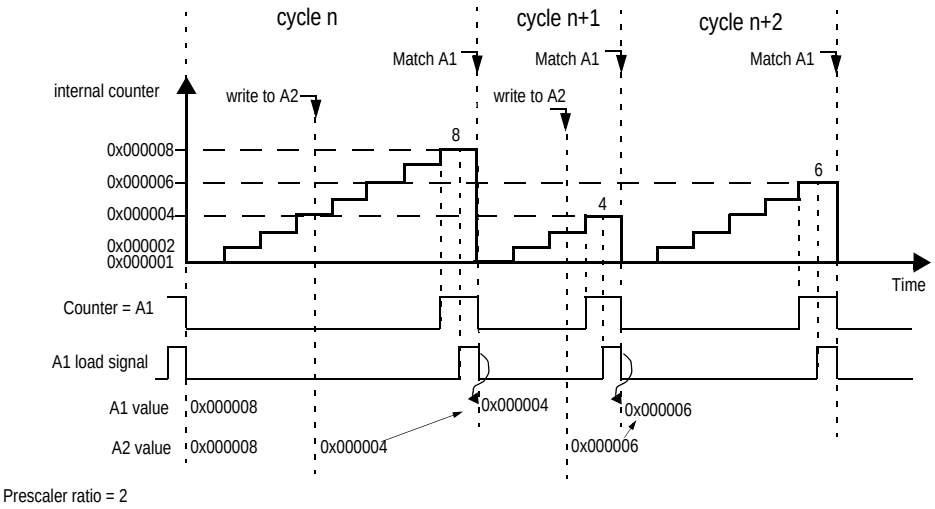


Figure 9-27. MCB mode A1 register update in up counter mode

Figure 9-28 shows the A1 register update in up/down counter mode. Note that A2 can be written at any time within cycle *n* in order to be used in cycle *n* + 1. Thus, A1 receives this new value at the next cycle boundary. Note that the update disable bits OU[*n*] of the EMIOSOUDIS register can be used to disable update of the A1 register.

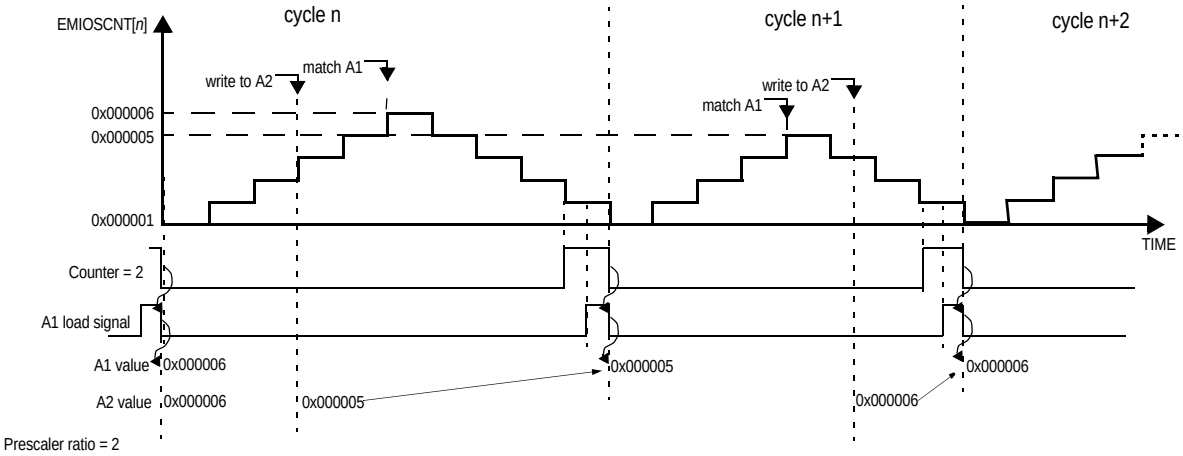


Figure 9-28. MCB mode A1 register update in up/down counter mode

9.5.1.1.5 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode

This mode (MODE[0:6]=10110b0) provides waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base when this mode is selected. The A1 register indicates the duty cycle and the B1 register the frequency. Both the A1 and B1 registers are double-buffered to allow smooth signal generation when changing the register values on the fly. 0% and 100% duty cycles are supported.

At OPWFMB mode entry, the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[*n*] register.

If when entering OPWFMB mode coming out of GPIO mode the internal counter value is not within the specified range, then the B match will not occur. This will cause the channel internal counter to wrap at the maximum counter value, which is 0xFFFF for a 16-bit counter. After the counter wrap occurs, it returns to 0x1 and resumes normal OPWFMB mode operation. Thus in order to avoid the counter wrap condition, make sure its value is within the 0x1 to B1 register value range when OPWFMB mode is entered.

When a match on comparator A occurs the output register is set to the value of EDPOL. When a match on comparator B occurs the output register is set to the complement of EDPOL. B1 match also causes the internal counter to transition to 0x1, thus restarting the counter cycle.

Only values greater than 0x1 are allowed to be written to the B1 register. Loading any other values leads to unpredictable results.

Figure 9-29 shows the operation of OPWFMB mode regarding output pin transitions and A1/B1 register match events. Note that the output pin transition occurs when the A1 or B1 match signal is deasserted, which is indicated by the A1 match negative edge detection signal. If register A1 is set to 0x4, the output pin transitions four counter periods after the cycle started, plus one system clock cycle. Note that in the example shown in Figure 9-29, the internal counter prescaler has a ratio of two.

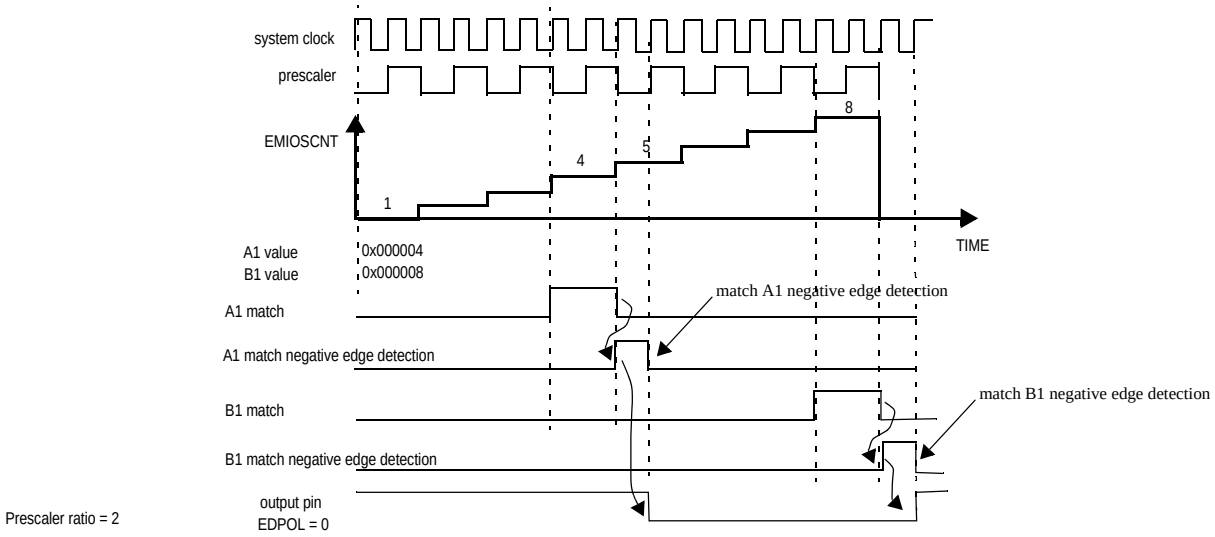


Figure 9-29. OPWFMB A1 and B1 match to Output Register delay

Figure 9-30 shows the generated output signal if A1 is set to 0x0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1 = 0x1, with the difference that in this case, the positive edge of the match signal is used to trigger the output pin transition instead of the negative edge used when A1 = 0x1. Note that the A1 positive edge match signal from cycle *n* + 1 occurs at the same time as B1 negative edge match signal from cycle *n*. This allows using the A1 positive edge match to mask the B1 negative edge match when they occur at the same time. The result is that no transition occurs on the output flip-flop and a 0% duty cycle is generated.

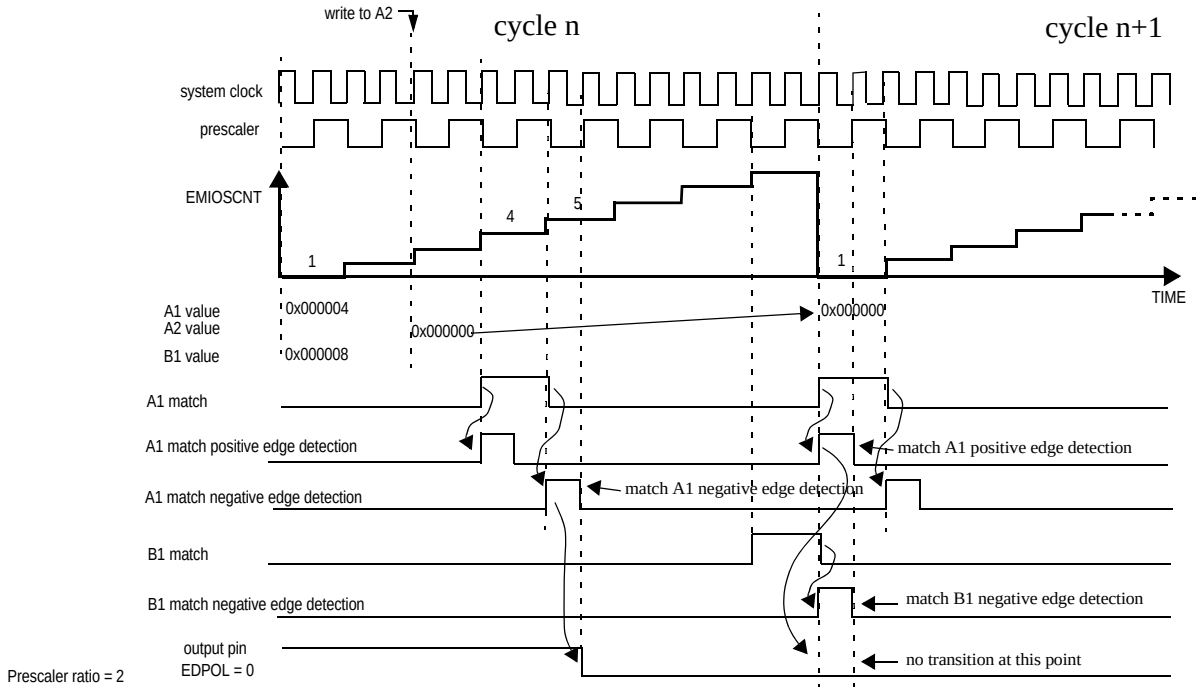


Figure 9-30. OPWFMB mode with A1 = 0 (0% duty cycle)

Figure 9-31 describes the timing for loading the A1 and B1 registers. The A1 and B1 loads use the same signal, which is generated at the last system clock period of a counter cycle. Thus, A1 and B1 are updated with A2 and B2 values, respectively, at the same time that the counter (EMIOSCNT[*n*]) is loaded with 0x1. This event is defined as the cycle boundary. The load signal pulse has a duration of one system clock period. If A2 and B2 are written within cycle *n*, their values are available at A1 and B1, respectively, at the first clock of cycle *n* + 1, and the new values are used for matches at cycle *n* + 1. The update disable bits OU[*n*] of the EMIOSOUDIS register can be used to control the update of these registers, thus allowing delay of the A1 and B1 register updates for synchronization purposes.

In Figure 9-31 it is assumed that both the channel and global prescalers are set to 0x1 (each divide ratio is two), meaning that the channel internal counter transitions at every four system clock cycles. FLAGs can be generated only on B1 matches when MODE[5] is cleared, or on both A1 and B1 matches when MODE[5] is set. Since the B1 flag occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle *n* were loaded to A1 or B1, respectively, thus generating matches in cycle *n* + 1. Note that the FLAG has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

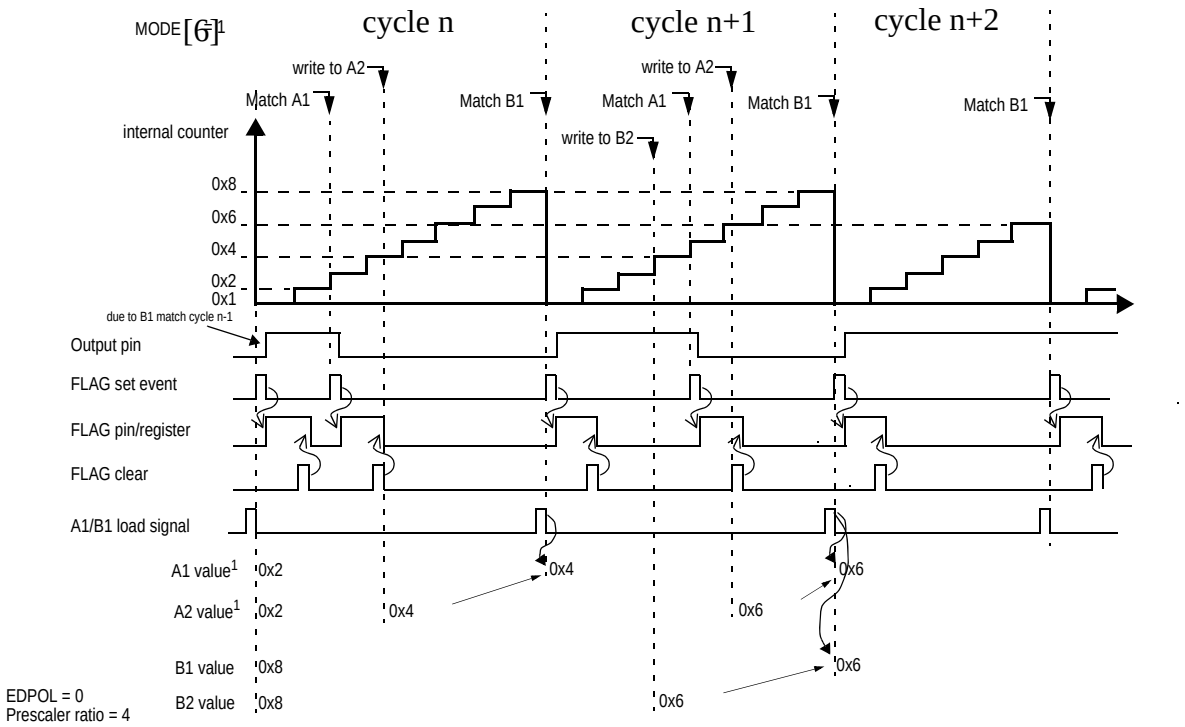


Figure 9-31. OPWFMB A1 and B1 Register update and flags

Figure 9-32 shows the operation of the Output Disable feature in OPWFMB mode. The output disable forces the channel output flip-flop to the value of the EDPOL bit. This functionality targets applications that use active high signals and a high-to-low transition at A1 match. In this case, EDPOL should be set to 0. Note that both the channel and global prescalers are set to 0x0 (each divide ratio is one), meaning that the channel internal counter transitions at every system clock cycle.

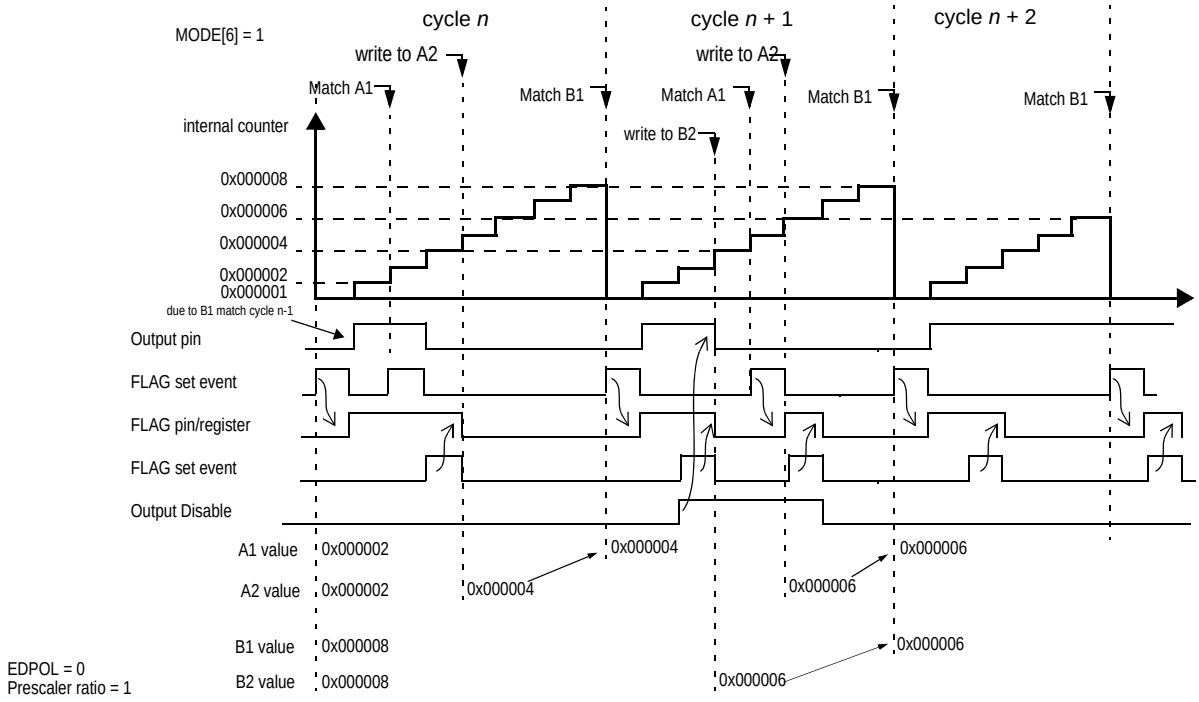


Figure 9-32. OPWFMB mode with active output disable

Note that the output disable has a synchronous operation, meaning that the assertion of the Output Disable input pin causes the channel output flip-flop to transition to EDPOL at the next system clock cycle. If the Output Disable input is deasserted, the output pin transition occurs at the next A1 or B1 match.

In Figure 9-32 it is assumed that the Output Disable input is enabled and selected for the channel. See Section 9.4.2.8, eMIOS200 UC Control Register (EMIOSC[n]), for a detailed description of the ODIS and ODISSL bits, which respectively control enablement and selection of the Output Disable inputs.

The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B, respectively. Similarly, a B1 match on FORCMB sets the internal counter to 0x1. The FLAG bit is not set by the FORCMA or FORCMB bits being asserted.

Figure 9-33 shows the generation of 100% and 0% duty cycle signals. It is assumed that EDPOL = 0 and the resultant prescaler value is 1. Initially, A1 = 0x8 and B1 = 0x8. In this case, B1 match has precedence over A1 match; thus the output flip-flop is set to the complement of the EDPOL bit. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater than or equal to B1.

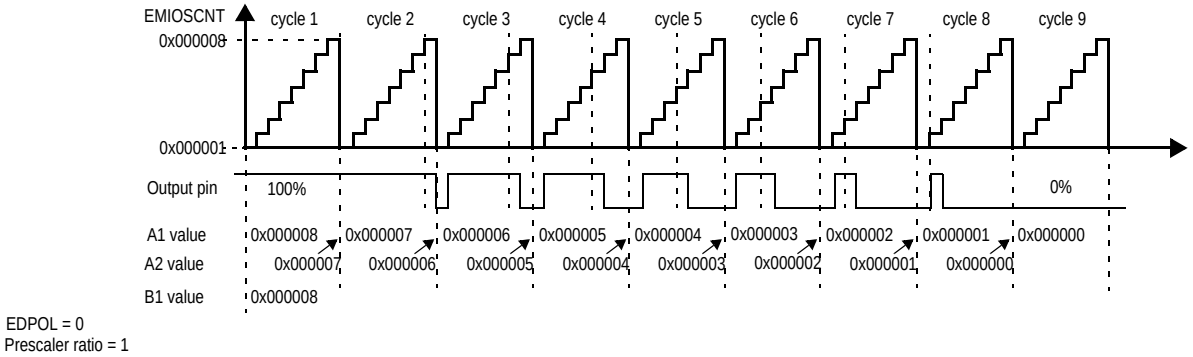


Figure 9-33. OPWFMB mode from 100% to 0% duty cycle

A 0% duty cycle signal is generated if A1 = 0x0, as shown in Figure 9-33, cycle 9. In this case, a B1 = 0x8 match from cycle 8 occurs at the same time as the A1 = 0x0 match from cycle 9. See Figure 9-30 for a description of the A1 and B1 match generation. In this case, the A1 match has precedence over the B1 match and the output signal transitions to EDPOL.

9.5.1.1.6 Output Pulse Width Modulation Buffered (OPWMB) mode

OPWMB mode (MODE[0:6]=11000b0) is used to generate pulses with programmable leading and trailing edge placement. An external counter driven in MCB Up mode must be selected from one of the counter buses. The A1 register value defines the first edge and the B1 value defines the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus; and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double-buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. See Figure 9-31 for more information about A1 and B1 register updating.

The FLAG bit can be generated at B1 matches when MODE[5] is cleared, or in both A1 and B1 matches when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A1 or B1, respectively. The FLAG bit is not set by the FORCMA and FORCMB operations.

At OPWMB mode entry, the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[*n*] register.

Here are some rules applicable to OPWMB mode:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle.
- A1 = 0 match from cycle *n* has precedence over B1 match from cycle *n* – 1.
- A1 matches are masked out if they occur after B1 match within the same cycle.
- Any value written to A2 or B2 on cycle *n* is loaded to the A1 and B1 registers at the following cycle boundary (assuming the OU[*n*] bit of EMIOSOUDIS register is not asserted). Thus, the new values will be used for A1 and B1 matches in cycle *n* + 1.

Figure 9-34 shows the operation of the OPWMB mode regarding A1 and B1 matches and the transition of the channel output pin. In this example, EDPOL is set to zero.

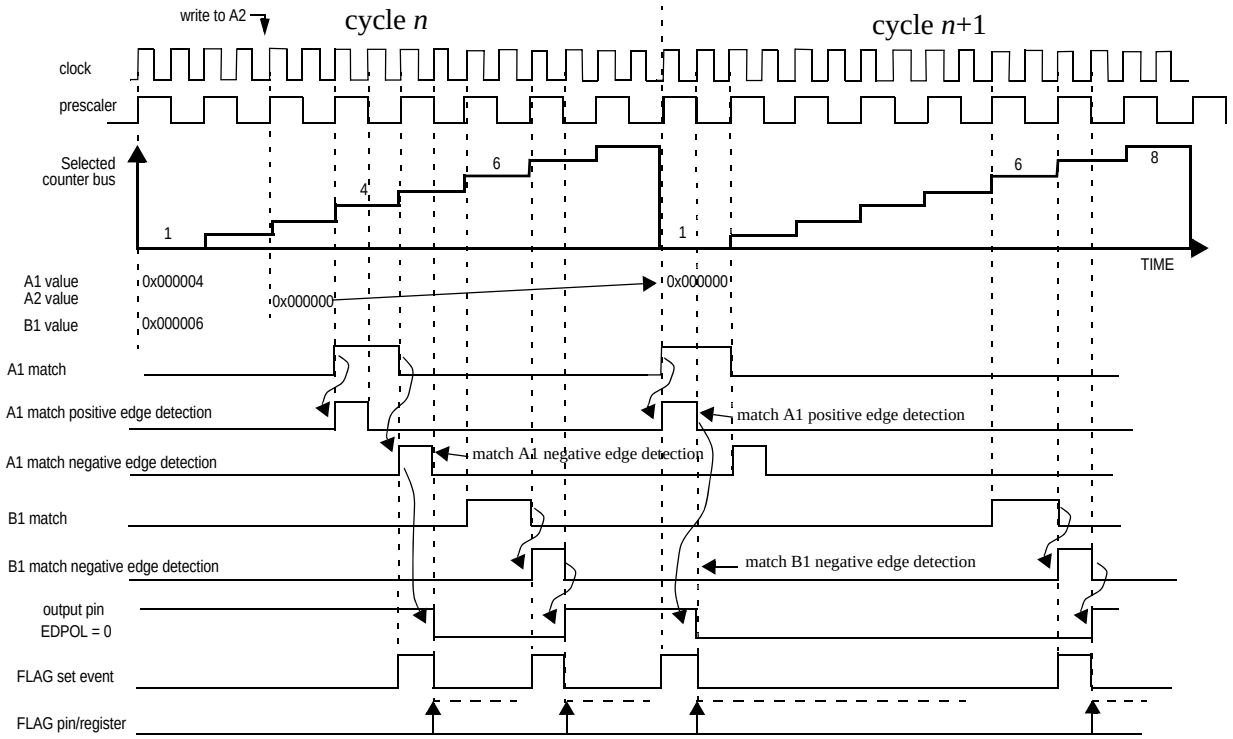


Figure 9-34. OPWMB mode matches and flags

Note that the output pin transitions are based on the negative edges of the A1 and B1 match signals. Figure 9-34 shows in cycle *n* + 1 the A1 register being set to zero. In this case, the match positive edge is used instead of the negative edge to transition the output flip-flop.

Figure 9-35 shows the channel operation for 0% duty cycle. Note that the A1 match positive edge signal occurs at the same time as the B1 = 0x8 negative edge signal. In this case, the A1 match has precedence over the B1 match, causing the output pin to remain at the EDPOL bit value, thus generating a 0% duty cycle signal.

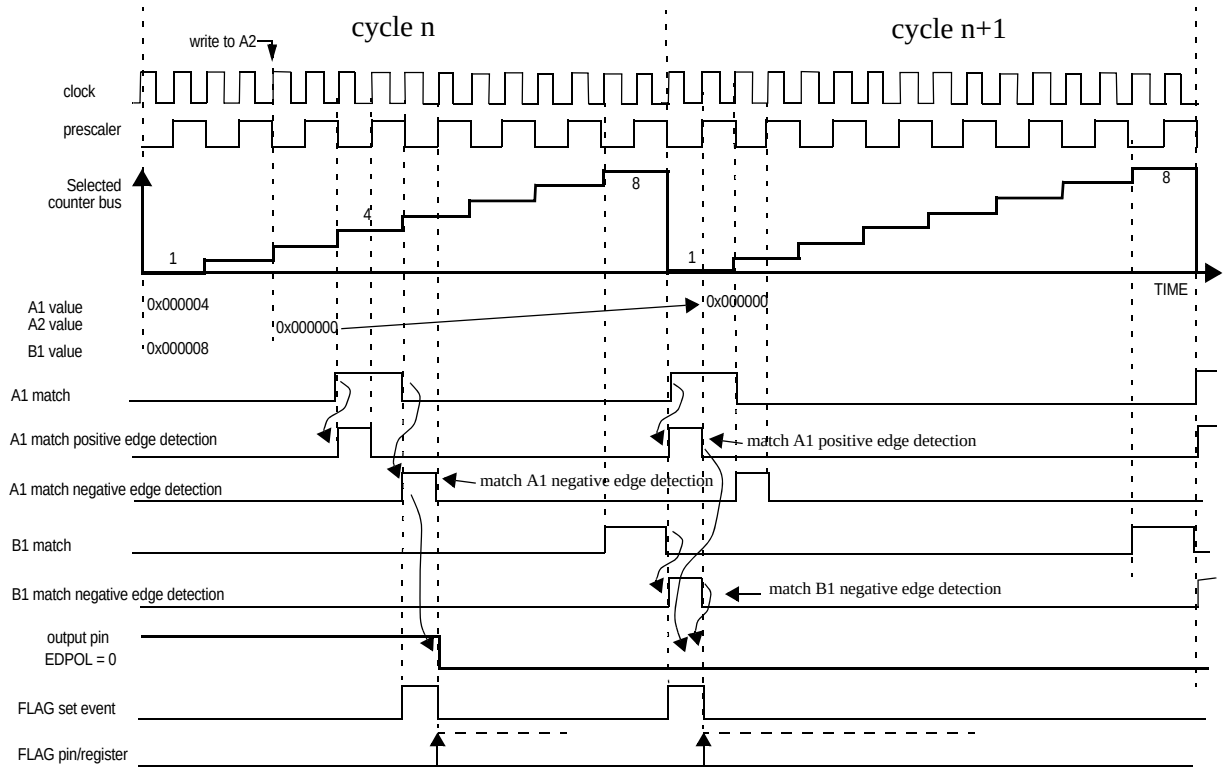


Figure 9-35. OPWMB mode with 0% duty cycle

Figure 9-36 shows the operation of OPWMB mode with the Output Disable signal being asserted. The Output Disable forces a transition in the output pin to the EDPOL bit value. After deassertion, the Output Disable allows the output pin to transition at the next A1 or B1 match. Note that the Output Disable does not modify the FLAG bit behavior. Note that there is one system clock delay between the assertion of the output disable signal and the transition of the output pin to EDPOL.

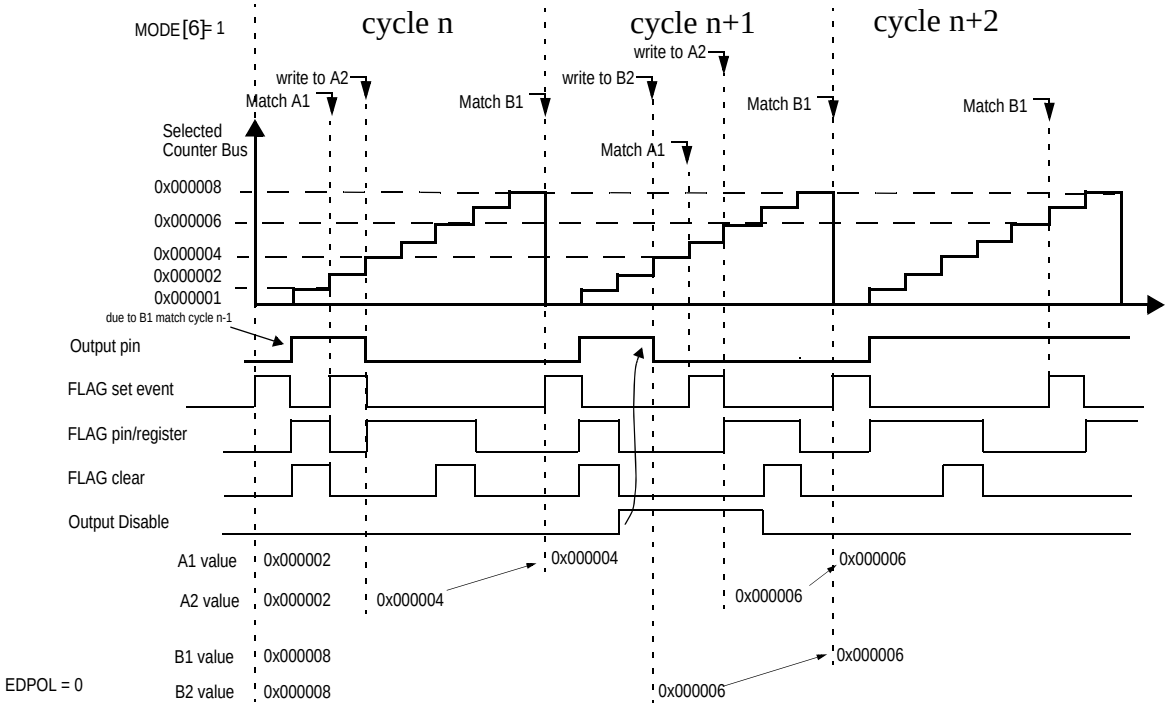


Figure 9-36. OPWMB mode with active output disable

Figure 9-37 shows a waveform changing from 100% to 0% duty cycle. In this case, EDPOL is zero. In this example, B1 is programmed to the same value as the period of the external selected time base.

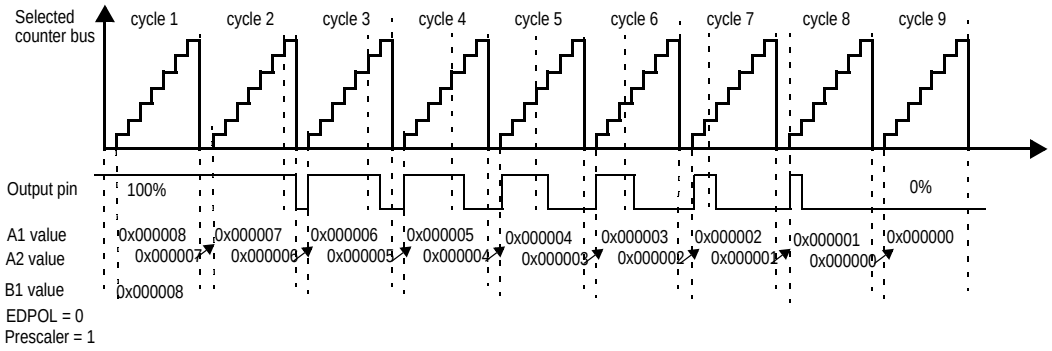


Figure 9-37. OPWMB mode from 100% to 0% duty cycle

In Figure 9-37, if B1 is set to a value lower than 0x8, it is not possible to achieve 0% duty cycle by changing only the A1 register value. Since B1 matches have precedence over A1 matches, the output pin transitions to the opposite of the EDPOL bit at B1 match. Note also that if for instance B1 is set to 0x9, B1 match does not occur; thus a 0% duty cycle signal is generated.

9.5.1.2 Input Programmable Filter (IPF)

The IPF ensures that only valid input pin transitions are received by the Unified Channel edge detector. A block diagram of the IPF is shown in Figure 9-38.

The IPF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[0:3] in the EMIOSC[*n*] register.

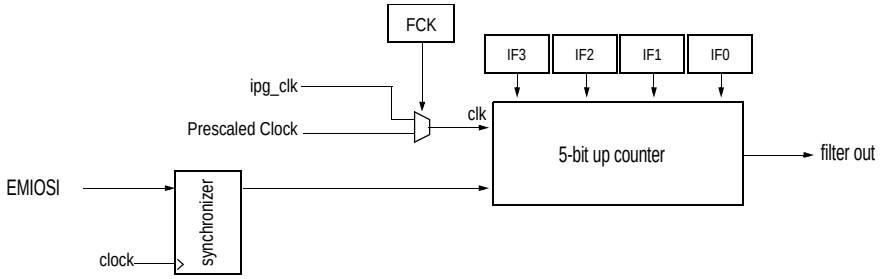


Figure 9-38. IPF submodule diagram

The input signal is synchronized by the system clock. When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter continues incrementing. If a counter overflows occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. A timing diagram of the input filter is shown in Figure 9-39.

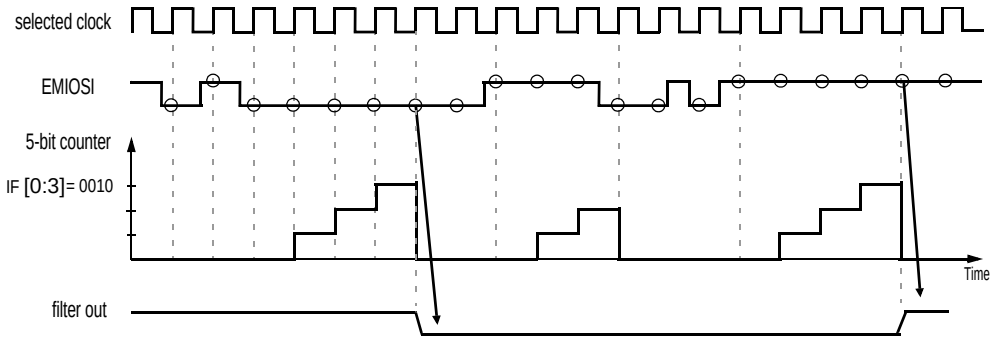


Figure 9-39. IPF example

The filter is not disabled during freeze state.

9.5.1.3 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock enable for the internal counter of the Unified Channels. The GCP output signal is prescaled by the value defined in Figure 9-17 according to the UCPRE[0:1] bits in the EMIOSC[*n*] register. The prescaler is enabled by setting the UCPREN bit in EMIOSC[*n*], and can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel.

In order to ensure safe operation and avoid glitches, the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at both the GPREN bit in the EMIOSMCR register and the UCPREN bit in the EMIOSC[*n*] register, thus disabling prescalers.

2. Write the desired value for prescaling rate at the UCPRE[0:1] bits in the EMIOSC[*n*] register.
3. Enable the channel prescaler by writing 1 at the UCPREN bit in the EMIOSC[*n*] register.
4. Enable the global prescaler by writing 1 at the GPREN bit in the EMIOSMCR register.

The prescaler is not disabled during freeze state.

9.5.1.4 Effect of Freeze on the Unified Channel

When in debug mode, the FRZ bit in the EMIOSMCR register and the FREN bit in the EMIOSC[*n*] register are both set, and the internal counter and the Unified Channel capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the Unified Channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or when the freeze enable bit is cleared (FRZ in the EMIOSMCR or FREN in the EMIOSC[*n*] register), the channel actions resume, but may be inconsistent until the channel enters GPIO mode again.

9.5.2 IP Bus Interface Unit (BIU)

The BIU provides the interface between the Internal Interface Bus (IIB) and the Peripheral Bus, allowing communication among all submodules and this IP interface.

The BIU allows 8-, 16-, and 32-bit access. They are performed over a 32-bit data bus in a single cycle clock.

9.5.2.1 Effect of Freeze on the BIU

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of BIU is not affected.

9.5.3 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the CPs of the channels. The main clock signal is prescaled by the value defined in Figure 9-10, according to the GPRE[0:7] bits in the EMIOSMCR register. The global prescaler is enabled by setting the GPREN bit in the EMIOSMCR register and can be stopped at any time by clearing this bit, thereby stopping the internal counters in all the channels.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 to the GPREN bit in the EMIOSMCR register, thus disabling global prescaler.
2. Write the desired value for the prescaling rate to the GPRE[0:7] bits in the EMIOSMCR register.
3. Enable global prescaler by writing 1 to the GPREN bit in the EMIOSMCR register.

The prescaler is not disabled during freeze state.

9.5.3.1 Effect of Freeze on the GCP

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of the GCP submodule is not affected. There is no freeze function in this submodule.

9.6 Initialization/application information

On resetting the eMIOS200, the Unified Channels enter GPIO input mode.

9.6.1 Considerations

Before changing an operating mode, the UC must be programmed to GPIO mode and the EMIOSA[*n*] and EMIOSB[*n*] registers must be updated with the correct values for the next operating mode. Then the EMIOSC[*n*] register can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base can be random, that is, matches can occur in random time if the contents of EMIOSA[*n*] or EMIOSB[*n*] were not updated with the correct value before the time base matches the previous contents of EMIOSA[*n*] or EMIOSB[*n*].

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

9.6.2 Application information

Correlated output signals can be generated by all output operation modes. Bits OU[*n*] of the EMIOSOUDIS register can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

It is recommended to drive Output Disable Input signals with the emios_flag_out signals of some UCs running in SAIC mode. When an output disable condition happens, the software interrupt routine must service the output channels before servicing the channels running SAIC. This procedure avoids glitches in the output pins.

9.6.2.1 Time Base Generation

For the OPWFM with internal clock source operation mode, the internal counter rate can be modified by configuring the clock prescaler ratio. Figure 9-40 shows an example of a time base with prescaler ratio equal to one.

NOTE

MCB and OPWFMB modes have different behavior.

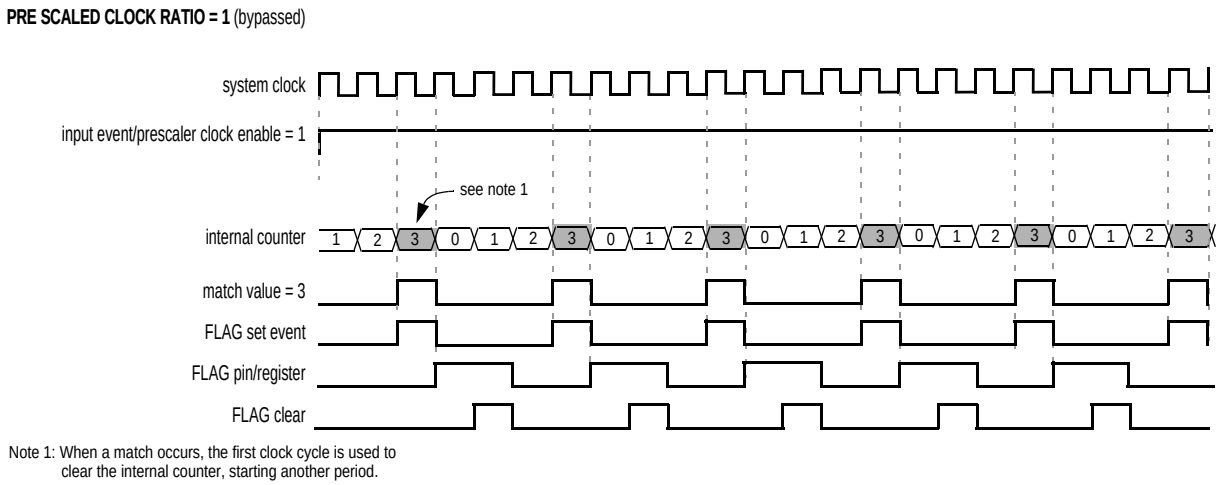


Figure 9-40. Time base period when running in the fastest prescaler ratio

If the prescaler ratio is greater than one or if the external clock is selected, the counter may behave in three different ways, depending on the channel mode:

- If MC mode and Clear on Match Start and External Clock source are selected, the internal counter behaves as shown in Figure 9-41.
- If MC mode and Clear on Match Start and Internal Clock source are selected, the internal counter behaves as shown in Figure 9-42.
- If MC mode and Clear on Match End are selected, the internal counter behaves as shown in Figure 9-43.
- If OPWFM mode is selected, the internal counter behaves as shown in Figure 9-42. The internal counter clears at the start of the match signal, skips the next prescaled clock edge, and then increments on the subsequent prescaled clock edge.

NOTE

MCB and OPWFMB modes have different behavior.

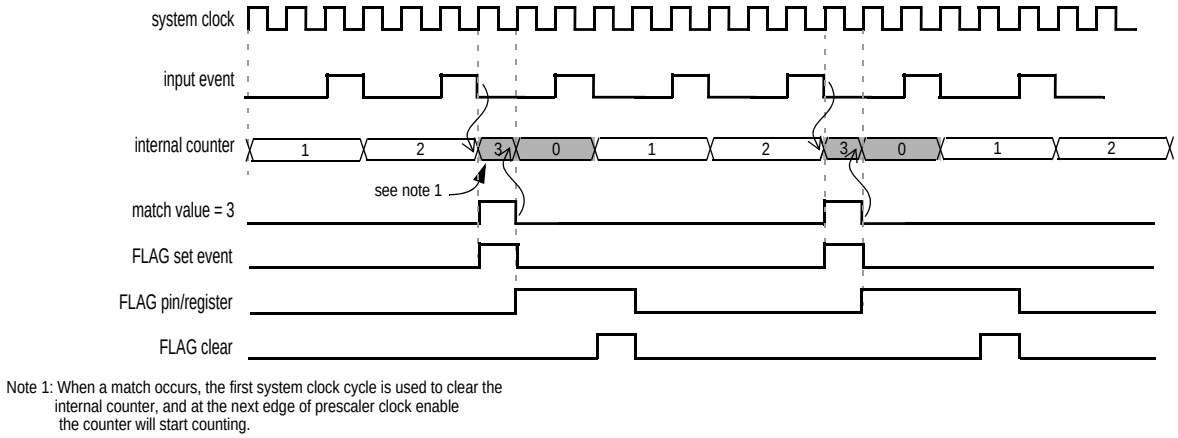
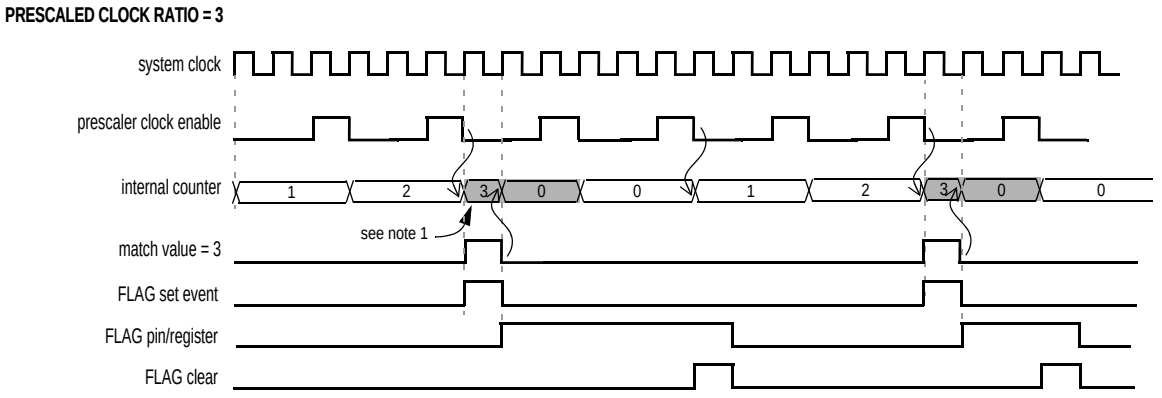
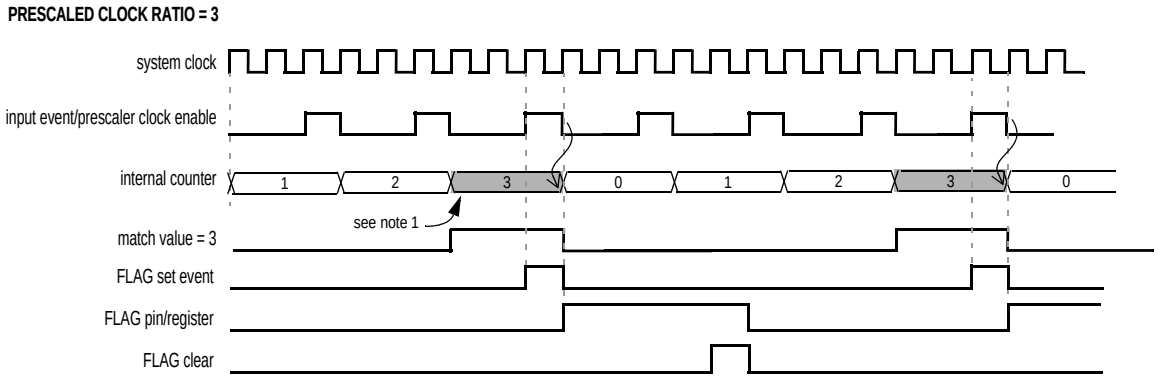


Figure 9-41. Time base generation with external clock and clear on match start



Note 1: When a match occurs, the first clock cycle is used to clear the

Figure 9-42. Time base generation with internal clock and clear on match start



Note 1: The match occurs only when the input event/prescaler clock enable is active.
Then, the internal counter is immediately cleared.

Figure 9-43. Time base generation with clear on match end

9.6.2.2 Coherent accesses

The FLAG set event can be detected by polling the FLAG bit or by enabling the interrupt or DMA request generation.

Reading the EMIOSA[*n*] register again in the same period as the last read of the EMIOSB[*n*] register may lead to incoherent results. This will occur if the last read of the EMIOSB[*n*] register occurred after a disabled B2 to B1 transfer.

9.6.2.3 Channel/modes initialization

The following basic steps summarize basic output mode startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable the Global Prescaler.
2. *[timebase channel]* Disable the Channel Prescaler.
3. *[timebase channel]* Write the initial value to the internal counter.
4. *[timebase channel]* Set A/B register.

5. *[timebase channel]* Set channel to MC(B) Up mode.
6. *[timebase channel]* Set the prescaler ratio.
7. *[timebase channel]* Enable the Channel Prescaler.
8. *[output channel]* Disable the Channel Prescaler.
9. *[output channel]* Set A/B register.
10. *[output channel]* Select the timebase input through the BSL[1:0] bits.
11. *[output channel]* Enter the output mode.
12. *[output channel]* Set the prescaler ratio (same ratio as the timebase channel).
13. *[output channel]* Enable the Channel Prescaler.
14. *[global]* Enable the Global Prescaler.

The timebase channel and the output channel may be the same for some applications such as in OPWFM(B) mode or whenever the output channel is intended to run the timebase itself.

The flags can be configured at any time.

Chapter 10
Crossbar Switch (XBAR)

10.1 Introduction

This chapter describes the multi-port crossbar switch (XBAR), which supports simultaneous connections between four master ports and four slave ports. A port splitter allows three of the MPC5606S slaves to be consolidated on one slave port. XBAR supports a 32-bit address bus width and a 32-bit data bus width at all master and slave ports.

The crossbar of MPC5606S is similar to that used on many PPC55xx and PPC56xx products except that it cannot be configured by software and that it has a hard-wired configuration.

10.2 Block diagram

Figure 10-1 shows a block diagram of the crossbar switch.

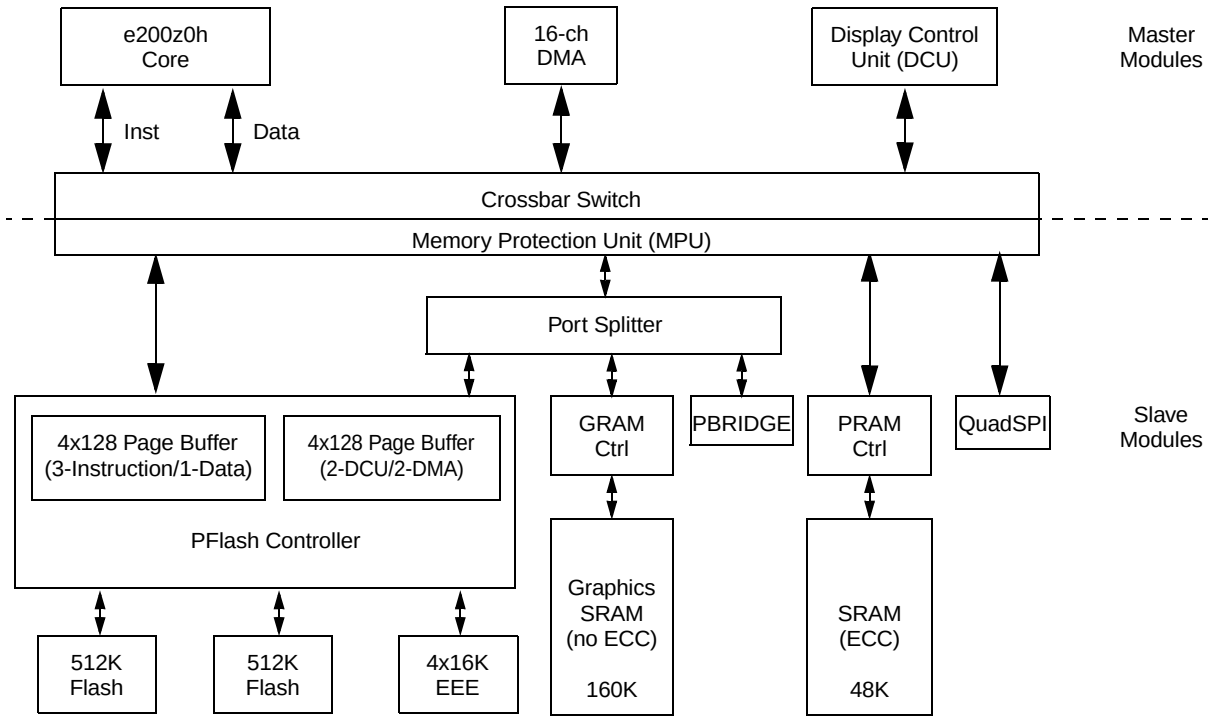


Figure 10-1. XBAR block diagram

10.3 Overview

The XBAR allows for concurrent transactions to occur from any master port to any slave port. It is possible for all master ports and slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic selects the higher

priority master and grants it ownership of the slave port. All other masters requesting that slave port are stalled until the higher priority master completes its transactions.

Requesting masters are granted access based on a fixed priority. A block diagram of the XBAR is shown in Figure 10-1.

10.4 Features

- Four Master ports:
  - core: e200z0h core instructions
  - core: e200z0h core data / Nexus
  - eDMA
  - Display Controller Unit (DCU)
- Six slave ports
  - PFlash-CPU
  - PFlash-DCU
  - Internal SRAM
  - Graphics SRAM
  - Peripheral bridge (PBRIDGE)
  - QuadSPI
- 32-bit address, 32-bit data paths
- Fully concurrent transfers between independent master and slave ports
- Fixed priority scheme and fixed parking strategy

10.5 Modes of operation

10.5.1 Normal mode

In normal mode, the XBAR provides the logic that controls crossbar switch configuration.

10.5.2 Debug mode

The XBAR operation is unchanged when the CPU has debug mode active.

10.6 Functional description

This section describes the functionality of the XBAR in more detail.

10.6.1 Overview

The main goal of the XBAR is to increase overall system performance by allowing multiple masters to communicate concurrently with multiple slaves. To maximize data throughput, it is essential to keep arbitration delays to a minimum.

This section examines data throughput from the point of view of masters and slaves, detailing when the XBAR stalls masters, or inserts bubbles on the slave side.

10.6.2 General operation

When a master makes an access to the XBAR from an idle master state, the access is taken immediately by the XBAR. If the targeted slave port of the access is available (that is, the requesting master is currently granted ownership of the slave port), the access is immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the XBAR by a granted master. If the targeted slave port of the access is busy or parked on a different master port, the requesting master receives wait states until the targeted slave port can service the master request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the XBAR appears to be simply another slave to the master device, the master device has no indication that it owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it is wait-stated.

A master is given control of a targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has the following conditions:

- Outstanding request to slave port A that has a long response time
- Pending access to a different slave port B
- Lower priority master also makes a request to the different slave port B.

In this case, the lower priority master is granted bus ownership of slave port B after a cycle of arbitration, assuming the higher priority master slave port A access is not terminated.

After a master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an Idle cycle, leaves that slave port for its next access, or loses control of the slave port to a higher priority master with a request to the same slave port. However, because all masters run a fixed-length burst transfer to a slave port, it retains control of the slave port until that transfer sequence is completed.

When a slave bus is idled by the XBAR, it is parked on the master which did the last transfer.

10.6.3 Master ports

A master access is taken if the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case, the XBAR is completely transparent and the master access is immediately transmitted on the slave bus and no arbitration delays are incurred. A master access stall if the access decodes to a slave port that is busy serving another master, parked on another master.

If the slave port is currently parked on another master, and no other master is requesting access to the slave port, then only one clock of arbitration is incurred. If the slave port is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters, then the master gains control over the slave port as soon as the data phase of the current access is completed. If the slave port is currently servicing another master of a higher priority, then the master gains control of the slave port after

the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

A master access is responded to with an error if the access decodes to a location not occupied by a slave port. This is the only time the XBAR directly responds with an error response. All other error responses received by the master are the result of error responses on the slave ports being passed through the XBAR.

10.6.4 Slave ports

The goal of the XBAR with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. To do this the XBAR must not insert any bubbles onto the slave bus unless absolutely necessary.

There is only one instance when the XBAR forces a bubble onto the slave bus when a master is actively making a request. This occurs when a handoff of bus ownership occurs and there are no wait states from the slave port. A requesting master which does not own the slave port is granted access after a one clock delay.

10.6.5 Priority assignment

Each master port is assigned a fixed 3-bit priority level (hard-wired priority). The following table shows the priority levels assigned to each master (the lowest has highest priority).

Table 10-1. Hardwired bus master priorities

Module	Port		Priority level
	Type	Number	
e200z0h core–CPU instructions	Master	0	7
e200z0h core–CPU data / Nexus	Master	0	6
eDMA	Master	2	5
Display Control Unit	Master	3	4

10.6.6 Arbitration

XBAR supports only a fixed-priority comparison algorithm.

10.6.6.1 Fixed priority operation

When operating in fixed-priority arbitration mode, each master is assigned a unique priority level in the XBAR_MPR. If two masters both request access to a slave port, the master with the highest priority in the selected priority register gains control over the slave port.

Any time a master makes a request to a slave port, the slave port checks to see if the new requesting master’s priority level is higher than that of the master that currently has control over the slave port (if any). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the higher priority master is granted control at the termination of any currently pending access, assuming the pending transfer is not part of a burst transfer.

A new requesting master must wait until the end of the fixed-length burst transfer, before it is granted control of the slave port. But if the new requesting master's priority level is lower than that of the master that currently has control of the slave port, the new requesting master is forced to wait until the master that currently has control of the slave port is finished accessing the current slave port.

10.6.6.1.1 Parking

If no master is currently requesting the slave port, the slave port is parked. The slave port parks always to the last master (park-on-last). When parked on the last master, the slave port is passing that master's signals through to the slave bus. When the master accesses the slave port again, no other arbitration penalties are incurred except that a one clock arbitration penalty is incurred for each access request to the slave port made by another master port. All other masters pay a one clock penalty.

Chapter 11
Deserial Serial Peripheral Interface (DSPI)

11.1 Introduction

This chapter describes the deserial serial peripheral interface (DSPI), which provides a synchronous serial bus for communication between the MCU and an external peripheral device.

This device implements DSPI 0 and DSPI 1. The “x” appended to signal names signifies the DSPI module to which the signal applies. Thus CS0_0 is the CS0 signal that applies to DSPI 0, CS0_1 is the CS0 signal that applies to DSPI 1.

11.2 Block diagram

A block diagram of the DSPI is shown in Figure 11-1.

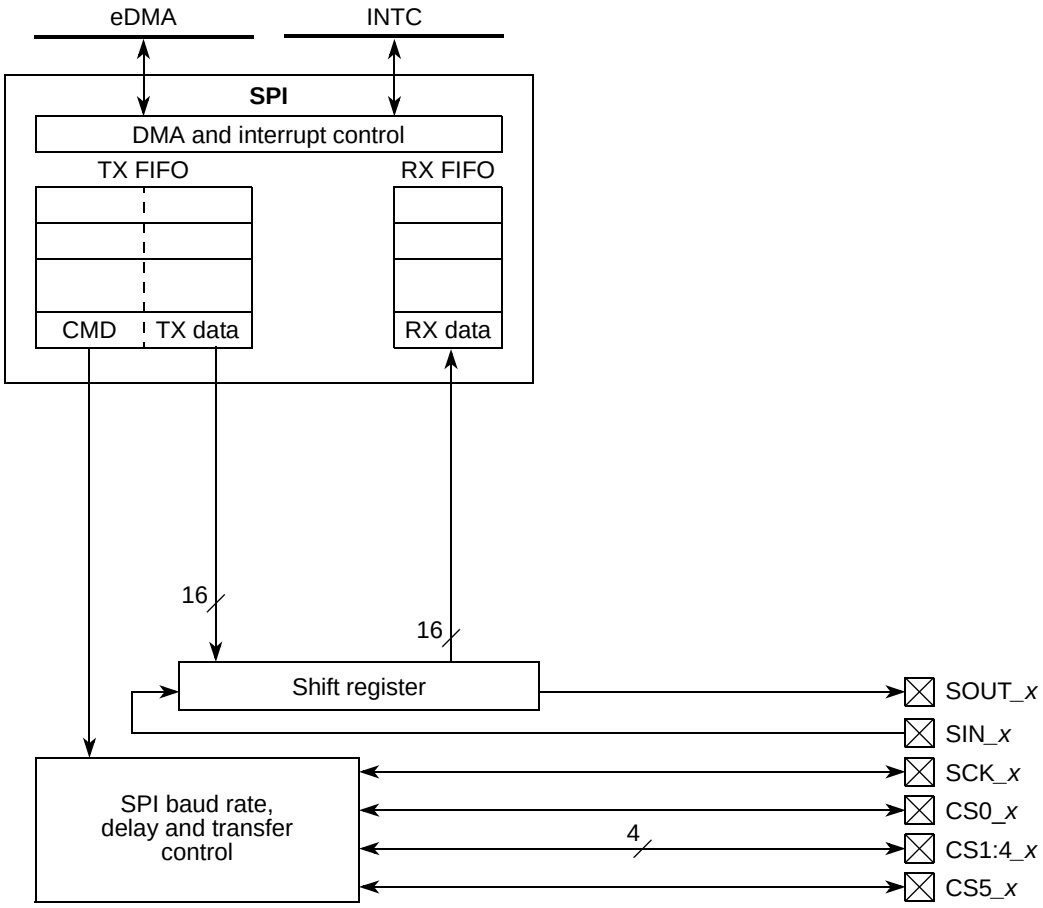


Figure 11-1. DSPI block diagram

11.3 Overview

The register content is transmitted using an SPI protocol. There are two identical DSPI modules (DSPI 0 and DSPI 1) on the device.

For queued operations the SPI queues reside in internal SRAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished through the use of the eDMA controller or through host software.

Figure 11-2 shows a DSPI with external queues in internal SRAM.

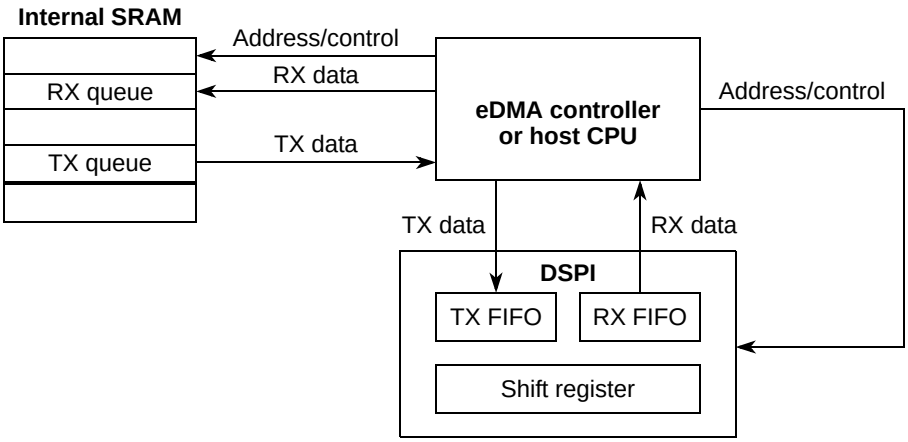


Figure 11-2. DSPI with queues and eDMA

11.4 Features

The DSPI supports these SPI features:

- Full-duplex, three-wire synchronous transfers
- Master and Slave mode
- Buffered transmit and receive operation using the TX and RX FIFOs, with depths of five entries
- Visibility into TX and RX FIFOs for ease of debugging
- FIFO bypass mode for low-latency updates to SPI queues

- Programmable transfer attributes on a per-frame basis
  - 8 clock and transfer attribute registers
  - Serial clock with programmable polarity and phase
  - Programmable delays
    - CS to SCK delay
    - SCK to CS delay
    - Delay between frames
  - Programmable serial frame size of 4 to 16 bits, expandable with software control
  - Continuously held chip select capability
- 3 peripheral chip selects, expandable to 64 with external demultiplexer
- Deglitching support for up to 32 peripheral chip selects with external demultiplexer
- 2 DMA conditions for SPI queues residing in RAM or flash
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
- 6 interrupt conditions:
    - End of queue reached (EOQF)
    - TX FIFO is not full (TFFF)
    - Transfer of current frame complete (TCF)
    - RX FIFO is not empty (RFDF)
    - FIFO overrun (attempt to transmit with an empty TX FIFO or serial frame received while RX FIFO is full) (RFOF)
    - FIFO under flow (slave only and SPI mode, the slave is asked to transfer data when the TX FIFO is empty) (TFUF)
- Modified SPI transfer formats for communication with slower peripheral devices
- Continuous serial communications clock (SCK)

11.5 Modes of operation

The DSPI has five modes of operation. These modes can be divided into two categories:

- Module-specific modes such as master, slave, and Module Disable modes
- MCU-specific modes such as external stop and debug modes

The module-specific modes are entered by host software writing to a register. The MCU-specific modes are controlled by signals external to the DSPI. The MCU-specific modes are modes that the entire device may enter, in parallel to the DSPI being in one of its module-specific modes.

11.5.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode, the SCK and CS*n* signals are controlled by the DSPI and configured as outputs. (SOUT is always an output.)

For more information, refer to Section 11.8.1.1, Master mode.

11.5.2 Slave mode

Slave mode allows the DSPI to communicate with SPI bus masters. In this mode the DSPI responds to externally controlled serial transfers. The DSPI cannot initiate serial transfers in Slave mode. In Slave mode, the SCK signal and the CS0_*x* signal are configured as inputs and provided by a bus master. CS0_*x* must be configured as input and pulled high. If the internal pullup is being used, then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE = 1], [WPS = 1]).

For more information, refer to Section 11.8.1.2, Slave mode.

11.5.3 Module Disable mode

The Module Disable mode is used for MCU power management. The clock to the non-memory mapped logic in the DSPI is stopped while in Module Disable mode. The DSPI enters the Module Disable mode when the MDIS bit in DSPI*x*_MCR is set.

For more information, refer to Section 11.8.1.3, Module Disable mode.

11.5.4 External Stop mode

The External Stop mode is used for MCU power management. The DSPI supports the IPI Green-Line Interface Stop mode mechanism. When a request is made to enter External Stop mode, the DSPI block acknowledges the request and completes the transfer in progress. When the DSPI reaches the frame boundary it signals that the system clocks to the DSPI block may be shut off.

11.5.5 Debug mode

Debug mode is used for system development and debugging. If the MCU is stopped by a debugger while the DSPI*x*_MCR[FRZ] bit is set, the DSPI halts operation on the next frame boundary. If the MCU is stopped by a debugger while the FRZ bit is cleared, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI.

For more information, refer to Section 11.8.1.5, Debug mode.

11.6 External signal description

11.6.1 Signal overview

Table 11-1 lists off-chip DSPI signals.

Table 11-1. Signal properties

Name	I/O Type	Function	
		Master mode	Slave mode
CS0_*x*	Output / input	Peripheral chip select 0	Slave select
CS1:2_x	Output	Peripheral chip select 1–2	Unused[1]
SIN_x	Input	Serial data in	Serial data in
SOUT_x	Output	Serial data out	Serial data out
SCK_x	Output / input	Serial clock (output)	Serial clock (input)

1 The SIU allows you to select alternate pin functions for the device.

11.6.2 Signal names and descriptions

11.6.2.1 Peripheral Chip Select / Slave Select (CS_0)

In Master mode, the CS_0 signal is a peripheral chip select output that selects the slave device to which the current transmission is intended.

In Slave mode, the CS_0 signal is a slave select input signal that allows an SPI master to select the DSPI as the target for transmission. CS_0 must be configured as input and pulled high. If the internal pullup is being used, then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE = 1], [WPS = 1]).

Set the IBE and OBE bits in the SIU_PCR for all CS_0 pins when the DSPI chip select or slave select primary function is selected for that pin. When the pin is used for DSPI Master mode as a chip select output, set the OBE bit. When the pin is used in DSPI Slave mode as a slave select input, set the IBE bit.

11.6.2.2 Peripheral Chip Selects 1–2 (CS1:2)

CS1:2 are peripheral chip select output signals in Master mode. In Slave mode these signals are not used.

11.6.2.3 Serial Input (SIN_*x*)

SIN_*x* is a serial data input signal.

11.6.2.4 Serial Output (SOUT_*x*)

SOUT_*x* is a serial data output signal.

11.6.2.5 Serial Clock (SCK_*x*)

SCK_*x* is a serial communication clock signal. In Master mode, the DSPI generates the SCK. In Slave mode, SCK_*x* is an input from an external bus master.

11.7 Memory map and register description

11.7.1 Memory map

Table 11-2 shows the DSPI memory map.

Table 11-2. DSPI detailed memory map

Address	Register name	Register description	Location
Base: 0xFFF9_0000 (DSPI 0) 0xFFF9_4000 (DSPI 1)	DSPI*x*_MCR	DSPI module configuration register	on page 291
Base + 0x0004	—	Reserved	—
Base + 0x0008	DSPI*x*_TCR	DSPI Transfer Count Register	on page 293
Base + 0x000C	DSPI*x*_CTAR0	DSPI Clock and Transfer Attributes Register 0	on page 293
Base + 0x0010	DSPI*x*_CTAR1	DSPI Clock and Transfer Attributes Register 1	on page 293
Base + 0x0014	DSPI*x*_CTAR2	DSPI Clock and Transfer Attributes Register 2	on page 293
Base + 0x0018	DSPI*x*_CTAR3	DSPI Clock and Transfer Attributes Register 3	on page 293
Base + 0x001C	DSPI*x*_CTAR4	DSPI Clock and Transfer Attributes Register 4	on page 293
Base + 0x0020	DSPI*x*_CTAR5	DSPI Clock and Transfer Attributes Register 5	on page 293
Base + 0x0024	DSPI*x*_CTAR6	DSPI Clock and Transfer Attributes Register 6	on page 293
Base + 0x0028	DSPI*x*_CTAR7	DSPI Clock and Transfer Attributes Register 7	on page 293
Base + 0x002C	DSPI*x*_SR	DSPI Status Register	on page 299
Base + 0x0030	DSPI*x*_RSER	DSPI DMA/Interrupt Request Select and Enable Register	on page 301
Base + 0x0034	DSPI*x*_PUSHR	DSPI Push TX FIFO Register	on page 303
Base + 0x0038	DSPI*x*_POPR	DSPI Pop RX FIFO Register	on page 305
Base + 0x003C	DSPI*x*_TXFR0	DSPI Transmit FIFO Register 0	on page 305
Base + 0x0040	DSPI*x*_TXFR1	DSPI Transmit FIFO Register 1	on page 305
Base + 0x0044	DSPI*x*_TXFR2	DSPI Transmit FIFO Register 2	on page 305
Base + 0x0048	DSPI*x*_TXFR3	DSPI Transmit FIFO Register 3	on page 305
Base + 0x004C– Base + 0x0078	—	Reserved	—
Base + 0x007C	DSPI*x*_RXFR0	DSPI Receive FIFO Register 0	on page 306
Base + 0x0080	DSPI*x*_RXFR1	DSPI Receive FIFO Register 1	on page 306
Base + 0x0084	DSPI*x*_RXFR2	DSPI Receive FIFO Register 2	on page 306
Base + 0x0088	DSPI*x*_RXFR3	DSPI Receive FIFO Register 3	on page 306
Base + 0x008C– Base + 0x00CC	—	Reserved	—

11.7.2 Register description

11.7.2.1 DSPI Module Configuration Register (DSPI*x*_MCR)

The DSPI*x*_MCR contains bits which configure attributes of the DSPI operation. The values of the HALT and MDIS bits can be changed at any time, but their effect begins on the next frame boundary. The HALT and MDIS bits in the DSPI*x*_MCR are the only bit values that software can change while the DSPI is running.

Address: Base + 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	0	ROOE	0	0	PCSIS5	PCSIS4	PCSIS3	PCSIS2	PCSIS1	PCSIS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MDIS	DIS_TXF	DIS_RXF	CLR_TXF	CLR_RXF	SMPL_PT		0	0	0	0	0	0	PES	HALT
W					w1c	w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 11-3. DSPI Module Configuration Register (DSPI*x*_MCR)

Table 11-3 describes the fields in the DSPI module configuration register.

Table 11-3. DSPI*x*_MCR field descriptions

Field	Description
0 MSTR	Master/Slave mode select. Configures the DSPI for Master mode or Slave mode. 0 DSPI is in Slave mode 1 DSPI is in Master mode
1 CONT_SCKE	Continuous SCK enable. Enables the serial communication clock (SCK) to run continuously. Refer to Section 11.8.6, Continuous serial communications clock, for details. 0 Continuous SCK disabled 1 Continuous SCK enabled
2–3 DCONF [0:1]	DSPI Configuration. The DCONF field selects between the different configurations of the DSPI. 00 SPI. 01 Reserved 10 Reserved 11 Reserved
4 FRZ	Freeze. Enables the DSPI transfers to be stopped on the next frame boundary when the device enters debug mode. 0 Do not halt serial transfers 1 Halt serial transfers
5 MTFE	Modified timing format enable. Enables a modified transfer format to be used. Refer to Section 11.8.5.4, Modified SPI transfer format (MTFE = 1, CPHA = 1), for more information. 0 Modified SPI transfer format disabled 1 Modified SPI transfer format enabled

Table 11-3. DSPI*x*_MCR field descriptions (continued)

Field	Description
6	Reserved. This bit is writable, but has no effect.
7 ROOE	Receive FIFO overflow overwrite enable. Enables an RX FIFO overflow condition to ignore the incoming serial data or to overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer that generated the overflow is ignored or put in the shift register. If the ROOE bit is set, the incoming data is put in the shift register. If the ROOE bit is cleared, the incoming data is ignored. Refer to Section 11.8.7.6, Receive FIFO overflow interrupt request (RFOF), for more information. 0 Incoming data is ignored 1 Incoming data is put in the shift register
8–9	Reserved, but implemented. These bits are writable, but have no effect.
10–15 PCSIS*n*	Peripheral chip select inactive state. Determines the inactive state of the CS0_*x* signal. CS0_*x* must be configured as inactive high for Slave mode operation. 0 The inactive state of CS0_*x* is low 1 The inactive state of CS0_*x* is high
16	Reserved.
17 MDIS	Module disable. Allows the clock to the non-memory mapped logic in the DSPI to stop, effectively putting the DSPI in a software-controlled power-saving state. Refer to Section 11.8.8, Power-saving features, for more information. The reset value of the MDIS bit is parameterized, with a default reset value of 0. 0 Enable DSPI clocks 1 Allow external logic to disable DSPI clocks
18 DIS_TXF	Disable transmit FIFO. Enables and disables the TX FIFO. When the TX FIFO is disabled, the transmit part of the DSPI operates as a simplified double-buffered SPI. Refer to Section 11.8.3.3, FIFO disable operation, for details. 0 TX FIFO is enabled 1 TX FIFO is disabled
19 DIS_RXF	Disable receive FIFO. Enables and disables the RX FIFO. When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. Refer to Section 11.8.3.3, FIFO disable operation, for details. 0 RX FIFO is enabled 1 RX FIFO is disabled
20 CLR_TXF	Clear TX FIFO. Flushes the TX FIFO. Write a 1 to the CLR_TXF bit to clear the TX FIFO counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter 1 Clear the TX FIFO counter
21 CLR_RXF	Clear RX FIFO. Flushes the RX FIFO. Write a 1 to the CLR_RXF bit to clear the RX counter. The CLR_RXF bit is always read as zero. 0 Do not clear the RX FIFO counter 1 Clear the RX FIFO counter
22–23 SMPL_ PT [0:1]	Sample Point. SMPL_PT field controls when the DSPI master samples SIN in Modified Transfer Format. Figure 11-14 shows where the master can sample the SIN pin. 00 DSPI samples SIN at driving SCK edge. 01 DSPI samples SIN one system clock after driving SCK edge. 10 DSPI samples SIN two system clocks after driving SCK edge. 11 Reserved.

Table 11-3. DSPI*x*_MCR field descriptions (continued)

Field	Description
24–30	Reserved.
31 HALT	Halt. Provides a mechanism for software to start and stop DSPI transfers. Refer to Section 11.8.2, Start and stop of DSPI transfers, for details on the operation of this bit. 0 Start transfers 1 Stop transfers

11.7.2.2 DSPI Transfer Count Register (DSPI*x*_TCR)

The DSPI*x*_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. The user must not write to the DSPI*x*_TCR while the DSPI is running.

Address: Base + 0x0008 Access:User read/write

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

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

Figure 11-4. DSPI Transfer Count Register (DSPI*x*_TCR)

Table 11-4 describes the field in the DSPI transfer count register.

Table 11-4. DSPI*x*_TCR field descriptions

Field	Description
0–15 SPI_TCNT [0:15]	SPI transfer counter. Counts the number of SPI transfers the DSPI makes. The SPI_TCNT field is incremented every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The transfer counter wraps around—in other words, incrementing the counter past 65535 resets the counter to zero.
16–31	Reserved.

11.7.2.3 DSPI Clock and Transfer Attributes Registers 0–7 (DSPI*x*_CTAR*n*)

The DSPI modules each contain eight clock and transfer attribute registers (DSPI*x*_CTAR*n*) which are used to define different transfer attribute configurations. Each DSPI*x*_CTAR controls:

- Frame size
- Baud rate and transfer delay values

- Clock phase
- Clock polarity
- MSB or LSB first

At the initiation of an SPI transfer, control logic selects the DSPI*x*_CTAR that contains the transfer's attributes.Do not write to the DSPI*x*_CTARs while the DSPI is running.

In Master mode, the DSPI*x*_CTAR*n* registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In Slave mode, a subset of the bit fields in the DSPI*x*_CTAR0 and DSPI*x*_CTAR1 registers are used to set the slave transfer attributes. Refer to the individual bit descriptions for details on which bits are used in Slave modes.

When the DSPI is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the DSPI*x*_CTAR registers is used on a per-frame basis. When the DSPI is configured as an SPI bus slave, the DSPI*x*_CTAR0 register is used.

Address:
Base + 0x000C (DSPI*x*_CTAR0) Base + 0x001C (DSPIx_CTAR4)
Base + 0x0010 (DSPI*x*_CTAR1) Base + 0x0020 (DSPI*x*_CTAR5)
Base + 0x0014 (DSPI*x*_CTAR2) Base + 0x0024 (DSPI*x*_CTAR6)
Base + 0x0018 (DSPI*x*_CTAR3) Base + 0x0028 (DSPI*x*_CTAR7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-5. DSPI Clock and Transfer Attributes Registers 0–7 (DSPI*x*_CTARn)

Table 11-5. DSPI*x*_CTAR*n* field descriptions

Field	Descriptions
0 DBR	Double Baud Rate. The DBR bit doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in Master mode. It effectively halves the baud rate division ratio supporting faster frequencies and odd division ratios for the SCK. When the DBR bit is set, the duty cycle of the SCK depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in Table 11-6. See the BR field description for details on how to compute the baud rate. If the overall baud rate is the system clock divided by two or divided by three, then neither the Continuous SCK Enable nor the Modified Timing Format Enable bits should be set. 0 The baud rate is computed normally with a 50/50 duty cycle 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler

Table 11-5. DSPI*x*_CTAR*n* field descriptions (continued)

Field	Descriptions
1–4 FMSZ[0:3]	Frame Size. The FMSZ field selects the number of bits transferred per frame. The FMSZ field is used in Master mode and Slave mode. Table 11-7 lists the frame size encodings. When operating in TSB confirmation, the FMSZ defines the point within the 32-bit (maximum length) frame where control of the CS switches from the DSPI_DSICR to the DSPI_DSICR1 register. The crossover point must range between 4 bits and 16 bits and is encoded as indicated in Table 11-7. The remaining frame after the crossover point, regardless of how many bits are remaining, will be controlled by the DSPI_DSICR1 register.
5 CPOL	Clock Polarity. The CPOL bit selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both Master and Slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the continuous selection format (see Section 11.8.5.5, Continuous selection format) is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low 1 The inactive state value of SCK is high
6 CPHA	Clock Phase. The CPHA bit selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both Master and Slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. Continuous SCK is only supported for CPHA=1. 0 Data is captured on the leading edge of SCK and changed on the following edge 1 Data is changed on the leading edge of SCK and captured on the following edge
7 LSBFE	LSB First. The LSBFE bit selects if the LSB or MSB of the frame is transferred first. This bit is used only in Master mode. When operating in TSB configuration, this bit should always be 1. 0 Data is transferred MSB first 1 Data is transferred LSB first
8–9 PCSSCK[0:1]	PCS to SCK Delay Prescaler. The PCSSCK field selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. This field is used only in Master mode. The table below lists the prescaler values. See the CSSCK[0:3] field description for details on how to compute the PCS to SCK Delay.

PCSSCK	PCS to SCK delay prescaler value
00	1
01	3
10	5
11	7

Table 11-5. DSPI*x*_CTAR*n* field descriptions (continued)

<table>
<tr><th>Field</th><th>Descriptions</th></tr>
<tr><td>10–11
PASC[0:1]</td><td>After SCK Delay Prescaler. The PASC field selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. This field is used only in Master mode. The table below lists the prescaler values. See the ASC[0:3] field description for details on how to compute the After SCK Delay.
<table>
<tr><th>PASC</th><th>After SCK delay prescaler value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>12–13
PDT[0:1]</td><td>Delay after Transfer Prescaler. The PDT field selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is used only in Master mode. The table below lists the prescaler values. See the DT[0:3] field description for details on how to compute the Delay after Transfer.
<table>
<tr><th>PDT</th><th>Delay after transfer prescaler value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>14–15
PBR[0:1]</td><td>Baud Rate Prescale. The PBR field selects the prescaler value for the baud rate. This field is used only in Master mode. The baud rate is the frequency of the Serial Communications Clock (SCK). The system clock is divided by the prescaler value before the baud rate selection takes place. The Baud Rate Prescaler values are listed in the table below. See the BR[0:3] field description for details on how to compute the baud rate.
<table>
<tr><th>PBR</th><th>Baud rate prescaler value</th></tr>
<tr><td>00</td><td>2</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>16–19
CSSCK[0:3]</td><td>PCS to SCK Delay Scaler. The CSSCK field selects the scaler value for the PCS to SCK delay. This field is used only in Master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. Table 11-8 list the scaler values.The PCS to SCK Delay is a multiple of the system clock period and it is computed according to the following equation:
$$t_{CSC} = \frac{1}{f_{SYS}} \times PCSSCK \times CSSCK$$ ***Eqn. 11-1***</td></tr>
</table>

Table 11-5. DSPI*x*_CTAR*n* field descriptions (continued)

Field	Descriptions
20–23 ASC[0:3]	After SCK Delay Scaler. The ASC field selects the scaler value for the After SCK Delay. This field is used only in Master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. Table 11-9 list the scaler values.The After SCK Delay is a multiple of the system clock period, and it is computed according to the following equation: $t_{ASC} = \frac{1}{f_{SYS}} \times PASC \times ASC$ ***Eqn. 11-2***
24–27 DT[0:3]	Delay after Transfer Scaler. The DT field selects the Delay after Transfer Scaler. This field is used only in Master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. Table 11-10 lists the scaler values. In the Continuous Serial Communications Clock operation the DT value is fixed to one TSCK, except when the TSBC bit from DSPI_DSICR register enables the TSB configuration. The Delay after Transfer is a multiple of the system clock period and it is computed according to the following equation: $t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT$ ***Eqn. 11-3***
28–31 BR[0:3]	Baud Rate Scaler. The BR field selects the scaler value for the baud rate. This field is used only in Master mode. The prescaled system clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. Table 11-11 lists the Baud Rate Scaler values.The baud rate is computed according to the following equation: $\text{SCK baud rate} = \frac{f_{SYS}}{PBR} \times \frac{1 + DBR}{BR}$ ***Eqn. 11-4***

Table 11-6. DSPI SCK duty cycle

DBR	CPHA	PBR	SCK duty cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 11-7. DSPI Transfer frame size

FMSZ	Frame size	FMSZ	Frame size
0000	Reserved	1000	9
0001	Reserved	1001	10
0010	Reserved	1010	11
0011	4	1011	12

Table 11-7. DSPI Transfer frame size (continued)

FMSZ	Frame size	FMSZ	Frame size
0100	5	1100	13
0101	6	1101	14
0110	7	1110	15
0111	8	1111	16

Table 11-8. DSPI PCS to SCK delay scaler

CSSCK	PCS to SCK delay scaler value	CSSCK	PCS to SCK delay scaler value
0000	2	1000	512
0001	4	1001	1024
0010	8	1010	2048
0011	16	1011	4096
0100	32	1100	8192
0101	64	1101	16384
0110	128	1110	32768
0111	256	1111	65536

Table 11-9. DSPI after SCK delay scaler

ASC	After SCK delay scaler value	ASC	After SCK delay scaler value
0000	2	1000	512
0001	4	1001	1024
0010	8	1010	2048
0011	16	1011	4096
0100	32	1100	8192
0101	64	1101	16384
0110	128	1110	32768
0111	256	1111	65536

Table 11-10. DSPI delay after transfer scaler

DT	Delay after transfer scaler value	DT	Delay after transfer scaler value
0000	2	1000	512
0001	4	1001	1024
0010	8	1010	2048
0011	16	1011	4096

Table 11-10. DSPI delay after transfer scaler (continued) (continued)

DT	Delay after transfer scaler value	DT	Delay after transfer scaler value
0100	32	1100	8192
0101	64	1101	16384
0110	128	1110	32768
0111	256	1111	65536

Table 11-11. DSPI baud rate scaler

BR	Baud rate scaler value	BR	Baud rate scaler value
0000	2	1000	256
0001	4	1001	512
0010	6	1010	1024
0011	8	1011	2048
0100	16	1100	4096
0101	32	1101	8192
0110	64	1110	16384
0111	128	1111	32768

11.7.2.4 DSPI Status Register (DSPI*x*_SR)

The DSPI*x*_SR contains status and flag bits. The bits are set by the hardware and reflect the status of the DSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the DSPI*x*_SR by writing a 1 to clear (w1c). Writing a 0 to a flag bit has no effect.

Address: Base + 0x002C — Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	0	0	RFOF	0	RFDF	0
W	w1c			w1c	w1c		w1c						w1c		w1c	
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI Status Register (DSPI*x*_SR)

Table 11-12 describes the fields in the DSPI status register.

Table 11-12. DSPI*x*_SR field descriptions

Field	Description
0 TCF	TransfeR Complete Flag. Indicates that all bits in a frame have been shifted out. The TCF bit is set after the last incoming data bit is sampled, but before the t_{ASC} delay starts. Refer to Section 11.8.5.1, Classic SPI transfer format (CPHA = 0) for details. The TCF bit is cleared by writing 1 to it. 0 Transfer not complete 1 Transfer complete
1 TXRXS	TX and RX Status. Reflects the status of the DSPI. Refer to Section 11.8.2, Start and stop of DSPI transfers for information on what clears and sets this bit. 0 TX and RX operations are disabled (DSPI is in Stopped state) 1 TX and RX operations are enabled (DSPI is in Running state)
2	Reserved.
3 EOQF	End Of Queue Flag. Indicates that transmission in progress is the last entry in a queue. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and after the last incoming data bit is sampled, but before the t_{ASC} delay starts. Refer to Section 11.8.5.1, Classic SPI transfer format (CPHA = 0) for details. The EOQF bit is cleared by writing 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command 1 EOQ bit is set in the executing SPI command **Note:** EOQF does not function in Slave mode.
4 TFUF	Transmit FIFO Underflow Flag. Indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in Slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in SPI Slave mode is empty, and a transfer is initiated by an external SPI master. The TFUF bit is cleared by writing 1 to it. 0 TX FIFO underflow has not occurred 1 TX FIFO underflow has occurred
5	Reserved.
6 TFFF	Transmit FIFO Fill Flag. Indicates that the TX FIFO can be filled. Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it, or an by acknowledgement from the eDMA controller when the TX FIFO is full. 0 TX FIFO is full 1 TX FIFO is not full
7–11	Reserved.
12 RFOF	Receive FIFO Overflow Flag. Indicates that an overflow condition in the RX FIFO has occurred. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit is cleared by writing 1 to it. 0 RX FIFO overflow has not occurred 1 RX FIFO overflow has occurred
13	Reserved.

Table 11-12. DSPI*x*_SR field descriptions (continued)

Field	Description
14 RFDF	Receive FIFO Drain Flag. Indicates that the RX FIFO can be drained. Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it, or by acknowledgement from the eDMA controller when the RX FIFO is empty. 0 RX FIFO is empty 1 RX FIFO is not empty **Note:** In the interrupt service routine, RFDF must be cleared only after the DSPIx_POPR register is read.
15	Reserved.
16–19 TXCTR [0:3]	TX FIFO Counter. Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the DSPI _PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20–23 TXNXTPTR [0:3]	Transmit Next Pointer. Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. Refer to Section 11.8.3.4, transmit First In First Out (TX FIFO) buffering mechanism for more details.
24–27 RXCTR [0:3]	RX FIFO Counter. Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the DSPI _POPR is read. The RXCTR is incremented after the last incoming data bit is sampled, but before the t_{ASC} delay starts. Refer to Section 11.8.5.1, Classic SPI transfer format (CPHA = 0) for details.
28–31 POPNXTPTR [0:3]	Pop Next Pointer. Contains a pointer to the RX FIFO entry that is returned when the DSPI*x*_POPR is read. The POPNXTPTR is updated when the DSPI*x*_POPR is read. Refer to Section 11.8.3.5, Receive First In First Out (RX FIFO) buffering mechanism for more details.

11.7.2.5 DSPI DMA / Interrupt Request Select and Enable Register (DSPI*x*_RSER)

The DSPI*x*_RSER serves two purposes: it enables flag bits in the DSPI*x*_SR to generate DMA requests or interrupt requests, and selects the type of request to generate. Refer to the bit descriptions for the type of requests that are supported. Do not write to the DSPI*x*_RSER while the DSPI is running.

Address: Base + 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_RE	0	0	EOQF_RE	TFUF_RE	0	TFFF_RE	TFFF_DIRS	0	0	0	0	RFOF_RE	0	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 11-6. DSPI DMA / Interrupt Request Select and Enable Register (DSPI*x*_RSER)

Table 11-13 describes the fields in the DSPI DMA / interrupt request and enable register.

Table 11-13. DSPI*x*_RSER field descriptions

Field	Description
0 TCF_RE	Transmission complete request enable. Enables TCF flag in the DSPI*x*_SR to generate an interrupt request. 0 TCF interrupt requests are disabled 1 TCF interrupt requests are enabled
1–2	Reserved.
3 EOQF_RE	DSPI finished request enable. Enables the EOQF flag in the DSPI*x*_SR to generate an interrupt request. 0 EOQF interrupt requests are disabled 1 EOQF interrupt requests are enabled
4 TFUF_RE	Transmit FIFO underflow request enable. The TFUF_RE bit enables the TFUF flag in the DSPI*x*_SR to generate an interrupt request. 0 TFUF interrupt requests are disabled 1 TFUF interrupt requests are enabled
5	Reserved.
6 TFFF_RE	Transmit FIFO fill request enable. Enables the TFFF flag in the DSPI*x*_SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA requests. 0 TFFF interrupt requests or DMA requests are disabled 1 TFFF interrupt requests or DMA requests are enabled
7 TFFF_DIRS	Transmit FIFO fill DMA or interrupt request select. Selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the DSPI*x*_SR is set, and the TFFF_RE bit in the DSPI*x*_RSER is set, this bit selects between generating an interrupt request or a DMA request. 0 Interrupt request is selected 1 DMA request is selected
8–11	Reserved.
12 RFOF_RE	Receive FIFO overflow request enable. Enables the RFOF flag in the DSPI*x*_SR to generate an interrupt request. 0 RFOF interrupt requests are disabled 1 RFOF interrupt requests are enabled
13	Reserved.
14 RFDF_RE	Receive FIFO drain request enable. Enables the RFDF flag in the DSPI*x*_SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt requests or DMA requests are disabled 1 RFDF interrupt requests or DMA requests are enabled

Table 11-13. DSPI*x*_RSER field descriptions (continued)

Field	Description
15 RFDF_DIRS	Receive FIFO drain DMA or interrupt request select. Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the DSPI*x*_SR is set, and the RFDF_RE bit in the DSPI*x*_RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request is selected 1 DMA request is selected
16–31	Reserved.

11.7.2.6 DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)

The DSPI*x*_PUSHR provides a means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. Refer to Section 11.8.3.4, transmit First In First Out (TX FIFO) buffering mechanism, for more information. Write accesses of 8 or 16 bits to the DSPI*x*_PUSHR transfer 32 bits to the TX FIFO.

NOTE

TXDATA is used in Master and Slave modes.

Address: Base + 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CONT	CTAS			EOQ	CT CNT	0	0	0	0	0	0	0	PCS 2	PCS 1	PCS 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-7. DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)

Table 11-14 describes the fields in the DSPI push transmit FIFO register.

Table 11-14. DSPI*x*_PUSHR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0
CONT</td><td>Continuous peripheral chip select enable. Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected CS signals to remain asserted between transfers. Refer to Section 11.8.5.5, Continuous selection format, for more information.
0 Return peripheral chip select signals to their inactive state between transfers
1 Keep peripheral chip select signals asserted between transfers</td></tr>
<tr><td>1–3
CTAS
[0:2]</td><td>Clock and transfer attributes select. Selects which of the DSPIx_CTARs is used to set the transfer attributes for the SPI frame. In SPI Slave mode, DSPIx_CTAR0 is used. The following table shows how the CTAS values map to the DSPIx_CTARs. There are eight DSPIx_CTARs in the device DSPI implementation.
Note: Use in SPI Master mode only.
<table>
<tr><th>CTAS</th><th>Use clock and transfer attributes from</th></tr>
<tr><td>000</td><td>DSPIx_CTAR0</td></tr>
<tr><td>001</td><td>DSPIx_CTAR1</td></tr>
<tr><td>010</td><td>DSPIx_CTAR2</td></tr>
<tr><td>011</td><td>DSPIx_CTAR3</td></tr>
<tr><td>100</td><td>DSPIx_CTAR4</td></tr>
<tr><td>101</td><td>DSPIx_CTAR5</td></tr>
<tr><td>110</td><td>DSPIx_CTAR6</td></tr>
<tr><td>111</td><td>DSPIx_CTAR7</td></tr>
</table></td></tr>
<tr><td>4
EOQ</td><td>End of queue. Provides a means for host software to signal to the DSPI that the current SPI transfer is the last in a queue. At the end of the transfer the EOQF bit in the DSPIx_SR is set.
0 The SPI data is not the last data to transfer
1 The SPI data is the last data to transfer
Note: Use in SPI Master mode only.</td></tr>
<tr><td>5
CTCNT</td><td>Clear SPI_TCNT. Provides a means for host software to clear the SPI transfer counter. The CTCNT bit clears the SPI_TCNT field in the DSPIx_TCR. The SPI_TCNT field is cleared before transmission of the current SPI frame begins.
0 Do not clear SPI_TCNT field in the DSPIx_TCR
1 Clear SPI_TCNT field in the DSPIx_TCR
Note: Use in SPI Master mode only.</td></tr>
<tr><td>6–7</td><td>Reserved.</td></tr>
<tr><td>8–9</td><td>Reserved, but implemented. These bits are writable, but have no effect.</td></tr>
<tr><td>10–12</td><td>Reserved.</td></tr>
<tr><td>13–15
PCSx</td><td>Peripheral chip select x. Selects which CSx signals are asserted for the transfer.
0 Negate the CSx signal
1 Assert the CSx signal
Note: Use in SPI Master mode only.</td></tr>
<tr><td>16–31
TXDATA
[0:15]</td><td>Transmit data. Holds SPI data for transfer according to the associated SPI command.
Note: Use TXDATA in Master and Slave modes.</td></tr>
</table>

11.7.2.7 DSPI POP RX FIFO Register (DSPI*x*_POPR)

The DSPI*x*_POPR allows you to read the RX FIFO. Refer to Section 11.8.3.5, Receive First In First Out (RX FIFO) buffering mechanism for a description of the RX FIFO operations. 8- or 16-bit read accesses to the DSPI*x*_POPR fetch the RX FIFO data, and update the counter and pointer.

NOTE

Reading the RX FIFO field fetches data from the RX FIFO. Once the RX FIFO is read, the read data pointer is moved to the next entry in the RX FIFO. Therefore, read DSPI*x*_POPR only when you need the data. For compatibility, configure the TLB (MMU table) entry for DSPI*x*_POPR as guarded.

Address: Base + 0x0038 Access: R/O

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-8. DSPI POP RX FIFO Register (DSPI*x*_POPR)

Table 11-15 describes the fields in the DSPI pop receive FIFO register.

Table 11-15. DSPI*x*_POPR field descriptions

Field	Description
0–15	Reserved, must be cleared.
16–31 RXDATA [0:15]	Received data. The RXDATA field contains the SPI data from the RX FIFO entry pointed to by the pop next data pointer (POPNXTPTR).

11.7.2.8 DSPI Transmit FIFO Registers 0–4 (DSPI*x*_TXFR*n*)

The DSPI*x*_TXFR*n* registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the DSPI*x*_TXFR*n* registers does not alter the state of the TX FIFO. The MCU uses four registers to implement the TX FIFO: DSPI*x*_TXFR0–DSPI*x*_TXFR3.

Address:
Base + 0x003C (DSPI*x*_TXFR0)
Base + 0x0040 (DSPI*x*_TXFR1)
Base + 0x0044 (DSPI*x*_TXFR2)
Base + 0x0048 (DSPI*x*_TXFR3)
Base + 0x004C (DSPI*x*_TXFR4)

Access: R/O

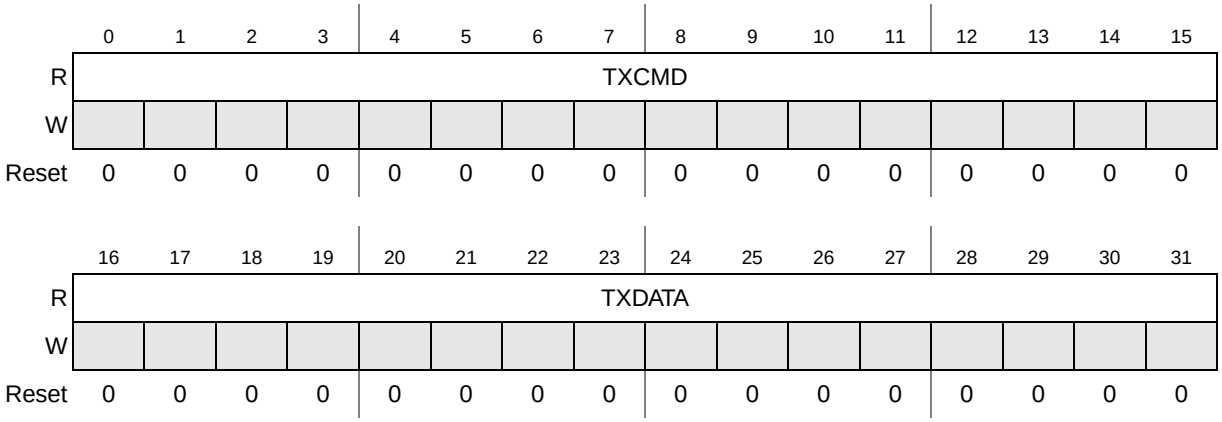


Figure 11-9. DSPI Transmit FIFO Register 0–4 (DSPI*x*_TXFR*n*)

Table 11-16 describes the fields in the DSPI transmit FIFO register.

Table 11-16. DSPI*x*_TXFR*n* field descriptions

Field	Description
0–15 TXCMD [0:15]	Transmit command. Contains the command that sets the transfer attributes for the SPI data. Refer to Section 11.7.2.6, DSPI PUSH TX FIFO Register (DSPIx_PUSHR), for details on the command field.
16–31 TXDATA [0:15]	Transmit data. Contains the SPI data to be shifted out.

11.7.2.9 DSPI Receive FIFO Registers 0–4 (DSPI*x*_RXFR*n*)

The DSPI*x*_RXFR*n* registers provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The DSPI*x*_RXFR registers are read-only. Reading the DSPI*x*_RXFR*n* registers does not alter the state of the RX FIFO. The device uses four registers to implement the RX FIFO: DSPI*x*_RXFR0–DSPI*x*_RXFR3.

Address: Access: R/O
Base + 0x007C (DSPI*x*_RXFR0)
Base + 0x0080 (DSPI*x*_RXFR1)
Base + 0x0084 (DSPI*x*_RXFR2)
Base + 0x0088 (DSPI*x*_RXFR3)
Base + 0x008C (DSPI*x*_RXFR4)

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-10. DSPI Receive FIFO Registers 0–4 (DSPI*x*_RXFR*n*)

Table 11-17 describes the field in the DSPI receive FIFO register.

Table 11-17. DSPI*x*_RXFR*n* field descriptions

Field	Description
0–15	Reserved, must be cleared.
16–31 RXDATA [15:0]	Receive data. Contains the received SPI data.

11.8 Functional description

The DSPI supports full-duplex, synchronous serial communications between the MCU and peripheral devices. All communications are through an SPI-like protocol.

The DSPI has one configuration:

- Serial peripheral interface (SPI) configuration in which the DSPI operates as a basic SPI or a queued SPI

The DCONF field in the DSPI*x*_MCR register determines the DSPI configuration. Refer to Table 11-3 for the DSPI configuration values.

The DSPI*x*_CTAR0–DSPI*x*_CTAR7 registers hold clock and transfer attributes.The SPI configuration can select which CTAR to use on a frame-by-frame basis by setting the CTAS field in the DSPI*x*_PUSHR.

The 16-bit shift register in the master and the 16-bit shift register in the slave are linked by the SOUT_*x* and SIN_*x* signals to form a distributed 32-bit register. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the registers are linked, data is exchanged between the master and the slave; the data that was in the master's shift register is now in the

shift register of the slave, and vice versa. At the end of a transfer, the TCF bit in the DSPI*x*_SR is set to indicate a completed transfer. Figure 11-11 illustrates how master and slave data are exchanged.

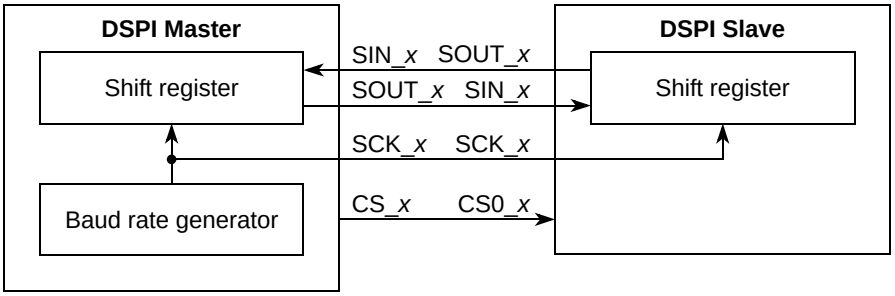


Figure 11-11. SPI serial protocol overview

The DSPI has three peripheral chip select (CS*x*) signals that select the slaves with which to communicate.

Transfer protocols and timing properties are shared by the three DSPI configurations; these properties are described independently of the configuration in Section 11.8.5, Transfer formats. The transfer rate and delay settings are described in section Section 11.8.4, DSPI baud rate and clock delay generation.

Refer to Section 11.8.8, Power-saving features, for information on the power-saving features of the DSPI.

11.8.1 Modes of operation

The DSPI modules have five available distinct modes:

- Master mode
- Slave mode
- Module Disable mode
- External stop mode
- Debug mode

Master, Slave, and Module Disable modes are module-specific modes. The external stop and debug modes are device-specific modes.

The module-specific modes are determined by bits in the DSPI*x*_MCR. The device-specific modes are modes that the entire device can enter in parallel with the DSPI being configured in one of its module-specific modes.

11.8.1.1 Master mode

In Master mode, the DSPI can initiate communications with peripheral devices. The DSPI operates as bus master when the MSTR bit in the DSPI*x*_MCR is set. The serial communications clock (SCK) is controlled by the master DSPI. All three DSPI configurations are valid in Master mode.

In SPI configuration, Master mode transfer attributes are controlled by the SPI command in the current TX FIFO entry. The CTAS field in the SPI command selects which of the eight DSPI*x*_CTARs are used to set the transfer attributes. Transfer attribute control is on a frame-by-frame basis.

Refer to Section 11.8.3, Serial Peripheral Interface (SPI) configuration, for more details.

11.8.1.2 Slave mode

In Slave mode the DSPI responds to transfers initiated by an SPI master. The DSPI operates as bus slave when the MSTR bit in the DSPI*x*_MCR is negated. The DSPI slave is selected by a bus master by having the slave's CS0_*x* asserted. In Slave mode the SCK is provided by the bus master. All transfer attributes are controlled by the bus master, except the clock polarity, clock phase, and the number of bits to transfer, which must be configured in the DSPI slave to communicate correctly.

11.8.1.3 Module Disable mode

The Module Disable mode is used for MCU power management. The clock to the non-memory-mapped logic in the DSPI is stopped while in Module Disable mode. The DSPI enters the Module Disable mode when the MDIS bit in DSPI*x*_MCR is set.

Refer to Section 11.8.8, Power-saving features, for more details on the Module Disable mode.

11.8.1.4 External Stop mode

For devices with low-power modes, the DSPI supports the Global Signal Stop mode mechanism. The DSPI will not acknowledge the request to enter External Stop mode until it has reached a frame boundary. When the DSPI has reached a frame boundary it will halt all operations and indicate that it is ready to have its clocks shut off. The DSPI exits External Stop mode and resumes normal operation once the clocks are turned on. Serial communications or register accesses made while in External Stop mode are ignored even if the clocks have not been shut off yet. See Section 11.8.8, Power-saving features, for more details on the External Stop mode.

11.8.1.5 Debug mode

The debug mode is used for system development and debugging. If the MCU is stopped by a debugger while the DSPI*x*_MCR[FRZ] bit is set, the DSPI halts operation on the next frame boundary and enters a stopped state. If the MCU is stopped by a debugger while the FRZ bit is cleared, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI. The FRZ bit operation is only available when the CPU has an active debug mode.

See Figure 11-12 for a state diagram.

11.8.2 Start and stop of DSPI transfers

The DSPI has two operating states: Stopped and Running. The states are independent of DSPI configuration. The default state of the DSPI is Stopped. In the Stopped state no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results. The TXRXS bit in the DSPI*x*_SR is cleared in this state. In the Running state, serial transfers take place. The TXRXS bit in the DSPI*x*_SR is set in the Running state.

Figure 11-12 shows a state diagram of the start and stop mechanism.

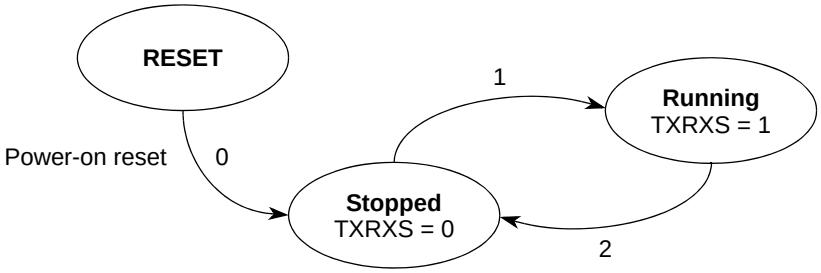


Figure 11-12. DSPI start and stop state diagram

The transitions are described in Table 11-18.

Table 11-18. State transitions for start and stop of DSPI transfers

Transition #	Current state	Next State	Description
0	RESET	Stopped	Generic power-on reset transition
1	Stopped	Running	The DSPI starts (transitions from Stopped to Running) when all of the following conditions are true: • EOQF bit is clear • Debug mode is not selected or the FRZ bit is clear • HALT bit is clear
2	Running	Stopped	The DSPI stops (transitions from Running to Stopped) after the current frame for any one of the following conditions: • EOQF bit is set • Debug mode is selected and the FRZ bit is set • HALT bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or on the next system clock cycle if no transfers are in progress.

11.8.3 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The DSPI is in SPI configuration when the DCONF field in the DSPI*x*_MCR is 0b00. The SPI frames can be from 4 to 16 bits long. The data to be transmitted can come from queues stored in RAM external to the DSPI. Host software or an eDMA controller can transfer the SPI data from the queues to a first-in first-out (FIFO) buffer. The received data is stored in entries in the receive FIFO (RX FIFO) buffer. Host software or an eDMA controller transfers the received data from the RX FIFO to memory external to the DSPI.

The FIFO buffer operations are described in Section 11.8.3.4, transmit First In First Out (TX FIFO) buffering mechanism, and Section 11.8.3.5, Receive First In First Out (RX FIFO) buffering mechanism.

The interrupt and DMA request conditions are described in Section 11.8.7, Interrupts/DMA requests.

The SPI configuration supports two module-specific modes; Master mode and Slave mode. The FIFO operations are similar for the Master mode and Slave mode. The main difference is that in Master mode

the DSPI initiates and controls the transfer according to the fields in the SPI command field of the TX FIFO entry. In Slave mode the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field of the TX FIFO entry is ignored.

11.8.3.1 SPI Master mode

In SPI Master mode the DSPI initiates the serial transfers by controlling the serial communications clock (SCK_*x*) and the peripheral chip select (CS*x*) signals. The SPI command field in the executing TX FIFO entry determines which CTARs are used to set the transfer attributes and which CS*x* signal to assert. The command field also contains various bits that help with queue management and transfer protocol. The data field in the executing TX FIFO entry is loaded into the shift register and shifted out on the serial out (SOUT_*x*) pin. In SPI Master mode each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame-by-frame basis.

Refer to Section 11.7.2.6, DSPI PUSH TX FIFO Register (DSPIx_PUSHR), for details on the SPI command fields.

11.8.3.2 SPI Slave mode

In SPI Slave mode the DSPI responds to transfers initiated by an SPI bus master. The DSPI does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the DSPI*x*_CTAR0 register.

11.8.3.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The TX and RX FIFOs are disabled separately. The TX FIFO is disabled by writing a 1 to the DIS_TXF bit in the DSPI*x*_MCR. The RX FIFO is disabled by writing a 1 to the DIS_RXF bit in the DSPI*x*_MCR.

The FIFO disable mechanisms are transparent to the user and to host software; transmit data and commands are written to the DSPI*x*_PUSHR and received data is read from the DSPI*x*_POPR. When the TX FIFO is disabled, the TFFF, TFUF, and TXCTR fields in DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_TXFRs and TXNXTPTR are undefined. When the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_RXFRs and POPNXTPTR are undefined.

Disable the TX and RX FIFOs only if the FIFO must be disabled as a requirement of the application's operating mode. A FIFO must be disabled before it is accessed. Failure to disable a FIFO prior to a first FIFO access is not supported, and can result in incorrect results.

11.8.3.4 transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data and SPI commands for transmission. The TX FIFO holds five entries, each consisting of a command field and a data field. SPI commands and data are added to the TX FIFO by writing to the DSPI push TX FIFO register (DSPI*x*_PUSHR). For more information on

DSPI*x*_PUSHR, refer to Section 11.7.2.6, DSPI PUSH TX FIFO Register (DSPIx_PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO counter field (TXCTR) in the DSPI Status Register (DSPI*x*_SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time the DSPI _PUSHR is written or SPI data is transferred into the shift register from the TX FIFO.

Refer to Section 11.7.2.4, DSPI Status Register (DSPIx_SR), for more information on DSPI*x*_SR.

The TXNXTPTR field indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR contains the positive offset from DSPI*x*_TXFR0 in number of 32-bit registers. For example, TXNXTPTR equal to two means that the DSPI*x*_TXFR2 contains the SPI data and command for the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register.

11.8.3.4.1 Filling the TX FIFO

Host software or the eDMA controller can add (push) entries to the TX FIFO by writing to the DSPI*x*_PUSHR. When the TX FIFO is not full, the TX FIFO fill flag (TFFF) in the DSPI*x*_SR is set. The TFFF bit is cleared when the TX FIFO is full and the eDMA controller indicates that a write to DSPI*x*_PUSHR is complete, or alternatively by host software writing a 1 to the TFFF in the DSPI*x*_SR. The TFFF can generate a DMA request or an interrupt request.

Refer to Section 11.8.7.2, Transmit FIFO fill interrupt or DMA request (TFFF), for details.

The DSPI ignores attempts to push data to a full TX FIFO; that is, the state of the TX FIFO is unchanged. No error condition is indicated.

11.8.3.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO counter is decremented by one. At the end of a transfer, the TCF bit in the DSPI*x*_SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a 1 to the CLR_TXF bit in DSPI*x*_MCR.

If an external SPI bus master initiates a transfer with a DSPI slave while the slave's DSPI TX FIFO is empty, the transmit FIFO underflow flag (TFUF) in the slave's DSPI*x*_SR is set.

Refer to Section 11.8.7.4, Transmit FIFO underflow interrupt request (TFUF), for details.

11.8.3.5 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds four received SPI data frames. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data is removed (popped) from the RX FIFO by reading the DSPI*x*_POPR register. RX FIFO entries can only be removed from the RX FIFO by reading the DSPI*x*_POPR or by flushing the RX FIFO.

Refer to Section 11.7.2.7, DSPI POP RX FIFO Register (DSPIx_POPR), for more information on the DSPI*x*_POPR.

The RX FIFO counter field (RXCTR) in the DSPI status register (DSPI*x*_SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the DSPI _POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the DSPI*x*_SR points to the RX FIFO entry that is returned when the DSPI*x*_POPR is read. The POPNXTPTR contains the positive, 32-bit word offset from DSPI*x*_RXFR0. For example, POPNXTPTR equal to two means that the DSPI*x*_RXFR2 contains the received SPI data that is returned when DSPI*x*_POPR is read. The POPNXTPTR field is incremented every time the DSPI*x*_POPR is read. POPNXTPTR rolls over every four frames on the MCU.

11.8.3.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. If the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO the RX FIFO counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the DSPI*x*_SR is set, indicating an overflow condition. Depending on the state of the ROOE bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is ignored or put in the shift register. If the ROOE bit is set, the incoming data is put in the shift register. If the ROOE bit is cleared, the incoming data is ignored.

11.8.3.5.2 Draining the RX FIFO

Host software or the eDMA can remove (pop) entries from the RX FIFO by reading the DSPI*x*_POPR. A read of the DSPI*x*_POPR decrements the RX FIFO counter by one. Attempts to pop data from an empty RX FIFO are ignored, and the RX FIFO counter remains unchanged. The data returned from reading an empty RX FIFO is undetermined.

Refer to Section 11.7.2.7, DSPI POP RX FIFO Register (DSPIx_POPR), for more information on DSPI*x*_POPR.

When the RX FIFO is not empty, the RX FIFO drain flag (RFDF) in the DSPI*x*_SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the eDMA controller indicates that a read from DSPI*x*_POPR is complete; alternatively the RFDF bit can be cleared by the host writing a 1 to it.

11.8.4 DSPI baud rate and clock delay generation

The SCK_*x* frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option of doubling the baud rate.

Figure 11-13 shows conceptually how the SCK signal is generated.

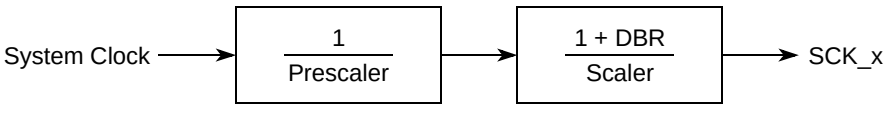


Figure 11-13. Communications clock prescalers and scalers

11.8.4.1 Baud rate generator

The baud rate is the frequency of the serial communication clock (SCK_*x*). The system clock is divided by a baud rate prescaler (defined by DSPI*x*_CTAR[PBR]) and baud rate scaler (defined by DSPI*x*_CTAR[BR]) to produce SCK_*x* with the possibility of doubling the baud rate. The DBR, PBR, and BR fields in the DSPI*x*_CTARs select the frequency of SCK_*x* using the following formula:

$$\text{SCK baud rate} = \frac{f_{SYS}}{\text{PBRPrescalerValue}} \times \frac{1 + \text{DBR}}{\text{BRScalerValue}}$$

Table 11-19 shows an example of a computed baud rate.

Table 11-19. Baud rate computation example

f_{SYS}	PBR	Prescaler value	BR	Scaler value	DBR value	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

11.8.4.2 CS to SCK delay (t_{CSC})

The CS_*x* to SCK_*x* delay is the length of time from assertion of the CS_*x* signal to the first SCK_*x* edge. Refer to Figure 11-14 for an illustration of the CS_*x* to SCK_*x* delay. The PCSSCK and CSSCK fields in the DSPI*x*_CTAR*n* registers select the CS_*x* to SCK_*x* delay, and the relationship is expressed by the following formula:

$$t_{CSC} = \frac{1}{f_{SYS}} \times \text{PCSSCK} \times \text{CSSCK}$$

Table 11-20 shows an example of the computed CS to SCK_*x* delay.

Table 11-20. CS to SCK delay computation example

PCSSCK	Prescaler value	CSSCK	Scaler value	f_{SYS}	CS to SCK delay
0b01	3	0b0100	32	100 MHz	0.96 µs

11.8.4.3 After SCK delay (t_{ASC})

The after SCK_*x* delay is the length of time between the last edge of SCK_*x* and the negation of CS_*x*. Refer to Figure 11-14 and Figure 11-15 for illustrations of the after SCK_*x* delay. The PASC and ASC fields in the DSPI*x*_CTAR*n* registers select the after SCK delay. The relationship between these variables is given in the following formula:

$$t_{ASC} = \frac{1}{f_{SYS}} \times \text{PASC} \times \text{ASC}$$

Table 11-21 shows an example of the computed after SCK delay.

Table 11-21. After SCK delay computation example

PASC	Prescaler value	ASC	Scaler value	f_{SYS}	After SCK delay
0b01	3	0b0100	32	100 MHz	0.96 µs

11.8.4.4 Delay after transfer (t_{DT})

The delay after transfer is the length of time between negation of the CS*x* signal for a frame and the assertion of the CS*x* signal for the next frame. The PDT and DT fields in the DSPI*x*_CTAR*n* registers select the delay after transfer.

Refer to Figure 11-14 for an illustration of the delay after transfer.

The following formula expresses the PDT/DT/delay after transfer relationship:

$$t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT$$

Table 11-22 shows an example of the computed delay after transfer.

Table 11-22. Delay after transfer computation example

PDT	Prescaler value	DT	Scaler value	f_{SYS}	Delay after transfer
0b01	3	0b1110	32768	100 MHz	0.98 ms

When in non-continuous clock mode the t_{DT} delay is configurable as outlined in the DSPI_CTAR*x* registers. When in continuous clock mode and TSB is not enabled, the delay is fixed at 1 SCK period. When in TSB and continuous mode the delay is programmed as outlined in the DSPI_CTAR*x* registers, but in the event that the delay does not coincide with an SCK period in duration, then the delay is extended to the next SCK active edge. Table 11-23 shows an example of how to compute the delay after transfer with the clock period of SCK defined as T_{SCK}. The values calculated assume 1 TSCK period = 4 ipg_clk.

Table 11-23. Delay after transfer computation example in TSB configuration

		PDT field			
	t_{DT}[1] (Tsck)	0	1	2	3
DT field	0[2]	1	2	3	4
	1	1	3	5	7
	2	2	6	10	14
	3	4	12	20	28
	4	8	24	40	56
	5	16	48	80	112
	6	32	96	160	224
	7	64	192	320	448
	8	128	384	640	896
	9	256	768	1280	1792
	10	512	1536	2560	3584
	11	1024	3072	5120	7168
	12	2048	6144	10240	14336
	13	4096	12288	20480	28672
	14	8192	24576	40960	57344
	15	16384	49152	81920	114688

1 Some values are not reachable (for example, 9, 11, 13, 15, 17, 18, 19...). To calculate these values, please see Equation 11-3.

2 The values in this row were rounded to the next integer value.

11.8.5 Transfer formats

The SPI serial communication is controlled by the serial communications clock (SCK_*x*) signal and the CS*x* signals. The SCK_*x* signal provided by the master device synchronizes shifting and sampling of the data by the SIN_*x* and SOUT_*x* pins. The CS*x* signals serve as enable signals for the slave devices.

When the DSPI is the bus master, the CPOL and CPHA bits in the DSPI clock and transfer attributes registers (DSPI*x*_CTAR*n*) select the polarity and phase of the serial clock, SCK_*x*. The polarity bit selects the idle state of the SCK_*x*. The clock phase bit selects if the data on SOUT_*x* is valid before or on the first SCK_*x* edge.

When the DSPI is the bus slave, CPOL and CPHA bits in the DSPI*x*_CTAR0 (SPI Slave mode) select the polarity and phase of the serial clock. Even though the bus slave does not control the SCK signal, the clock polarity, clock phase, and number of bits to transfer must be identical for the master device and the slave device to ensure proper transmission.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0
- Classic SPI with CPHA = 1
- Modified transfer format with CPHA = 0
- Modified transfer format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the DSPI*x*_MCR selects between classic SPI format and modified transfer format. The classic SPI formats are described in Section 11.8.5.1, Classic SPI transfer format (CPHA = 0), and Section 11.8.5.2, Classic SPI transfer format (CPHA = 1). The modified transfer formats are described in Section 11.8.5.3, Modified SPI transfer format (MTFE = 1, CPHA = 0), and Section 11.8.5.4, Modified SPI transfer format (MTFE = 1, CPHA = 1).

In the SPI configuration, the DSPI provides the option of keeping the CS signals asserted between frames. Refer to Section 11.8.5.5, Continuous selection format, for details.

11.8.5.1 Classic SPI transfer format (CPHA = 0)

The transfer format shown in Figure 11-14 is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN_*x* pins on the odd-numbered SCK_*x* edges and change the data on their SOUT_*x* pins on the even-numbered SCK_*x* edges.

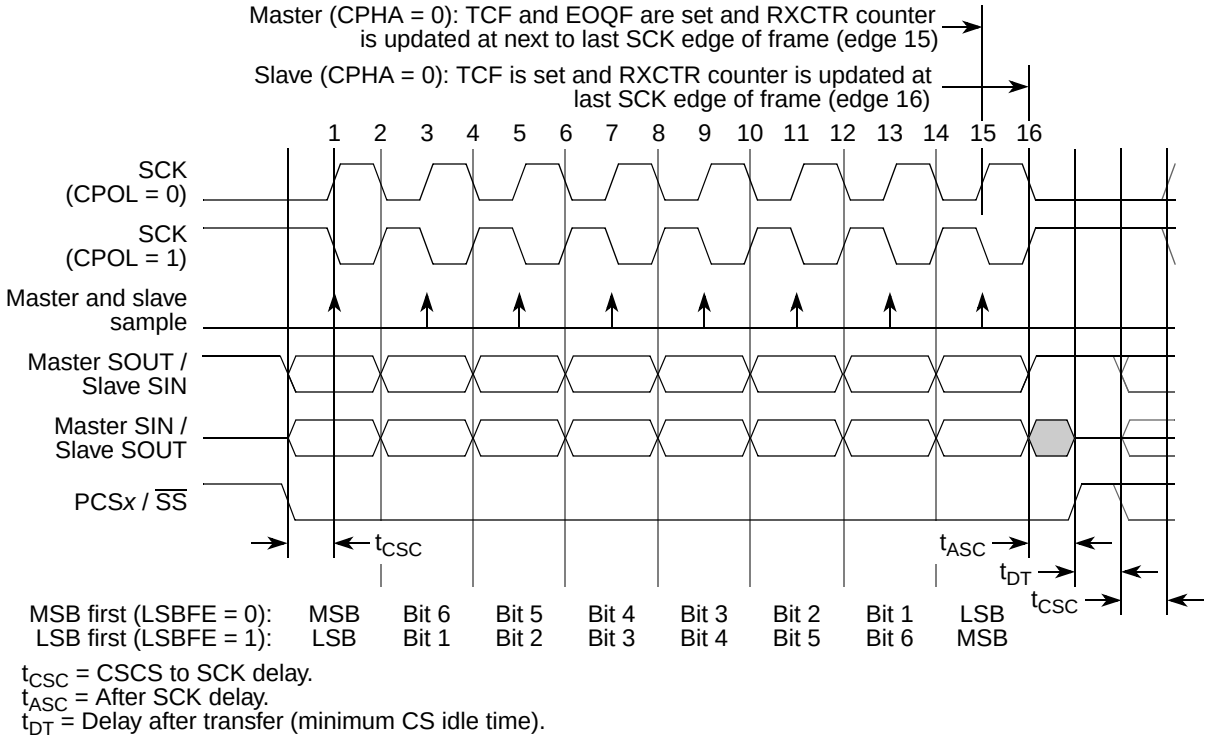


Figure 11-14. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)

The master initiates the transfer by placing its first data bit on the SOUT_*x* pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT_*x* pin. After the t_{CSC} delay has elapsed, the master outputs the first edge of SCK_*x*. This is the edge used by the master and slave devices to sample the first input data bit on their serial data input signals. At the second edge of the SCK_*x* the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN_*x* pins on the odd-numbered clock edges and change the data on their SOUT_*x* pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the CS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

For the CPHA = 0 condition of the master, TCF and EOQF are set and the RXCTR counter is updated at the next-to-last serial clock edge of the frame (edge 15) of Figure 11-14.

For the CPHA = 0 condition of the slave, TCF is set and the RXCTR counter is updated at the last serial clock edge of the frame (edge 16) of Figure 11-14.

11.8.5.2 Classic SPI transfer format (CPHA = 1)

The transfer format shown in Figure 11-15 is used to communicate with peripheral SPI slave devices that require the first SCK_*x* edge before the first data bit becomes available on the slave SOUT_*x* pin. In this format the master and slave devices change the data on their SOUT_*x* pins on the odd-numbered SCK_*x* edges and sample the data on their SIN_*x* pins on the even-numbered SCK_*x* edges.

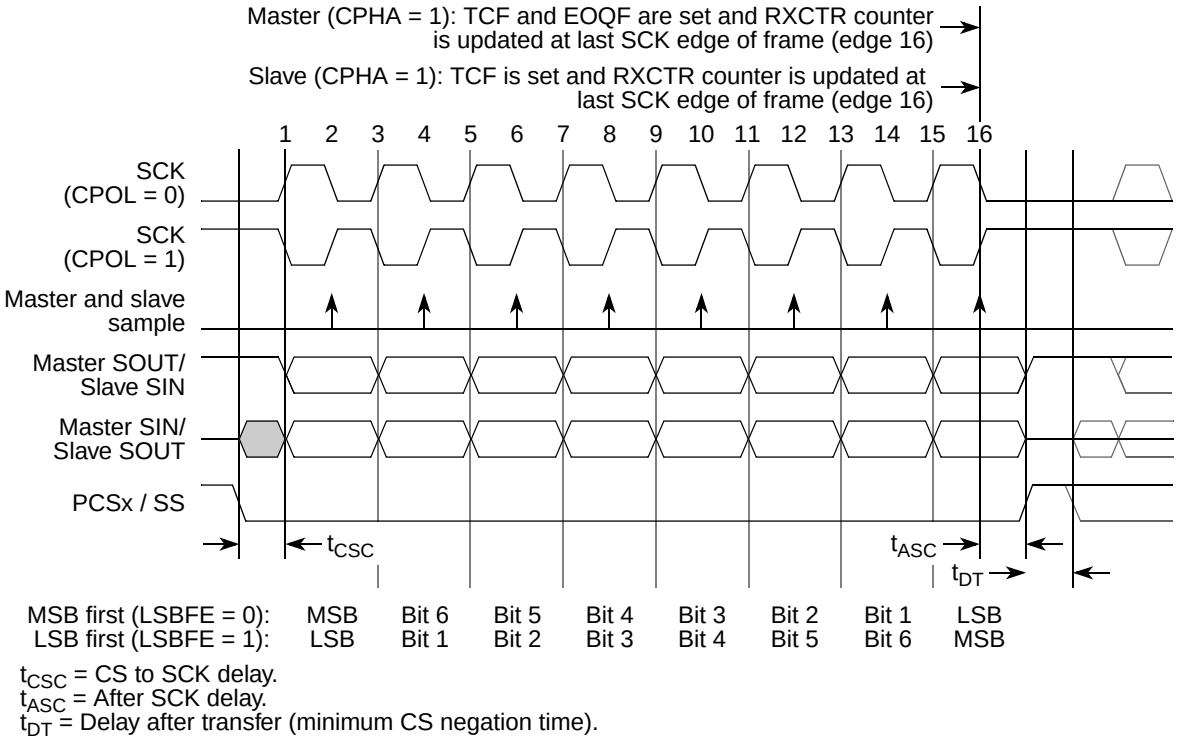


Figure 11-15. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)

The master initiates the transfer by asserting the CS*x* signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK_*x* edge and at the same time places valid data on the master SOUT_*x* pin. The slave responds to the first SCK_*x* edge by placing its first data bit on its slave SOUT_*x* pin.

At the second edge of the SCK_*x*, the master and slave sample their SIN_*x* pins. For the rest of the frame the master and the slave change the data on their SOUT_*x* pins on the odd-numbered clock, a delay of t_{ASC} is inserted before the master negates the CS*x* signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

For CPHA = 1 the master EOQF and TCF and slave TCF are set at the last serial clock edge (edge 16) of Figure 11-15. For CPHA = 1 the master and slave RXCTR counters are updated on the same clock edge.

11.8.5.3 Modified SPI transfer format (MTFE = 1, CPHA = 0)

In this modified transfer format, both the master and the slave sample later in the SCK period than in classic SPI mode, to allow for delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

NOTE

For the modified transfer format to operate correctly, you must thoroughly analyze the SPI link timing budget.

The master and the slave place data on the SOUT_*x* pins at the assertion of the CS*x* signal. After the CS*x* to SCK_*x* delay has elapsed, the first SCK_*x* edge is generated. The slave samples the master SOUT_*x* signal on every odd-numbered SCK_*x* edge. The slave also places new data on the slave SOUT_*x* on every odd-numbered clock edge.

The master places its second data bit on the SOUT_*x* line one system clock after an odd-numbered SCK_*x* edge. The point where the master samples the slave SOUT_*x* is selected by writing to the SMPL_PT field in the DSPI*x*_MCR. Table 11-24 lists the number of system clock cycles between the active edge of SCK_*x* and the master sample point for different values of the SMPL_PT bit field. The master sample point can be delayed by one or two system clock cycles.

Table 11-24. Delayed master sample point

SMPL_PT	Number of system clock cycles between odd-numbered edge of SCK and sampling of SIN
00	0
01	1
10	2
11	Invalid value

Figure 11-16 shows the modified transfer format for CPHA = 0. Only the condition where CPOL = 0 is illustrated. The delayed master sample points are indicated with a lighter shaded arrow.

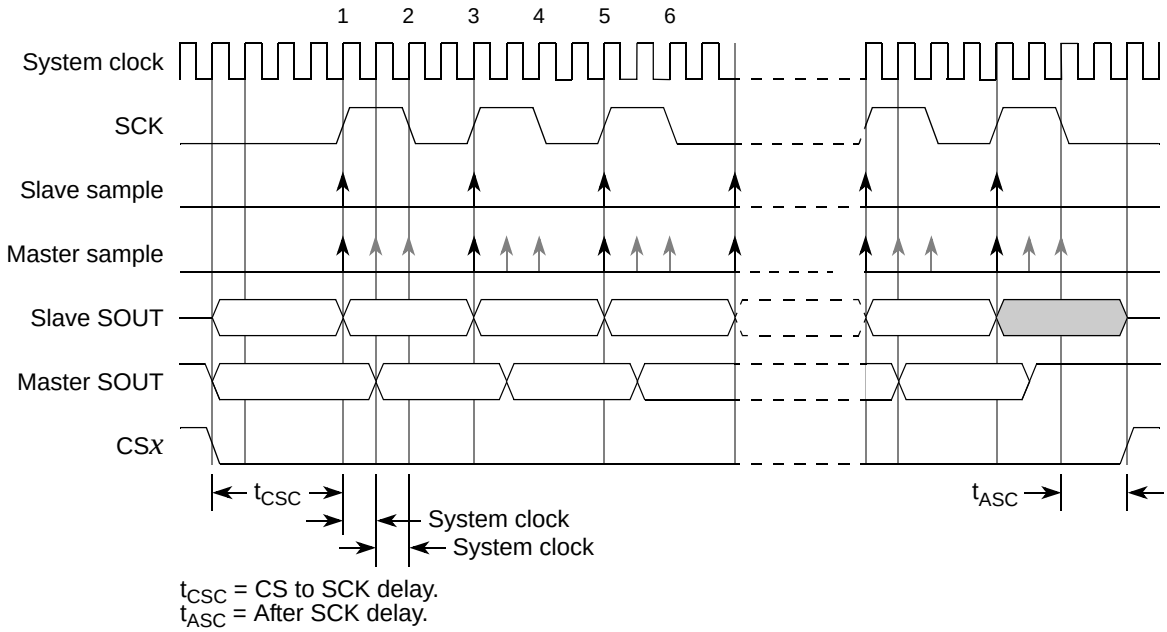


Figure 11-16. DSPI modified transfer format (MTFE = 1, CPHA = 0, $f_{SCK} = f_{SYS} / 4$)

11.8.5.4 Modified SPI transfer format (MTFE = 1, CPHA = 1)

At the start of a transfer the DSPI asserts the CS signal to the slave device. After the CS to SCK delay has elapsed, the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even-numbered edges of SCK. The master samples the slave SOUT signal on the odd-numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge is visible on the master SCK pin during the sampling of the last bit. The SCK to CS delay must be greater or equal to half of the SCK period.

NOTE

For the modified transfer format to operate correctly, analyze the SPI link timing budget thoroughly.

Figure 11-17 shows the modified transfer format for CPHA = 1. Only the condition where CPOL = 0 is described.

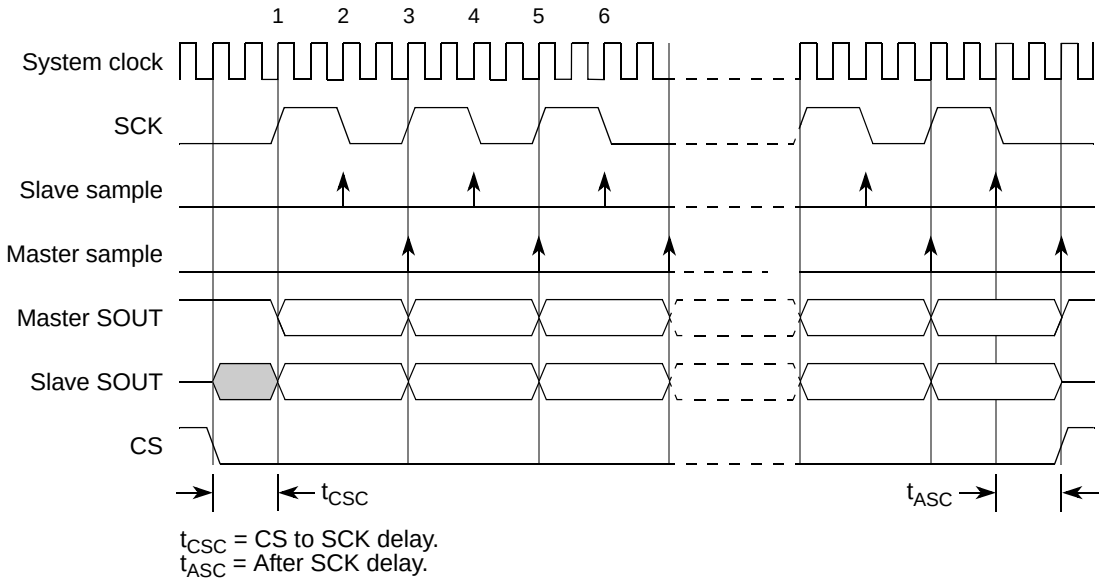


Figure 11-17. DSPI modified transfer format (MTFE = 1, CPHA = 1, $f_{SCK} = f_{SYS} / 4$)

11.8.5.5 Continuous selection format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The continuous selection format provides the flexibility to handle both cases. The continuous selection format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the DSPI drives the asserted chip select signals to their idle states in between frames. The idle states of the chip select signals are selected by the PCSIS field in the DSPI*x*_MCR.

Figure 11-18 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 0.

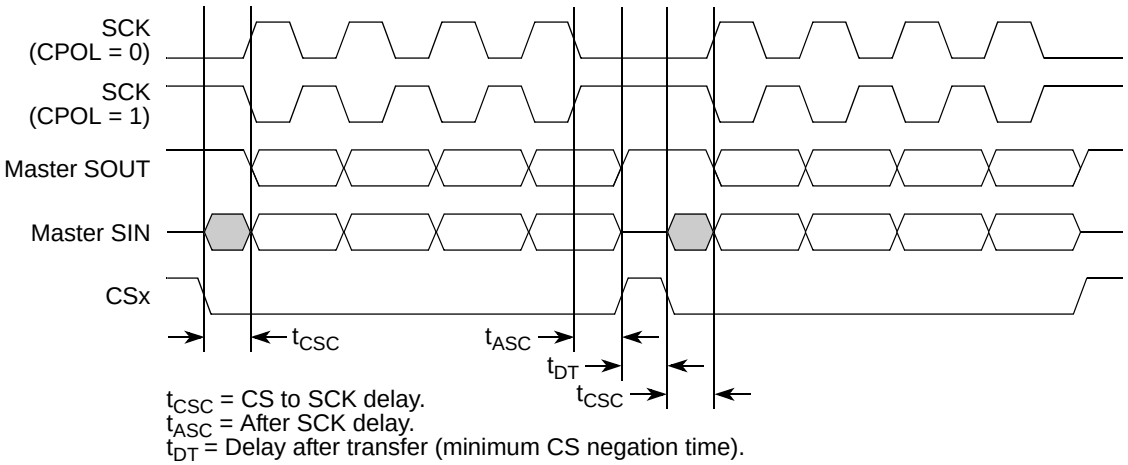


Figure 11-18. Example of non-continuous format (CPHA = 1, CONT = 0)

When the CONT = 1 and the CS signal for the next transfer is the same as for the current transfer, the CS signal remains asserted for the duration of the two transfers. The delay between transfers (t_{DT}) is not inserted between the transfers.

Figure 11-19 shows the timing diagram for two 4-bit transfers with CPHA = 1 and CONT = 1.

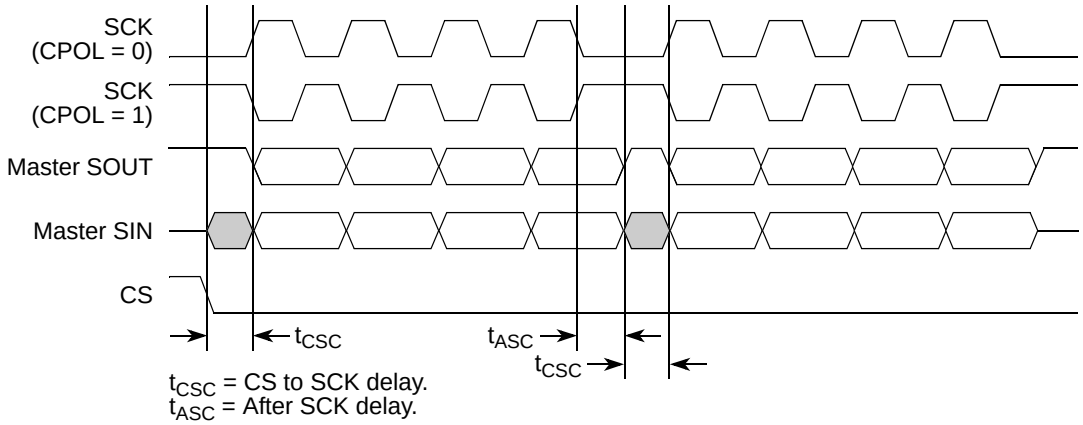


Figure 11-19. Example of continuous transfer (CPHA = 1, CONT = 1)

In Figure 11-19, the period length at the start of the next transfer is the sum of t_{ASC} and t_{CSC}. It does not include a half-clock period. The default settings for these provide a total of four system clocks. In many situations, t_{ASC} and t_{CSC} must be increased if a full half-clock period is required.

When the CONT bit = 1 and the CS signals for the next transfer are different from the present transfer, the CS signals behave as if the CONT bit was not set.

Switching CTAR registers or changing which PCS signals are asserted between frames while using Continuous Selection can cause errors in the transfer. The PCS signal should be negated before CTAR is switched or different PCS signals are selected.

11.8.5.6 Clock polarity switching between DSPI transfers

If you want to switch polarity between non-continuous DSPI frames, it is important to remember that the edge generated by the change in the idle state of the clock occurs one system clock before the assertion of the chip select for the next frame.

NOTE

It is mandatory to fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty. For example, while transmitting in Master mode, it should be ensured that the last entry in the TX FIFO, after which TX FIFO becomes empty, must have the CONT bit in command frame as deasserted (CONT bit = 0). While operating in Slave mode, it should be ensured that when the last entry in the TX FIFO is completely transmitted (that is, the corresponding TCF flag is asserted and TX FIFO is empty), the slave should be de-selected for any further serial communication. Otherwise, an underflow error occurs.

Refer to Section 11.7.2.3, DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn).

In Figure 11-20, time ‘A’ shows the one clock interval. Time ‘B’ is user programmable from a minimum of two system clocks.

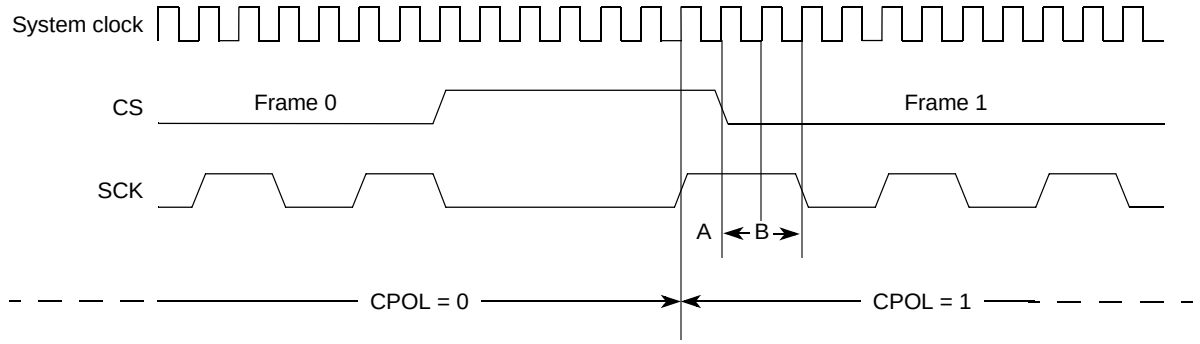


Figure 11-20. Polarity switching between frames

11.8.6 Continuous serial communications clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the DSPI*x*_MCR. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Setting CPHA = 0 is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for modified transfer format.

Clock and transfer attributes for the continuous SCK mode are set according to the following rules:

- The TX FIFO must be cleared before initiating any SPI configuration transfer.
- When the DSPI is in SPI configuration, CTAR0 shall be used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame should be CTAR0.
- In all configurations, the currently selected CTAR shall remain in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.
- To ensure PCS stability during data transmission in Continuous Selection Format and with Continuous SCK clock enabled, make sure that the data with reset CONT bit is written to DSPI_PUSHR register before the previous data sub-frame (with CONT bit set) transfer is over.
- If multiple CTARs are used in Continuous Peripheral Chip Select mode, ensure that the following conditions are met: if DSPI*x*_CTAR*n*[CPHA] = 1 and DSPI*x*_MCR[CONT_SCKE] = 0, then ensure that DSPIx_CTAR*n*[CPOL, CPHA, PCSSCK, or PBR] bits do not change between frames. If DSPI*x*_CTAR*n*[CPHA] = 0 or DSPI*x*_MCR[CONT_SCKE] = 1, then ensure that no bit field of DSPI*x*_CTAR*n* changes between frames except DSPI*x*_CTAR*n*[PBR].
- When in Continuous SCK mode, CTAR0 should always be used for the SPI transfer, and the TX FIFO must be clear using the MCR[CLR_TXF] field before initiating transfer.

The device is designed to use the same baud rate for all transfers made while using the continuous SCK. Switching clock polarity between frames while using continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into Module Disable mode.

Enabling continuous SCK disables the CS to SCK delay and the after SCK delay. The delay after transfer is fixed at one SCK cycle. Figure 11-21 shows timing diagram for continuous SCK format with continuous selection disabled.

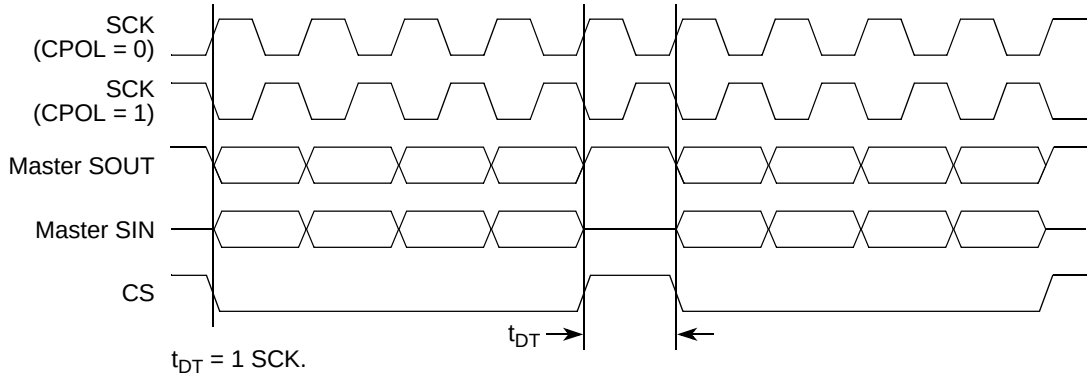


Figure 11-21. Continuous SCK timing diagram (CONT= 0)

If the CONT bit in the TX FIFO entry is set, CS remains asserted between the transfers when the CS signal for the next transfer is the same as for the current transfer. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO
- Continuous SCK with CONT bit set and entering Stopped state (refer to Section 11.8.2, Start and stop of DSPI transfers)
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode

Figure 11-22 shows timing diagram for continuous SCK format with continuous selection enabled.

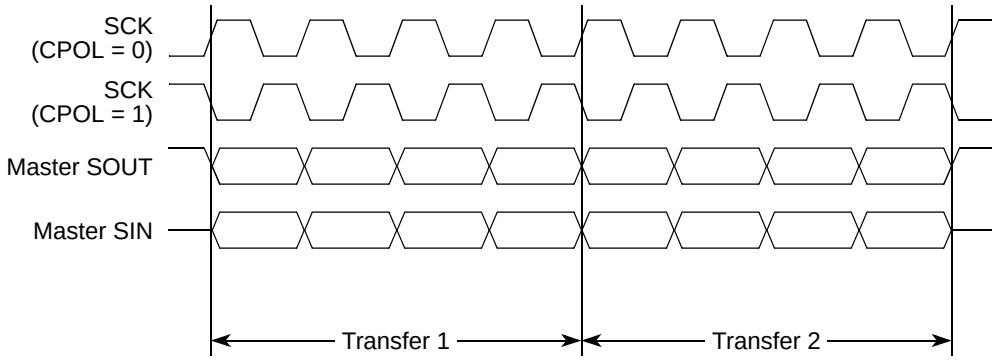


Figure 11-22. Continuous SCK timing diagram (CONT=1)

11.8.7 Interrupts/DMA requests

The DSPI has five conditions that can generate interrupt requests only, and two conditions that can generate interrupt.

Table 11-25 lists the seven conditions.

Table 11-25. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of transfer queue has been reached (EOQ)	EOQF	X	
TX FIFO is not full	TFFF	X	X
Current frame transfer is complete	TCF	X	
TX FIFO underflow has occurred	TFUF	X	
RX FIFO is not empty	RFDF	X	X
RX FIFO overflow occurred	RFOF	X	
A FIFO overrun occurred[1]	TFUF ORed with RFOF	X	

[1] The FIFO overrun condition is created by ORing the TFUF and RFOF flags together.

Each condition has a flag bit and a request enable bit. The flag bits are described in Section 11.7.2.4, DSPI Status Register (DSPIx_SR), and the request enable bits are described in Section 11.7.2.5, DSPI DMA / Interrupt Request Select and Enable Register (DSPIx_RSER). The TX FIFO fill flag (TFFF) and RX FIFO drain flag (RFDF) generate interrupt requests or DMA requests, depending on the TFFF_DIRS and RFDF_DIRS bits in the DSPI*x*_RSER.

11.8.7.1 End of queue interrupt request (EOQF)

The end of queue request indicates that the end of a transmit queue has been reached. The end of queue request is generated when the EOQ bit in the executing SPI command is asserted and the EOQF_RE bit in the DSPI*x*_RSER is set. Refer to the EOQ bit description in Section 11.7.2.4, DSPI Status Register (DSPIx_SR). Refer to Figure 11-14 and Figure 11-15 that illustrate when EOQF is set.

11.8.7.2 Transmit FIFO fill interrupt or DMA request (TFFF)

The transmit FIFO fill request indicates that the TX FIFO is not full. The transmit FIFO fill request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the DSPI*x*_RSER is set. The TFFF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

11.8.7.3 Transfer complete interrupt request (TCF)

The transfer complete request indicates the end of the transfer of a serial frame. The transfer complete request is generated at the end of each frame transfer when the TCF_RE bit is set in the DSPI*x*_RSER. Refer to the TCF bit description in Section 11.7.2.4, DSPI Status Register (DSPIx_SR). Refer to Figure 11-14 and Figure 11-15 that illustrate when TCF is set.

11.8.7.4 Transmit FIFO underflow interrupt request (TFUF)

The transmit FIFO underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in Slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in Slave mode and SPI configuration is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the DSPI*x*_RSER is set, an interrupt request is generated.

11.8.7.5 Receive FIFO drain interrupt or DMA request (RFDF)

The receive FIFO drain request indicates that the RX FIFO is not empty. The receive FIFO drain request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the DSPI*x*_RSER is set. The RFDF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

11.8.7.6 Receive FIFO overflow interrupt request (RFOF)

The receive FIFO overflow request indicates that an overflow condition in the RX FIFO has occurred. A receive FIFO overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the DSPI*x*_RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is either ignored or shifted into the shift register. If the ROOE bit is set, the incoming data is shifted into the shift register. If the ROOE bit is negated, the incoming data is ignored.

11.8.7.7 FIFO overrun request (TFUF) or (RFOF)

The FIFO overrun request indicates that at least one of the FIFOs in the DSPI has exceeded its capacity. The FIFO overrun request is generated by logically ORing together the RX FIFO overflow and TX FIFO underflow signals.

11.8.8 Power-saving features

The DSPI supports three power-saving strategies:

- External stop mode
- Module Disable mode—clock gating of non-memory mapped logic
- Clock gating of slave interface signals and clock to memory-mapped logic

11.8.8.1 External Stop mode

The DSPI supports the Stop mode protocol. When a request is made to enter External Stop mode, the DSPI block acknowledges the request by negating ipg_stop_ack. When the DSPI is ready to have its clocks shut off, the ipg_stop_ack signal is asserted. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it asserts ipg_stop_ack. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode. Implementation of IPI Green Line Stop mode in an SoC is optional.

11.8.8.2 Module Disable mode

Module Disable mode is a module-specific mode that the DSPI can enter to save power. Host software can initiate the Module Disable mode by writing a 1 to the MDIS bit in the DSPI*x*_MCR. In Module Disable mode, the DSPI is in a dormant state, but the memory-mapped registers are still accessible. Certain read or write operations have a different effect when the DSPI is in Module Disable mode. Reading the RX FIFO pop register does not change the state of the RX FIFO. Likewise, writing to the TX FIFO push register does not change the state of the TX FIFO. Clearing either of the FIFOs does not have any effect in Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the DSPI*x*_MCR do not have any effect in Module Disable mode. In Module Disable mode, all status bits and register flags in the DSPI return the correct values when read, but writing to them has no effect. Writing to the DSPI*x*_TCR during Module Disable mode does not have an effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

11.8.8.3 Slave interface signal gating

The DSPI module enable signal is used to gate slave interface signals such as address, byte enable, read/write, and data. This prevents toggling slave interface signals from consuming power unless the DSPI is accessed.

11.9 Initialization and application information

11.9.1 How to change queues

DSPI queues are not part of the DSPI module, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI configuration. This section presents an example of how to change queues for the DSPI.

1. The last command word from a queue is executed. The EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the DSPI*x*_SR is set.
3. The setting of the EOQF flag disables both serial transmission and serial reception of data, putting the DSPI in the Stopped state. The TXRXS bit is negated to indicate the Stopped state.
4. The eDMA continues to fill the TX FIFO until it is full or until step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the eDMA controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in DSPI*x*_SR or by checking RFDF in the DSPI*x*_SR after each read operation of the DSPI*x*_POPR.
7. Modify DMA descriptor of TX and RX channels for new queues.
8. Flush TX FIFO by writing a 1 to the CLR_TXF bit in the DSPI*x*_MCR register and flush the RX FIFO by writing a 1 to the CLR_RXF bit in the DSPI*x*_MCR register.

9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the DSPI*x*_TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

11.9.2 Baud rate settings

Table 11-26 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the DSPI*x*_CTARs. The values are calculated at a 64 MHz system frequency with DBR = 0.

Table 11-26. Baud rate values

		Baud rate divider prescaler values (DSPI_CTAR[PBR])			
		2	**3**	**5**	**7**
Baud Rate Scaler Values (DSPI_CTAR[BR])	**2**	16.0 MHz	10.7 MHz	6.4 MHz	4.57 MHz
	4	8 MHz	5.33 MHz	3.2 MHz	2.28 MHz
	6	5.33 MHz	3.56 MHz	2.13 MHz	1.52 MHz
	8	4 MHz	2.67 MHz	1.60 MHz	1.15 MHz
	16	2 MHz	1.33 MHz	800 kHz	571 kHz
	32	1 MHz	670 kHz	400 kHz	285 kHz
	64	500 kHz	333 kHz	200 kHz	142 kHz
	128	250 kHz	166 kHz	100 kHz	71.7 kHz
	256	125 kHz	83.2 kHz	50 kHz	35.71 kHz
	512	62.5 kHz	41.6 kHz	25 kHz	17.86 kHz
	1024	31.2 kHz	20.8 kHz	12.5 kHz	8.96 kHz
	2048	15.6 kHz	10.4 kHz	6.25 kHz	4.47 kHz
	4096	7.81 kHz	5.21 kHz	3.12 kHz	2.23 kHz
	8192	3.90 kHz	2.60 kHz	1.56 kHz	1.11 kHz
	16384	1.95 kHz	1.31 kHz	781 Hz	558 Hz
	32768	979 Hz	653 Hz	390 Hz	279 Hz

11.9.3 Delay settings

Table 11-27 shows the values for the delay after transfer (t_{DT}) and CS to SCK delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the DSPI*x*_CTARs. The values calculated assume a 100 MHz system frequency.

Table 11-27. Delay values

		Delay prescaler values (DSPI_CTAR[PBR])			
		1	**3**	**5**	**7**
Delay scaler values (DSPI_CTAR[DT])	**2**	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

11.9.4 Calculation of FIFO pointer addresses

The user has complete visibility of the TX and RX FIFO contents through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped. For the TX FIFO the first-in pointer is the transmit next pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the pop next pointer (POPNXTPTR).

Refer to Section 11.8.3.4, transmit First In First Out (TX FIFO) buffering mechanism, and Section 11.8.3.5, Receive First In First Out (RX FIFO) buffering mechanism, for details on the FIFO operation. The TX FIFO is chosen for the illustration, but the concepts carry over to the RX FIFO.

Figure 11-23 illustrates the concept of first-in and last-in FIFO entries along with the FIFO counter.

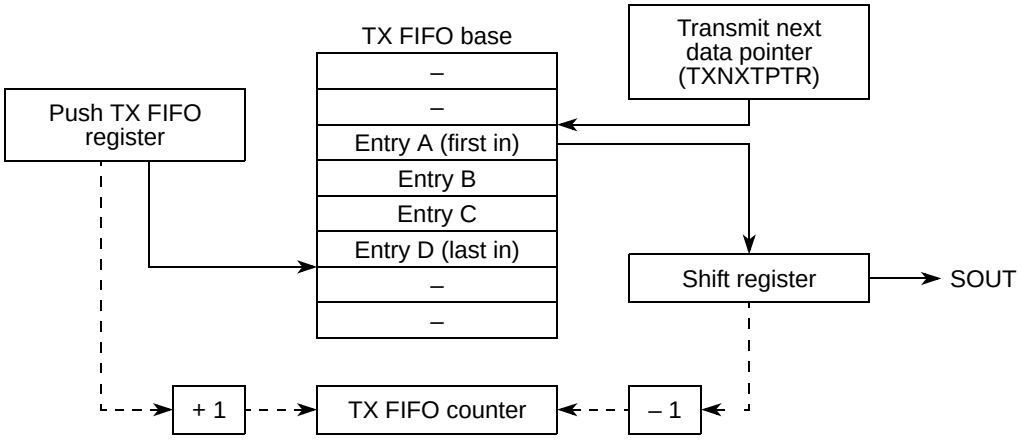


Figure 11-23. TX FIFO pointers and counter

11.9.4.1 Address calculation for the first-in entry and last-in entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

First-in entry address = TX FIFO base + 4 (TXNXTPTR)

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

Last-in entry address = TX FIFO base + 4 x [(TXCTR + TXNXTPTR – 1) modulo TX FIFO depth]

where:

TX FIFO base = base address of transmit FIFO
TXCTR = transmit FIFO counter
TXNXTPTR = transmit next pointer
TX FIFO depth = transmit FIFO depth, implementation specific

11.9.4.2 Address calculation for the first-in entry and last-in entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

First-in entry address = RXFIFO base + 4 x (POPNXTPTR)

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Last-in entry address = RXFIFO base + 4 x [(RXCTR + POPNXTPTR – 1) modulo RXFIFO depth]

where:

RXFIFO base = base address of receive FIFO
RXCTR = receive FIFO counter
POPNXTPTR = pop next pointer

RX FIFO depth = receive FIFO depth, implementation specific

Chapter 12
Display Control Unit (DCU)

12.1 Introduction

The Display Controller Unit (DCU) module is a system master that fetches graphics stored in internal or external memory and displays them on a TFT LCD panel. A wide range of panel sizes is supported and the timing of the interface signals is highly configurable. Graphics are read directly from memory and then blended in real-time, which allows for dynamic content creation with minimal CPU intervention. Graphics may be encoded in a variety of formats to optimize memory usage. The DCU also has the capability of displaying real-time video from an external video source.

12.1.1 Overview

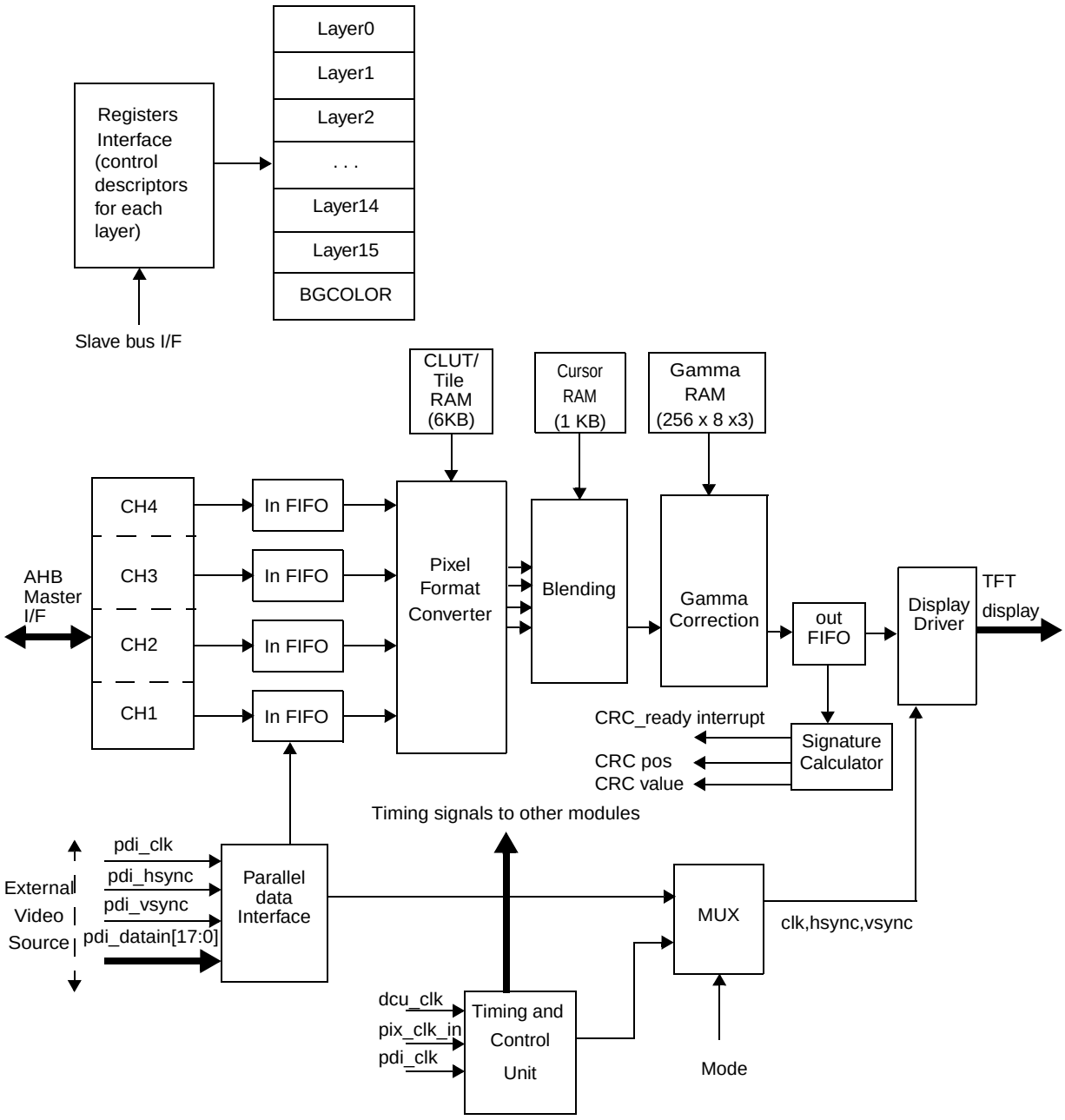


Figure 12-1. DCU block diagram

Figure 12-1 shows the DCU architecture. This comprises two distinct sections. The lower section shows the functional blocks of the DCU that fetch the graphic and video content and drive the TFT LCD panel. The upper section describes the user interface through which the user configures the graphical content of the TFT LCD panel.

The sections are analogous to the structure of communications modules, such as the FlexCAN, where one part of the module is configured to connect with the communications bus through bit-timing, parity, baud rate, etc., while a different part is used to store the data content and message identifiers.

The configuration of the lower section is dependent on the specific TFT LCD panel and optional real-time video hardware that are attached to the DCU inputs and outputs. In most cases, this is configured once for the hardware in use before the DCU is enabled. When active, this section automatically:

- Calculates the relevant graphical content for each pixel
- Fetches the source graphics from memory using its internal DMA channels (labelled CH1 to CH4)
- Converts the graphic value of each fetched pixel into full quality color format (if required)
- Calculates the required pixel value by blending the values of up to four separate graphics
- Performs a gamma correction on the pixel value (if required)
- Sends the pixel value to the TFT LCD display over its data bus
- Sets flags to indicate end of frame, buffer threshold, and other status changes

The upper section describes the characteristics of the graphics to be displayed on the panel and how they are blended together. The DCU manages the graphical content of the panel through sets of registers called layers. There are 16 layers available in the DCU and each contains the following information:

- Horizontal and vertical size of graphic
- Position of graphic on the panel
- Address of graphic in memory
- Color encoding format and color palettes (if required)
- Type and depth of blending
- Range of colors identified for chroma blending
- Tile size

The values in these registers may be changed at any time, and the panel content will be updated when the next full frame is ready to be displayed. The layers are set to a fixed priority, and this is used by the lower section to define which layers are blended, in which order, on the panel.

The upper section also contains configuration registers for a cursor graphic, the default background color, interrupt enables, test graphic, and simple register protection settings.

12.1.2 Features

The DCU has these features:

- Full RGB888 output to TFT LCD panel
- 16 graphics layers, a default background color layer and a cursor layer with integrated blinking option
- Blending of each pixel using up to 4 source layers dependent on size of panel
- Programmable panel size up to a maximum of Wide VGA (800 x 480)
- Gamma correction with 8-bit resolution on each color component
- Safety mode for tagging pixels on highest priority layers

- Digital video input with and without sync extraction per ITU-R BT.656 supporting multiple video input formats including RGB666, RGB565, monochrome and YCbCr422
- Dedicated memory blocks to store a cursor and Color Look Up Tables (CLUTs)

Each graphic layer has the following attributes

- Can be placed with one pixel resolution in either axis
- Supports multiple color-encoding formats including 1, 2, 4, and 8 bits per pixel indexed colors, RGB565 and RGB888 direct colors, and ARGB1555, ARGB4444 and ARGB8888 direct colors with an alpha channel
- Alpha blending with 8-bit resolution
- Chroma-key blending for anti-mask encoding
- Multiple alpha and chroma-key blending modes
- Transparency modes for anti-aliased text and graphics
- Luminance mode for highlighting content
- Tile mode for efficient creation of textured background content

12.1.3 Modes of operation

The DCU has four modes of operation:

- **DCU_OFF**: When in this mode, the DCU is turned off. All the logic in the design is put in reset state to reduce power.
- **NORMAL_MODE**: The DCU displays and blends the graphics specified by the layer descriptors.
- **PDI_MODE**: A mode which fetches video from an external video source and combines that with the graphics configured on the layers.
- **COLBAR_MODE**: Color bar generation for testing purposes.

12.2 External signal description

12.2.1 Overview

The DCU has up to 22 input signals and up to 30 output signals. See Figure 12-2. The choice of signals used depends on the configuration of the DCU. All active signals must be enabled by configuring the appropriate PCR registers in the SIUL module.

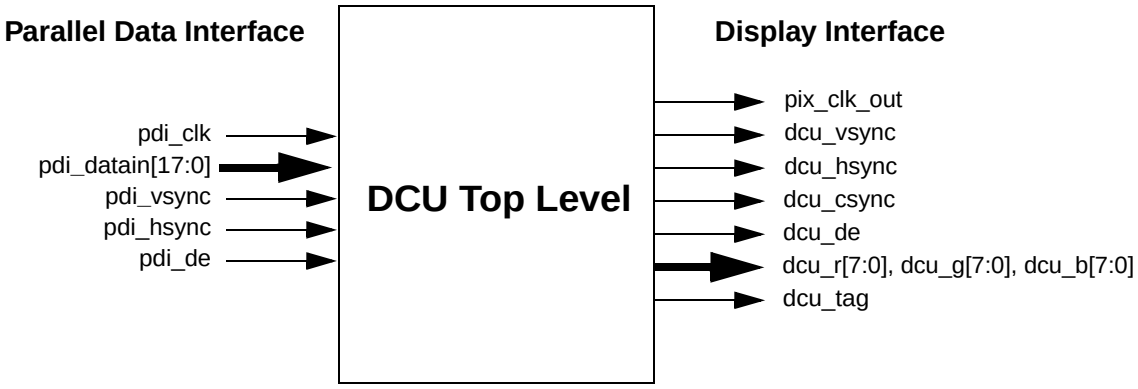


Figure 12-2. External signals

12.2.2 Detailed signal descriptions

Table 12-1. Detailed signal descriptions

Signal	Direction	Description
Parallel Data Interface (Camera Interface)		
pdi_clk	IN	Clock for the parallel data from the input video data
pdi_vsync	IN	Vertical sync to indicate the start of new frame for the display
pdi_hsync	IN	Horizontal sync to indicate the start of new line for the display
pdi_de	IN	Data Enable for the camera data input
pdi_datain[17:0]	IN	18-bit parallel input data for the display
Display Interface		
pix_clk_out	OUT	Pixel clock used to drive the display panel
dcu_vsync	OUT	Vertical sync signal, indicating the beginning of a new frame
dcu_hsync	OUT	Horizontal sync signal, indicating the beginning of a new line
dcu_csync	OUT	Composite Sync Signal, combining horizontal and vertical sync signals to form a composite sync signal. It includes both the HSYNC pulse and the VSYNC pulse. **Note:** Not used on this device.
dcu_tag	OUT	When high, this signal indicates that the pixel is tagged and an application can calculate CRC externally on this pixel.
dcu_de	OUT	Data Enable. Qualifies the data output (dcu_ld)
dcu_r[7:0], dcu_g[7:0], dcu_b[7:0]	OUT	Red, green and blue data output.

12.3 Memory map and register definition

12.3.1 Memory map

Table 12-2 shows the memory map of the DCU.

Table 12-2. DCU memory map

Parameter	Address range
Register address space	0x0000 – 0x03FF
Cursor address space	0x0400 – 0x07FF
Gamma_R address space	0x0800 – 0x0BFF
Gamma_G address space	0x0C00 – 0x0FFF
Gamma_B address space	0x1000 – 0x13FF
Empty space	0x1400 – 0x1FFF
CLUT/Tile address space	0x2000 – 0x3FFF

12.3.2 Register map

Table 12-3 provides the register map of the DCU.

Only 32-bit writes and 32-bit aligned access are supported. Byte and half-word accesses are not supported.

Table 12-3. DCU register map

Address offset	Register	Access	Reset value	Location
General registers				
0x000	CtrlDescL0_1 Register	R/W	0x00000000	on page 353
0x004	CtrlDescL0_2 Register	R/W	0x00000000	on page 354
0x008	CtrlDescL0_3 Register	R/W	0x00000000	on page 355
0x00C	CtrlDescL0_4 Register	R/W	0x00000000	on page 356
0x010	CtrlDescL0_5 Register	R/W/	0x00000000	on page 358
0x014	CtrlDescL0_6 Register	R/W/	0x00000000	on page 359
0x018	CtrlDescL0_7 Register	R/W/	0x00000000	on page 361
0x01C	CtrlDescL1_1 Register	R/W	0x00000000	on page 353
0x020	CtrlDescL1_2 Register	R/W	0x00000000	on page 354
0x024	CtrlDescL1_3 Register	R/W	0x00000000	on page 355
0x028	CtrlDescL1_4 Register	R/W	0x00000000	on page 356
0x02C	CtrlDescL1_5 Register	R/W	0x00000000	on page 358
0x030	CtrlDescL1_6 Register	R/W	0x00000000	on page 359
0x034	CtrlDescL1_7 Register	R/W	0x00000000	on page 361

Table 12-3. DCU register map (continued)

Address offset	Register	Access	Reset value	Location
0x038	CtrlDescL2_1 Register	R/W	0x00000000	on page 353
0x03C	CtrlDescL2_2 Register	R/W	0x00000000	on page 354
0x040	CtrlDescL2_3 Register	R/W	0x00000000	on page 355
0x044	CtrlDescL2_4 Register	R/W	0x00000000	on page 356
0x048	CtrlDescL2_5 Register	R/W	0x00000000	on page 358
0x04C	CtrlDescL2_6 Register	R/W	0x00000000	on page 359
0x050	CtrlDescL2_7 Register	R/W	0x00000000	on page 361
0x054	CtrlDescL3_1 Register	R/W	0x00000000	on page 353
0x058	CtrlDescL3_2 Register	R/W	0x00000000	on page 354
0x05C	CtrlDescL3_3 Register	R/W	0x00000000	on page 355
0x060	CtrlDescL3_4 Register	R/W	0x00000000	on page 356
0x064	CtrlDescL3_5 Register	R/W	0x00000000	on page 358
0x068	CtrlDescL3_6 Register	R/W	0x00000000	on page 359
0x06C	CtrlDescL3_7 Register	R/W	0x00000000	on page 361
0x070	CtrlDescL4_1 Register	R/W	0x00000000	on page 353
0x074	CtrlDescL4_2 Register	R/W	0x00000000	on page 354
0x078	CtrlDescL4_3 Register	R/W	0x00000000	on page 355
0x07C	CtrlDescL4_4 Register	R/W	0x00000000	on page 356
0x080	CtrlDescL4_5 Register	R/W	0x00000000	on page 358
0x084	CtrlDescL4_6 Register	R/W	0x00000000	on page 359
0x088	CtrlDescL4_7 Register	R/W	0x00000000	on page 361
0x08C	CtrlDescL5_1 Register	R/W	0x00000000	on page 353
0x090	CtrlDescL5_2 Register	R/W	0x00000000	on page 354
0x094	CtrlDescL5_3 Register	R/W	0x00000000	on page 355
0x098	CtrlDescL5_4 Register	R/W	0x00000000	on page 356
0x09C	CtrlDescL5_5 Register	R/W	0x00000000	on page 358
0x0A0	CtrlDescL5_6 Register	R/W	0x00000000	on page 359
0x0A4	CtrlDescL5_7 Register	R/W	0x00000000	on page 361
0x0A8	CtrlDescL6_1 Register	R/W	0x00000000	on page 353
0x0AC	CtrlDescL6_2 Register	R/W	0x00000000	on page 354
0x0B0	CtrlDescL6_3 Register	R/W	0x00000000	on page 355
0x0B4	CtrlDescL6_4 Register	R/W	0x00000000	on page 356
0x0B8	CtrlDescL6_5 Register	R/W	0x00000000	on page 358

Table 12-3. DCU register map (continued)

Address offset	Register	Access	Reset value	Location
0x0BC	CtrlDescL6_6 Register	R/W	0x00000000	on page 359
0x0C0	CtrlDescL6_7 Register	R/W	0x00000000	on page 361
0x0C4	CtrlDescL7_1 Register	R/W	0x00000000	on page 353
0x0C8	CtrlDescL7_2 Register	R/W	0x00000000	on page 354
0x0CC	CtrlDescL7_3 Register	R/W	0x00000000	on page 355
0x0D0	CtrlDescL7_4 Register	R/W	0x00000000	on page 356
0x0D4	CtrlDescL7_5 Register	R/W	0x00000000	on page 358
0x0D8	CtrlDescL7_6 Register	R/W	0x00000000	on page 359
0x0DC	CtrlDescL7_7 Register	R/W	0x00000000	on page 361
0x0E0	CtrlDescL8_1 Register	R/W	0x00000000	on page 353
0x0E4	CtrlDescL8_2 Register	R/W	0x00000000	on page 354
0x0E8	CtrlDescL8_3 Register	R/W	0x00000000	on page 355
0x0EC	CtrlDescL8_4 Register	R/W	0x00000000	on page 356
0x0F0	CtrlDescL8_5 Register	R/W	0x00000000	on page 358
0x0F4	CtrlDescL8_6 Register	R/W	0x00000000	on page 359
0x0F8	CtrlDescL8_7 Register	R/W	0x00000000	on page 361
0x0FC	CtrlDescL9_1 Register	R/W	0x00000000	on page 353
0x100	CtrlDescL9_2 Register	R/W	0x00000000	on page 354
0x104	CtrlDescL9_3 Register	R/W	0x00000000	on page 355
0x108	CtrlDescL9_4 Register	R/W	0x00000000	on page 356
0x10C	CtrlDescL9_5 Register	R/W	0x00000000	on page 358
0x110	CtrlDescL9_6 Register	R/W	0x00000000	on page 359
0x114	CtrlDescL9_7 Register	R/W	0x00000000	on page 361
0x118	CtrlDescL10_1 Register	R/W	0x00000000	on page 353
0x11C	CtrlDescL10_2 Register	R/W	0x00000000	on page 354
0x120	CtrlDescL10_3 Register	R/W	0x00000000	on page 355
0x124	CtrlDescL10_4 Register	R/W	0x00000000	on page 356
0x128	CtrlDescL10_5 Register	R/W	0x00000000	on page 358
0x12C	CtrlDescL10_6 Register	R/W	0x00000000	on page 359
0x130	CtrlDescL10_7 Register	R/W	0x00000000	on page 361
0x134	CtrlDescL11_1 Register	R/W	0x00000000	on page 353
0x138	CtrlDescL11_2 Register	R/W	0x00000000	on page 354
0x13C	CtrlDescL11_3 Register	R/W	0x00000000	on page 355

Table 12-3. DCU register map (continued)

Address offset	Register	Access	Reset value	Location
0x140	CtrlDescL11_4 Register	R/W	0x00000000	on page 356
0x144	CtrlDescL11_5 Register	R/W	0x00000000	on page 358
0x148	CtrlDescL11_6 Register	R/W	0x00000000	on page 359
0x14C	CtrlDescL11_7 Register	R/W	0x00000000	on page 361
0x150	CtrlDescL12_1 Register	R/W	0x00000000	on page 353
0x154	CtrlDescL12_2 Register	R/W	0x00000000	on page 354
0x158	CtrlDescL12_3 Register	R/W	0x00000000	on page 355
0x15C	CtrlDescL12_4 Register	R/W	0x00000000	on page 356
0x160	CtrlDescL12_5 Register	R/W	0x00000000	on page 358
0x164	CtrlDescL12_6 Register	R/W	0x00000000	on page 359
0x168	CtrlDescL12_7 Register	R/W	0x00000000	on page 361
0x16C	CtrlDescL13_1 Register	R/W	0x00000000	on page 353
0x170	CtrlDescL13_2 Register	R/W	0x00000000	on page 354
0x174	CtrlDescL13_3 Register	R/W	0x00000000	on page 355
0x178	CtrlDescL13_4 Register	R/W	0x00000000	on page 356
0x17C	CtrlDescL13_5 Register	R/W	0x00000000	on page 358
0x180	CtrlDescL13_6 Register	R/W	0x00000000	on page 359
0x184	CtrlDescL13_7 Register	R/W	0x00000000	on page 361
0x188	CtrlDescL14_1 Register	R/W	0x00000000	on page 353
0x18C	CtrlDescL14_2 Register	R/W	0x00000000	on page 354
0x190	CtrlDescL14_3 Register	R/W	0x00000000	on page 355
0x194	CtrlDescL14_4 Register	R/W	0x00000000	on page 356
0x198	CtrlDescL14_5 Register	R/W	0x00000000	on page 358
0x19C	CtrlDescL14_6 Register	R/W	0x00000000	on page 359
0x1A0	CtrlDescL14_7 Register	R/W	0x00000000	on page 361
0x1A4	CtrlDescL15_1 Register	R/W	0x00000000	on page 353
0x1A8	CtrlDescL15_2 Register	R/W	0x00000000	on page 354
0x1AC	CtrlDescL15_3 Register	R/W	0x00000000	on page 355
0x1B0	CtrlDescL15_4 Register	R/W	0x00000000	on page 356
0x1B4	CtrlDescL15_5 Register	R/W	0x00000000	on page 358
0x1B8	CtrlDescL15_6 Register	R/W	0x00000000	on page 359
0x1BC	CtrlDescL15_7 Register	R/W	0x00000000	on page 361
0x1C0	CtrlDescCursor_1 Register	R/W	0x00000000	on page 361

Table 12-3. DCU register map (continued)

Address offset	Register	Access	Reset value	Location
0x1C4	CtrlDescCursor_2 Register	R/W	0x00000000	on page 362
0x1C8	CtrlDescCursor_3 Register	R/W	0x00000000	on page 363
0x1CC	CtrlDescCursor_4 Register	R/W	0x00000000	on page 363
0x1D0	DCU_MODE Register	R/W	0x00000000	on page 364
0x1D4	BGND Register	R/W	0x00000000	on page 366
0x1D8	DISP_SIZE Register	R/W	0x00000000	on page 367
0x1DC	HSYN_PARA Register	R/W	0x00C01803	on page 368
0x1E0	VSYN_PARA Register	R/W	0x00C01803	on page 368
0x1E4	SYNPOL Register	R/W	0x00000000	on page 369
0x1E8	THRESHOLD Register	R/W	0x0000780A	on page 370
0x1EC	INT_STATUS Register	R	0x00000000	on page 371
0x1F0	INT_MASK Register	R/W	0x000F7FFF	on page 373
0x1F4	COLBAR_1 Register	R/W	0xFF000000	on page 376
0x1F8	COLBAR_2 Register	R/W	0xFF0000FF	on page 376
0x1FC	COLBAR_3 Register	R/W	0xFF00FFFF	on page 377
0x200	COLBAR_4 Register	R/W	0xFF00FF00	on page 377
0x204	COLBAR_5 Register	R/W	0xFFFFFF00	on page 378
0x208	COLBAR_6 Register	R/W	0xFFFF0000	on page 378
0x20C	COLBAR_7 Register	R/W	0xFFFF00FF	on page 379
0x210	COLBAR_8 Register	R/W	0xFFFFFFFF	on page 379
0x214	DIV_RATIO Register	R/W	0x0000001F	on page 379
0x218	SIGN_CALC_1 Register	R/W	0x00000000	on page 380
0x21C	SIGN_CALC_2 Register	R/W	0x00000000	on page 381
0x220	CRC_VAL Register	R/W	0x00000000	on page 381
0x224	PDI_STATUS Register	R/W	0x00000000	on page 382
0x228	Mask_PDI_STATUS Register	R/W	0x000003FF	on page 383
0x22C	PARR_ERR_STATUS Register	R/W	0x00000000	on page 384
0x230	Mask_PARR_ERR_STATUS Register	R/W	0x0007FFFF	on page 387
0x234	THRESHOLD_INP_BUF_1 Register	R/W	0xFF00FF00	on page 389
0x238	THRESHOLD_INP_BUF_2 Register	R/W	0xFF00FF00	on page 389
0x23C	LUMA_COMP Register	R/W	0x9512A254	on page 390
0x240	CHROMA_RED Register	R/W	0x03310000	on page 391
0x244	CHROMA_GREEN Register	R/W	0x06600F38	on page 391

Table 12-3. DCU register map (continued)

Address offset	Register	Access	Reset value	Location
0x248	CHROMA_BLUE Register	R/W	0x00000409	on page 392
0x24C	CRC_POS Register	R/W	0x00000000	on page 393
0x250	FG0_fcolor Register	R/W	0x00000000	on page 393
0x254	FG0_bcolor Register	R/W	0x00000000	on page 394
0x258	FG1_fcolor Register	R/W	0x00000000	on page 393
0x25C	FG1_bcolor Register	R/W	0x00000000	on page 394
0x260	FG2_fcolor Register	R/W	0x00000000	on page 393
0x264	FG2_bcolor Register	R/W	0x00000000	on page 394
0x268	FG3_fcolor Register	R/W	0x00000000	on page 393
0x26C	FG3_bcolor Register	R/W	0x00000000	on page 394
0x270	FG4_fcolor Register	R/W	0x00000000	on page 393
0x274	FG4_bcolor Register	R/W	0x00000000	on page 394
0x278	FG5_fcolor Register	R/W	0x00000000	on page 393
0x27C	FG5_bcolor Register	R/W	0x00000000	on page 394
0x280	FG6_fcolor Register	R/W	0x00000000	on page 393
0x284	FG6_bcolor Register	R/W	0x00000000	on page 394
0x288	FG7_fcolor Register	R/W	0x00000000	on page 393
0x28C	FG7_bcolor Register	R/W	0x00000000	on page 394
0x290	FG8_fcolor Register	R/W	0x00000000	on page 393
0x294	FG8_bcolor Register	R/W	0x00000000	on page 394
0x298	FG9_fcolor Register	R/W	0x00000000	on page 393
0x29C	FG9_bcolor Register	R/W	0x00000000	on page 394
0x2A0	FG10_fcolor Register	R/W	0x00000000	on page 393
0x2A4	FG10_bcolor Register	R/W	0x00000000	on page 394
0x2A8	FG11_fcolor Register	R/W	0x00000000	on page 393
0x2AC	FG11_bcolor Register	R/W	0x00000000	on page 394
0x2B0	FG12_fcolor Register	R/W	0x00000000	on page 393
0x2B4	FG12_bcolor Register	R/W	0x00000000	on page 394
0x2B8	FG13_fcolor Register	R/W	0x00000000	on page 393
0x2BC	FG13_bcolor Register	R/W	0x00000000	on page 394
0x2C0	FG14_fcolor Register	R/W	0x00000000	on page 393
0x2C4	FG14_bcolor Register	R/W	0x00000000	on page 394
0x2C8	FG15_fcolor Register	R/W	0x00000000	on page 393

Table 12-3. DCU register map (continued)

Address offset	Register	Access	Reset value	Location
0x2CC	FG15_bcolor Register	R/W	0x00000000	on page 394
0x2D0–0x2FC	Reserved			
0x300	Global Protection Register	R/W	0x00000000	on page 395
0x304	Soft Lock Bit Register L0	R/W	0x00000000	on page 396
0x308	Soft Lock Bit Register L1	R/W	0x00000000	on page 397
0x30C	Soft Lock Bit Register DISP_SIZE	R/W	0x00000000	on page 399
0x310	Soft Lock Bit Register VSYNC/HSYNC PARA	R/W	0x00000000	on page 400
0x314	Soft Lock Bit Register POL	R/W	0x00000000	on page 401
0x318	Soft Lock Bit Register L0_TRANSP	R/W	0x00000000	on page 401
0x31C	Soft Lock Bit Register L1_TRANSP	R/W	0x00000000	on page 402

12.3.3 Register summary

Table 12-4. Register descriptions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CtrlDescL0_1 0x000	R	0	0	0	0	0	0	HEIGHT									
	W																
	R	0	0	0	0	WIDTH											
	W																
CtrlDescL0_2 0x004	R	0	0	0	0	0	0	POSY									
	W																
	R	0	0	0	0	POSX											
	W																
CtrlDescL0_3 0x008	R	ADDR															
	W																
	R																
	W																
CtrlDescL0_4 0x00C	R	EN	TILE_EN	DATA_SEL	SAFETY_EN	TRANS								BPP			
	W																
	R	0	LUOFFS											0	BB	AB	
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CtrlDescL0_5 0x010	R	0	0	0	0	0	0	0	0	CKMAX_R							
	W																
	R	CKMAX_G								CKMAX_B							
	W																
CtrlDescL0_6 0x014	R	0	0	0	0	0	0	0	0	CKMIN_R							
	W																
	R	CKMIN_G								CKMIN_B							
	W																
CtrlDescL0_7 0x018	R	0	0	0	0	0	0	TILE_VER_SIZE									
	W																
	R	0	0	0	0	0	0	0	0	TILE_HOR_SIZE							
	W																
CtrlDescCursor_1 0x1C0	R	0	0	0	0	0	0	HEIGHT									
	W																
	R	0	0	0	0	0	0	WIDTH									
	W																
CtrlDescCursor_2 0x1C4	R	0	0	0	0	0	0	POSY									
	W																
	R	0	0	0	0	0	0	POSX									
	W																
CtrlDescCursor_3 0x1C8	R	CUR_EN	0	0	0	0	0	0	0	DEFAULT_CURSOR_COLOR[0:7]							
	W																
	R	DEFAULT_CURSOR_COLOR[8:23]															
	W																
CtrlDescCursor_4 0x1CC	R	0	0	0	0	0	0	0	0	HWC_BLINK_OFF							
	W																
	R	0	0	0	0	0	0	0	EN_BLINK	HWC_BLINK_ON							
	W																

Table 12-4. Register descriptions (continued)

Name		0 / 16	1 / 17	2 / 18	3 / 19	4 / 20	5 / 21	6 / 22	7 / 23	8 / 24	9 / 25	10 / 26	11 / 27	12 / 28	13 / 29	14 / 30	15 / 31
DCU_MODE 0x1D0	R	DCU_SW_RESET	0	0	0	0	0	0	0	0	BLEND_ITER			PDI_SYNC_LOCK			
	W																
	R	PDI_INTERPOL_EN	RASTER_EN	PDI_EN	PDI_BYTE_REV	PDI_DE_MODE	PDI_NARROW_MODE	PDI_MODE		PDI_SLAVE_Mode	TAG_EN	SIG_EN	PDI_SYNC	0	EN_GAMMA	DCU_MODE	
	W																
BGND 0x1D4	R	0	0	0	0	0	0	0	0	BGND_R							
	W																
	R	BGND_G								BGND_B							
	W																
DISP_SIZE 0x1D8	R	0	0	0	0	0	0	DELTA_Y									
	W																
	R	0	0	0	0	0	0	0	0	DELTA_X							
	W																
HSYN_PARA 0x1DC	R	0	BP_H									0	0	PW_H[0:3]			
	W																
	R	PW_H[4:8]					0	0	FP_H								
	W																
VSYN_PARA 0x1E0	R	0	BP_V									0	0	PW_V[0:3]			
	W																
	R	PW_V[4:8]					0	0	FP_V								
	W																
SYN_POL 0x1E4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	INV_PDI_DE	INV_PDI_HS	INV_PDI_VS	INV_PDI_CLK	INV_PXCK	NEG	BP_VS	BP_HS	INV_CS	INV_VS	INV_HS
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
THRESHOLD 0x1E8	R	0	0	0	0	0	0	LS_BF_VS									
	W																
	R	OUT_BUF_HIGH								OUT_BUF_LOW							
	W																
INT_STATUS 0x1EC	R	0	0	0	0	0	0	0	0	0	0	0	0	P4_FIFO_HI_FLAG	P4_FIFO_LO_FLAG	P3_FIFO_HI_FLAG	P3_FIFO_LO_FLAG
	W													w1c	w1c	w1c	w1c
	R	0	DMA_TRANS_FINISH	0	0	IPM_ERROR	PROG_END	P2_FIFO_HI_FLAG	P2_FIFO_LO_FLAG	P1_FIFO_HI_FLAG	P1_FIFO_LO_FLAG	CRC_OVERFLOW	CRC_READY	VS_BLANK	LS_BF_VS	UNDRUN	VSYNC
	W		w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
INT_MASK 0x1F0	R	0	0	0	0	0	0	0	0	0	0	0	0	M_P4_FIFO_HI_FLAG	M_P4_FIFO_LO_FLAG	M_P3_FIFO_HI_FLAG	M_P3_FIFO_LO_FLAG
	W																
	R	0	M_DMA_TRANS_FINISH	0	0	M_IPM_ERROR	M_PROG_END	M_P2_FIFO_HI_FLAG	M_P2_FIFO_LO_FLAG	M_P1_FIFO_HI_FLAG	M_P1_FIFO_LO_FLAG	M_CRC_OVERFLOW	M_CRC_READY	M_VS_BLANK	M_LS_BF_VS	M_UNDRUN	M_VSYNC
	W																
COLBAR_1 0x1F4	R	1	1	1	1	1	1	1	1	COLBAR_1_R							
	W																
	R	COLBAR_1_G								COLBAR_1_B							
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
COLBAR_2 0x1F8	R	1	1	1	1	1	1	1	1	COLBAR_2_R							
	W																
	R	COLBAR_2_G								COLBAR_2_B							
	W																
COLBAR_3 0x1FC	R	1	1	1	1	1	1	1	1	COLBAR_3_R							
	W																
	R	COLBAR_3_G								COLBAR_3_B							
	W																
COLBAR_4 0x200	R	1	1	1	1	1	1	1	1	COLBAR_4_R							
	W																
	R	COLBAR_4_G								COLBAR_4_B							
	W																
COLBAR_5 0x204	R	1	1	1	1	1	1	1	1	COLBAR_5_R							
	W																
	R	COLBAR_5_G								COLBAR_5_B							
	W																
COLBAR_6 0x208	R	1	1	1	1	1	1	1	1	COLBAR_6_R							
	W																
	R	COLBAR_6_G								COLBAR_6_B							
	W																
COLBAR_7 0x20C	R	1	1	1	1	1	1	1	1	COLBAR_7_R							
	W																
	R	COLBAR_7_G								COLBAR_7_B							
	W																
COLBAR_8 0x210	R	1	1	1	1	1	1	1	1	COLBAR_8_R							
	W																
	R	COLBAR_8_G								COLBAR_8_B							
	W																
DIV_RATIO 0x214	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DIV_RATIO							
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SIGN_CALC_1 0x218	R	0	0	0	0	0	0	SIG_VER_SIZE									
	W																
	R	0	0	0	0	0	0	SIG_HOR_SIZE									
	W																
SIGN_CALC_2 0x21C	R	0	0	0	0	0	0	SIG_VER_POS									
	W																
	R	0	0	0	0	0	0	SIG_HOR_POS									
	W																
CRC_VAL 0x220	R	CRC_VAL															
	W																
	R																
	W																
PDI_STATUS 0x224	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	PDI_BLANKING_ERR	PDI_ECC_ERR2	PDI_ECC_ERR1	PDI_LOCK_LOST	PDI_LOCK_DET	PDI_VSYNC_DET	PDI_HSYNC_DET	PDI_DE_DET	PDI_CLK_LOST	PDI_CLK_DET
	W							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Mask_PDI_STATUS 0x228	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	M_PDI_BLANKING_ERR	M_PDI_ECC_ERR2	M_PDI_ECC_ERR1	M_PDI_LOCK_LOST	M_PDI_LOCK_DET	M_PDI_VSYNC_DET	M_PDI_HSYNC_DET	M_PDI_DE_DET	M_PDI_CLK_LOST	M_PDI_CLK_DET
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PARR_ERR_STATUS 0x22C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	HWC_ERR	SIG_ERR	DISP_ERR
	W														w1c	w1c	w1c
	R	L15_PARR_ERR	L14_PARR_ERR	L13_PARR_ERR	L12_PARR_ERR	L11_PARR_ERR	L10_PARR_ERR	L9_PARR_ERR	L8_PARR_ERR	L7_PARR_ERR	L6_PARR_ERR	L5_PARR_ERR	L4_PARR_ERR	L3_PARR_ERR	L2_PARR_ERR	L1_PARR_ERR	L0_PARR_ERR
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Mask_PARR_ERR_STATUS 0x230	R	0	0	0	0	0	0	0	0	0	0	0	0	0	M_HWC_ERR	M_SIG_ERR	M_DISP_ERR
	W																
	R	M_L15_PARR_ERR	M_L14_PARR_ERR	M_L13_PARR_ERR	M_L12_PARR_ERR	M_L11_PARR_ERR	M_L10_PARR_ERR	M_L9_PARR_ERR	M_L8_PARR_ERR	M_L7_PARR_ERR	M_L6_PARR_ERR	M_L5_PARR_ERR	M_L4_PARR_ERR	M_L3_PARR_ERR	M_L2_PARR_ERR	M_L1_PARR_ERR	M_L0_PARR_ERR
	W																
THRESHOLD_INP_BUF 0x234	R	INP_BUF_P2_HI								INP_BUF_P2_LO							
	W																
	R	INP_BUF_P1_HI								INP_BUF_P1_LO							
	W																
THRESHOLD_INP_BUF 0x238	R	INP_BUF_P4_HI								INP_BUF_P4_LO							
	W																
	R	INP_BUF_P3_HI								INP_BUF_P3_LO							
	W																
LUMA_COMP 0x23C	R	Y_RED											Y_GREEN[0:4]				
	W																
	R	Y_GREEN[5:9]						Y_BLUE									
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CHROMA_RED 0x240	R	0	0	0	0	0	Cr_RED										
	W																
	R	0	0	0	0	Cb_RED											
	W																
CHROMA_GREEN 0x244	R	0	0	0	0	0	Cr_GREEN										
	W																
	R	0	0	0	0	Cb_GREEN											
	W																
CHROMA_BLUE 0x248	R	0	0	0	0	0	Cr_BLUE										
	W																
	R	0	0	0	0	Cb_BLUE											
	W																
CRC_POS 0x24c	R	CRC_POS															
	W																
	R																
	W																
FGx_fcolor 0x250	R	1	1	1	1	1	1	1	1	FGX_FCOLOR[0:7]							
	W																
	R	FGX_FCOLOR[8:23]															
	W																
FGx_bcolor 0x254	R	1	1	1	1	1	1	1	1	FGX_BCOLOR[0:7]							
	W																
	R	FGX_BCOLOR[8:23]															
	W																
Global_protection 0x300	R	HLB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Soft_Lock_Bit L0 0x304	R	0	0	0	0	SLB_L0_1	SLB_L0_2	SLB_L0_3	SLB_L0_4	0	0	0	0	SLB_L0_5	SLB_L0_6	SLB_L0_7	0
	W	WEN_LO_1	WEN_LO_2	WEN_LO_3	WEN_LO_4					WEN_LO_5	WEN_LO_6	WEN_LO_7					
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
Soft_Lock_Bit L1 0x308	R	0	0	0	0	SLB_L1_1	SLB_L1_2	SLB_L1_3	SLB_L1_4	0	0	0	0	SLB_L1_5	SLB_L1_6	SLB_L1_7	0
	W	WEN_L1_1	WEN_L1_2	WEN_L1_3	WEN_L1_4					WEN_L1_5	WEN_L1_6	WEN_L1_7					
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
Soft_Lock_DISP_SIZE 0x30c	R	0	0	0	0	SLB_DISP	0	0	0	0	0	0	0	0	0	0	0
	W	WEN_DISP															
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
Soft_Lock_HSYNC/VSYNC PARA 0x310	R	0	0	0	0	SLB_HSYNC	SLB_VSYNC	0	0	0	0	0	0	0	0	0	0
	W	WEN_HSYNC	WEN_VSYNC														
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
Soft_Lock_POL 0x314	R	0	0	0	0	SLB_POL	0	0	0	0	0	0	0	0	0	0	0
	W	WEN_POL															
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 12-4. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Soft_Lock L0_TRANSP 0x318	R	0	0	0	0	SLB_FCOLOR	SLB_BCOLOE	0	0	0	0	0	0	0	0	0	0
	W	WEN_FCOLOR	WEN_BCOLOR														
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
Soft_Lock L1_TRANSP 0x31C	R	0	0	0	0	SLB_FCOLOR	SLB_BCOLOE	0	0	0	0	0	0	0	0	0	0
	W	WEN_FCOLOR	WEN_BCOLOR														
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

12.3.4 Register descriptions

This section describes the DCU registers.

12.3.4.1 Control Descriptor L0_1 Register (CtrlDescL0_1)

Figure 12-3 represents the control descriptor L0_1 register. This register sets the height and width of the layer associated with the register.

Offset:

0x000 (CtrlDescL0_1)	0x0E0 (CtrlDescL8_1)
0x01C (CtrlDescL1_1)	0x0FC (CtrlDescL9_1)
0x038 (CtrlDescL2_1)	0x118 (CtrlDescL10_1)
0x054 (CtrlDescL3_1)	0x134 (CtrlDescL11_1)
0x070 (CtrlDescL4_1)	0x150 (CtrlDescL12_1)
0x08C (CtrlDescL5_1)	0x16C (CtrlDescL13_1)
0x098 (CtrlDescL6_1)	0x188 (CtrlDescL14_1)
0x0C4 (CtrlDescL7_1)	0x194 (CtrlDescL15_1)

Access: User read/write

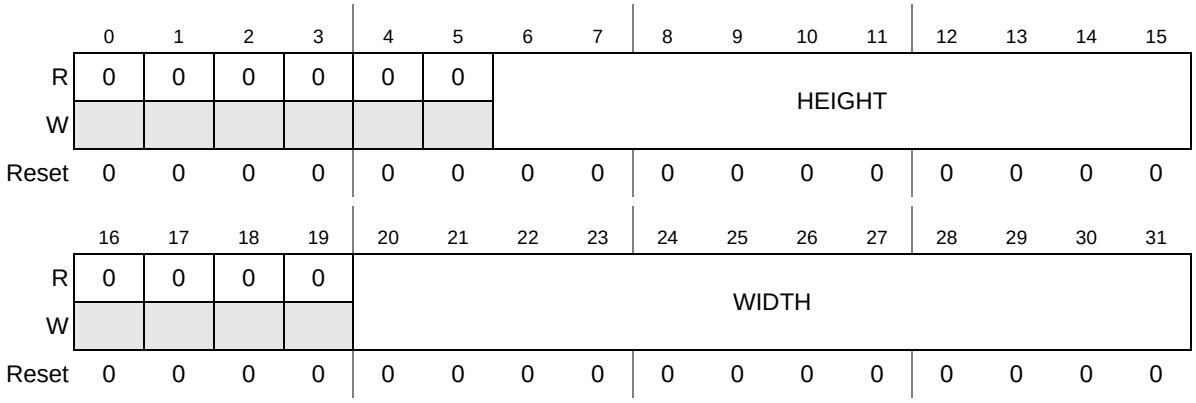


Figure 12-3. CtrlDescL0_1 Register

Table 12-5. CtrlDescL0_1 field descriptions

Field	Description
6–15 HEIGHT	Height of the layer in pixels
20–31 WIDTH	Width of the layer (in pixels). The layer width must be in multiples of the number of pixels that can be stored in 32 bits (except for the special case of 1 bit per pixel), and therefore differs depending on color encoding. For example, if 2 bits per pixel format is used, then the layer width must be configured in multiples of 16. See Section 12.4.5.3, Layer size and positioning.

12.3.4.2 Control Descriptor L0_2 Register

Figure 12-4 represents the control descriptor L0_2 register. This register sets the origin (top/left) of the layer associated with the register.

Offset:

0x004 (CtrlDescL0_2)	0x0E4 (CtrlDescL8_2)
0x020 (CtrlDescL1_2)	0x100 (CtrlDescL9_2)
0x03C (CtrlDescL2_2)	0x11C (CtrlDescL10_2)
0x058 (CtrlDescL3_2)	0x138 (CtrlDescL11_2)
0x074 (CtrlDescL4_2)	0x154 (CtrlDescL12_2)
0x090 (CtrlDescL5_2)	0x170 (CtrlDescL13_2)
0x0AC (CtrlDescL6_2)	0x18C (CtrlDescL14_2)
0x0C8 (CtrlDescL7_2)	0x198 (CtrlDescL15_2)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	POSX											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-4. CtrlDescL0_2 Register

Table 12-6. CtrlDescL0_2 field descriptions

Field	Description
6–15 POSY	Amount of pixels from the top of display frame
20–31 POSX	Amount of pixels from the left of display frame

12.3.4.3 Control Descriptor L0_3 Register

Figure 12-5 represents the control descriptor L0_3 register. This register sets the beginning address of layer data.

Offset:
0x008 (CtrlDescL0_3) 0x0E8 (CtrlDescL8_3)
0x024 (CtrlDescL1_3) 0x104 (CtrlDescL9_3)
0x040 (CtrlDescL2_3) 0x120 (CtrlDescL10_3)
0x05C (CtrlDescL3_3) 0x13C (CtrlDescL11_3)
0x078 (CtrlDescL4_3) 0x158 (CtrlDescL12_3)
0x094 (CtrlDescL5_3) 0x174 (CtrlDescL13_3)
0x0b0 (CtrlDescL6_3) 0x18C (CtrlDescL14_3)
0x0CC (CtrlDescL7_3) 0x1AC (CtrlDescL15_3)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-5. CtrlDescL0_3 Register

Table 12-7. CtrlDescL0_3 field descriptions

Field	Description
0–31 ADDR	Address of layer data in the memory. The address programmed should be 32-bit aligned.

12.3.4.4 Control Descriptor L0_4 Register

Figure 12-6 represents the control descriptor L0_4 register. This register controls various graphics options and whether the layer is enabled.

Offset:

0x00C (CtrlDescL0_4)	0x0EC (CtrlDescL8_4)
0x028 (CtrlDescL1_4)	0x108 (CtrlDescL9_4)
0x044 (CtrlDescL2_4)	0x124 (CtrlDescL10_4)
0x060 (CtrlDescL3_4)	0x140 (CtrlDescL11_4)
0x07C (CtrlDescL4_4)	0x15C (CtrlDescL12_4)
0x098 (CtrlDescL5_4)	0x178 (CtrlDescL13_4)
0x0B4 (CtrlDescL6_4)	0x190 (CtrlDescL14_4)
0x0D0 (CtrlDescL7_4)	0x1B0 (CtrlDescL15_4)

Access: User read/write

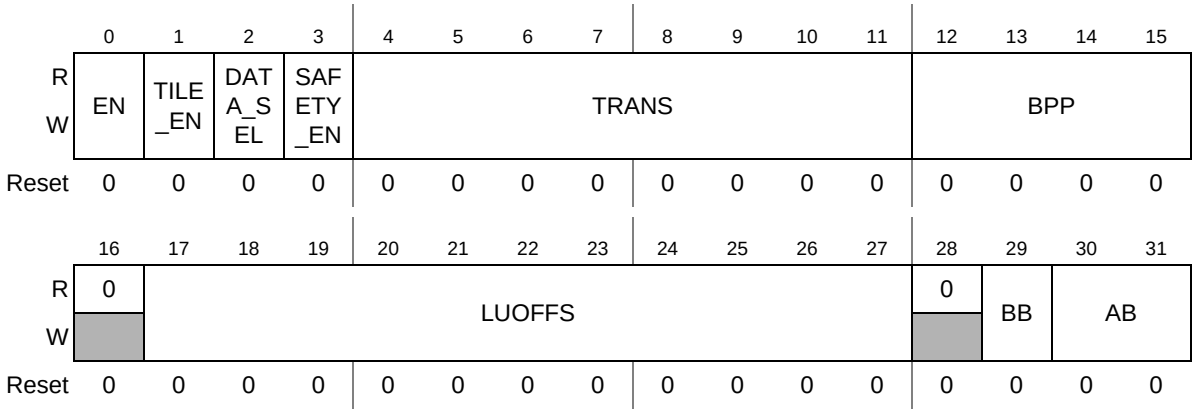


Figure 12-6. CtrlDescL0_4 Register

Table 12-8. CtrlDescL0_4 field descriptions

Field	Description
0 EN	Enable the layer 0 OFF 1 ON
1 TILE_EN	Enable the Tile mode 0 OFF 1 ON
2 DATA_SEL	Selects the Tile data either from MCU memory or CLUT 0 Tile mode data resides in the MCU memory 1 Tile mode data resides in the CLUT
3 SAFETY_EN	Safety Mode Enable Bit. Valid only for layer 0 and layer 1. For registers of all other layers, this should be set to 0. 0 Safety mode is disabled 1 Safety mode is enabled for this layer
4–11 TRANS	Transparency Level. Specifies the alpha value for the layer. This value may be used by the blending engine to blend pixels on this layer. Value can vary between 0-255 where 0 is completely transparent and 255 is completely opaque.

Table 12-8. CtrlDescL0_4 field descriptions (continued)

Field	Description
12–15 BPP	Bits Per Pixel 0000 1 bpp 0001 2 bpp 0010 4 bpp 0011 8 bpp 0100 16 bpp (RGB565) 0101 24 bpp 0110 32 bpp (ARGB8888) 0111 Transparency mode 4 bpp 1000 Transparency mode 8bpp 1001 Luminance offset mode 4 bpp 1010 Luminance offset mode 8 bpp 1011 16 bpp (ARGB1555) 1100 16 bpp (ARGB4444) 1101–1111 Reserved
17–27 LUOFFS	Look Up Table offset. Value gives the offset to the start address of the CLUT or tile (when used in internal tile mode) in the CLUT/TILE RAM.
29 BB	Chroma Keying 0 OFF 1 ON
30–31 AB	Alpha Blending 00 No Alpha blending 01 Blend only the pixels selected by chroma keying in case BB = 1 10 Blend the whole frame 11 Same functionality as 00 (No Alpha blending).

12.3.4.5 Control Descriptor L0_5 Register

Figure 12-7 represents the control Descriptor L0_5 register. This register sets the maximum Chroma Keying values for RGB.

Refer to Section 12.4.5.5, Alpha and Chroma-key blending, for a description of Chroma Keying.

Offset:
0x010 (CtrlDescL0_5)
0x02C (CtrlDescL1_5)
0x048 (CtrlDescL2_5)
0x064 (CtrlDescL3_5)
0x080 (CtrlDescL4_5)
0x09C (CtrlDescL5_5)
0x0B8 (CtrlDescL6_5)
0x0D4 (CtrlDescL7_5
)
0x0F0 (CtrlDescL8_5)
0x10C (CtrlDescL9_5)
0x128 (CtrlDescL10_5)
0x144 (CtrlDescL11_5)
0x160 (CtrlDescL12_5)
0x17C (CtrlDescL13_5)
0x198 (CtrlDescL14_5)
0x1B4 (CtrlDescL15_5)

Access: User read/write

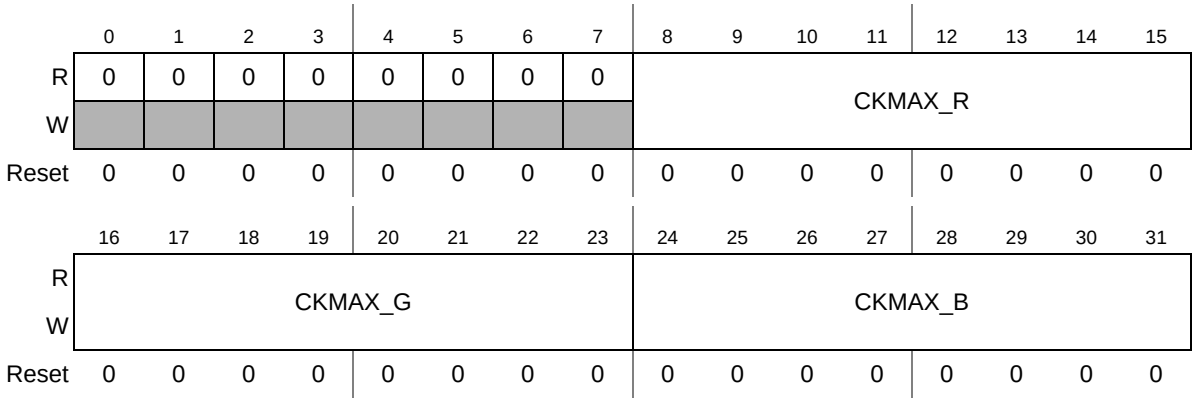


Figure 12-7. CtrlDescL0_5 Register

Table 12-9. CtrlDescL0_5 field descriptions

Field	Description
8–15 CKMAX_R	Chroma Keying Max Red Component
16–23 CKMAX_G	Chroma Keying Max Green Component
24–31 CKMAX_B	Chroma Keying Max Blue Component

12.3.4.6 Control Descriptor L0_6 Register

Figure 12-8 represents the control descriptor L0_6 register. This register sets the minimum Chroma Keying values for RGB. Refer to Section 12.4.5.5, Alpha and Chroma-key blending, for a description of Chroma Keying.

Offset:

0x014 (CtrlDescL0_6)	0x0F4 (CtrlDescL8_6)
0x030 (CtrlDescL1_6)	0x110 (CtrlDescL9_6)
0x04C (CtrlDescL2_6)	0x12C (CtrlDescL10_6)
0x068 (CtrlDescL3_6)	0x148 (CtrlDescL11_6)
0x084 (CtrlDescL4_6)	0x164 (CtrlDescL12_6)
0x0A0 (CtrlDescL5_6)	0x180 (CtrlDescL13_6)
0x0BC (CtrlDescL6_6)	0x198 (CtrlDescL14_6)
0x0D8 (CtrlDescL7_6)	0x1B8 (CtrlDescL15_6)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CKMIN_G								CKMIN_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-8. CtrlDescL0_6 Register

Table 12-10. CtrlDescL0_6 field descriptions

Field	Description
8–15 CKMIN_R	Chroma Keying Minimum Red Component
16–23 CKMIN_G	Chroma Keying Minimum Green Component
24–31 CKMIN_B	Chroma Keying Minimum Blue Component

12.3.4.7 Control Descriptor L0_7 Register

Figure 12-9 represents the Control Descriptor L0_7 Register.

Offset:
0x018 (CtrlDescL0_7) 0x0F8 (CtrlDescL8_7)
0x034 (CtrlDescL1_7) 0x114 (CtrlDescL9_7
0x050 (CtrlDescL2_7) 0x130 (CtrlDescL10_7)
0x06C (CtrlDescL3_7) 0x14C (CtrlDescL11_7)
0x088 (CtrlDescL4_7) 0x168 (CtrlDescL12_7)
0x094 (CtrlDescL5_7) 0x184 (CtrlDescL13_7)
0x0C0 (CtrlDescL6_7) 0x19C (CtrlDescL14_7)
0x0DC (CtrlDescL7_7) 0x1BC (CtrlDescL15_7)

Access: User read/write

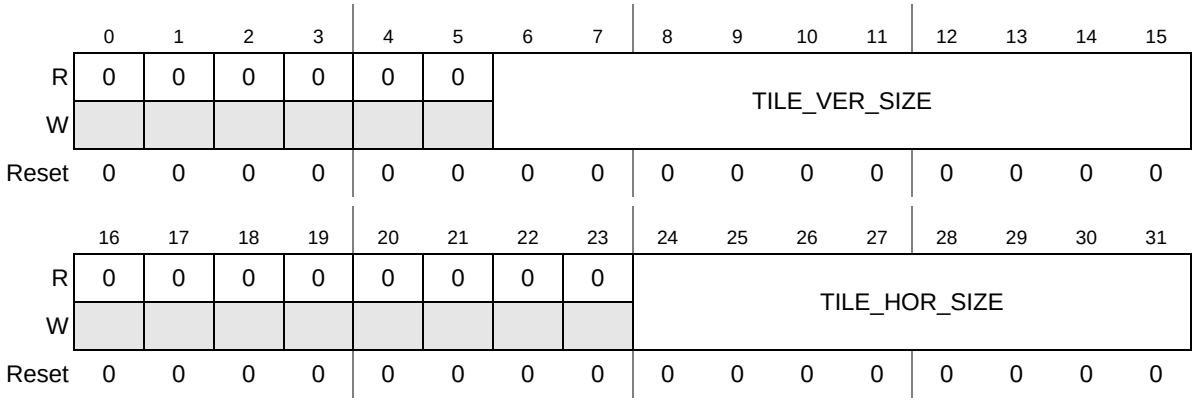


Figure 12-9. Control Descriptor L0_7 Register

Table 12-11. Control Descriptor L0_7 Register

Field	Description
6–15 TILE_VER_SIZE	Height of the TILE (in pixels)
24–31 TILE_HOR_SIZE	Width of the TILE (in multiples of 16 pixels)

For the other 16 layers, the Control Descriptor Register set is identical.

12.3.4.8 Control Descriptor Cursor 1 Register (CtrlDescCursor_1)

Figure 12-10 represents the Control Descriptor Cursor 1 register.

Offset: 0x1C0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	WIDTH									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-10. Control Descriptor Cursor 1 Register (CtrlDescCursor_1)

Table 12-12. CtrlDescCursor_1 field descriptions

Field	Description
6–15 HEIGHT	Height of the cursor in pixels
22–31 WIDTH	Width of the cursor in pixels

12.3.4.9 Control Descriptor Cursor 2 Register (CtrlDescCursor_2)

Figure 12-11 represents the Control descriptor Cursor 2 register.

Offset: 0x1C4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	POSX									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-11. Control Descriptor Cursor 2 Register (CtrlDescCursor_2)

Table 12-13. CtrlDescCursor_2 field descriptions

Field	Description
6–15 POSY	Y position of the cursor in pixels
22–31 POSX	X position of the cursor in pixels

12.3.4.10 Control Descriptor Cursor 3 Register (CtrlDescCursor_3)

Figure 12-12 represents the Control Descriptor Cursor 3 register.

Offset: 0x1C8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DEFAULT_CURSOR_COLOR[9:24]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-12. Control Descriptor Cursor 3 Register (CtrlDescCursor_3)

Table 12-14. Control Descriptor Cursor_3 field descriptions

Field	Description
0 CUR_EN	Cursor Enable signal 0 Cursor is disabled 1 Enable the cursor
8–31 DEFAULT_CURSOR_COLOR	Default pixel color value for the cursor. In the DCU, the pixel value for the cursor is fixed for a particular frame.

12.3.4.11 Control Descriptor Cursor 4 Register (CtrlDescCursor_4)

Figure 12-13 represents the Control Descriptor Cursor 4 register.

Offset: 0x1CC Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	EN_BLINK	HWC_BLINK_ON							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-13. Control Descriptor Cursor 4 Register (CtrlDescCursor_4)

Table 12-15. CtrlDescCursor_4 field descriptions

Field	Description
8–15 HWC_BLINK_OFF	HWC blink register. Loads the counter value (number of frames) for which the cursor will remain turned OFF.
23 EN_BLINK	Enable the cursor blink mode. 0 Disable the blink mode 1 Enable the blink mode
24–31 HWC_BLINK_ON	HWC blink register. Loads the counter value (number of frames) for which the cursor will remain turned ON.

12.3.4.12 DCU Mode Register (DCU_MODE)

Figure 12-14 represents the DCU_MODE register.This register sets the mode in which DCU is operating.

Offset: 0x1D0 Access: User read/write

Figure 12-14. DCU Mode Register (DCU_MODE)

Table 12-16. DCU_MODE field descriptions

Field	Description
0 DCU_SW_RESET	Used to clear all the registers to reset state 0 DCU registers are not cleared 1 All the DCU registers are put in reset state
9–11 BLEND_ITER	Defines the number of planes used for blending. 010 Two-plane blending 011 Three-plane blending 100 Four-plane blending All other values: two-plane blending is selected and BLEND_ITER is set to 2.
12–15 PDI_SYNC_LOCK	Defines the number of frames which should be received by the PDI validation state machine before it locks and sets the PDI_LOCK_DET bit in the PDI Status Register (see Section 12.3.4.26, PDI Status Register).
16 PDI_INTERPOL_EN	Control Bit to decide whether the conversion from YCbCr 4:2:2 to 4:4:4 needs to be done using interpolation or Chroma value is same for two pixels 0 Chroma value is same for two pixels 1 Interpolation is enabled.
17 RASTER_EN	Enables raster scanning of pixel data including the VSYNC and HSYNC signals and the pixel data. For correct operation RASTER_EN should only be changed when DCU_MODE is configured to off. Changes to this bit take effect after the completion of the current frame. 0 Disabled 1 Enabled

Table 12-16. DCU_MODE field descriptions (continued)

Field	Description
18 PDI_EN	Enables the PDI 0 Disabled 1 Enabled
19 PDI_BYTE_REV	Controls the byte ordering in Narrow mode 0 LSB is followed by MSB data 1 MSB is followed by LSB data
20 PDI_DE_MODE	Enables the PDI data Enable mode. Here Data Enable is treated as an input. 0 Value on data Enable signal is ignored 1 Data enable signal must be present in incoming stream
21 PDI_NARROW_MODE	Enables the PDI Narrow mode (refer to Section 12.8.2.4, Normal and Narrow mode) 0 Narrow mode is disabled 1 Narrow mode is enabled
22–23 PDI_MODE	Defines the different modes in which PDI is operating 00 8-bit monochrome data input 01 16-bit RGB 565 format 10 18-bit RGB 666 data format. 11 YCbCr data in 4:2:2 format.
24 PDI_SLAVE_MODE	Enables PDI Slave mode 0 Disabled 1 Enabled
25 TAG_EN	Enables the calculation of CRC only on the safety layers 0 CRC calculated over the whole area of interest (area of interest given by SIG_DESC registers) 1 Calculates CRC only on safety enabled layers
26 SIG_EN	Enables the signature calculator block 0 Signature calculator is disabled 1 Signature calculator is enabled
27 PDI_SYNC	Decides whether the PDI uses external or internal synchronization. 0 External Synchronization. The PDI receives the SYNC (hsync, vsync) signals from external source. 1 Internal Synchronization. PDI extracts the SYNC information from the digital data.
29 EN_GAMMA	Enables/Disables the Gamma correction 0 Gamma correction is disabled 1 Gamma correction is enabled
30–31 DCU_MODE	DCU operating mode and pixel clock enable 00 DCU off 01 Normal mode. Pixel clock active and panel content controlled by layer configuration 10 Test mode. Panel content fetched from CLUT/Tile memory 11 Color Bar Generation. Pixel clock active and panel content controlled by color bar registers.

12.3.4.13 BGND Register

Figure 12-15 represents the BGND register.

Offset: 0x1D4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BGND_G								BGND_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-15. BGND Register

Table 12-17. BGND field descriptions

Field	Description
8–15 BGND_R	Red component of the default color displayed in the sectors where no layer is active
16–23 BGND_G	Green component of the default color displayed in the sectors where no layer is active
24–31 BGND_B	Blue component of the default color displayed in the sectors where no layer is active

12.3.4.14 DISP_SIZE Register

Figure 12-16 represents the DISP_SIZE register

Offset: 0x1D8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DELTA_X							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-16. DISP_SIZE Register

Table 12-18. DISP_SIZE field descriptions

Field	Description
6–15 DELTA_Y	Sets the display size vertical resolution (in pixels)
24–31 DELTA_X	Sets the display size horizontal resolution (in multiples of 16 pixels)

12.3.4.15 HSYN_PARA Register

Figure 12-17 represents the HSYN_PARA register. HSYN_PARA register sets timing parameters related to the horizontal synchronization signal generation. The fields FP_H, BP_H, and PW_H stand for HSYNC signal front-porch, back-porch, and active pulse width, respectively.

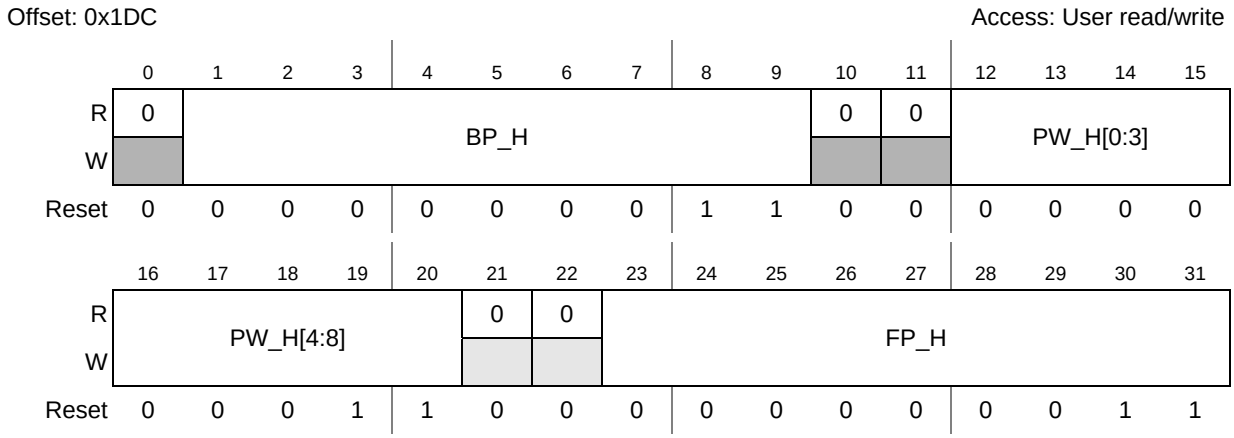


Figure 12-17. HSYN_PARA Register

Table 12-19. HSYN_PARA field descriptions

Field	Description
1–9 BP_H	HSYNC back-porch pulse width (in pixel clock cycles). Pulse width has a minimum value of 1.
12–20 PW_H	HSYNC active pulse width (in pixel clock cycles).
23–31 FP_H	HSYNC front-porch pulse width (in pixel clock cycles). Pulse width has a minimum value of 1.

12.3.4.16 VSYN_PARA Register

Figure 12-18 represents the VSYN_PARA register. VSYN_PARA register sets timing parameters related to the vertical synchronization signal generation. The fields FP_V, BP_V, and PW_V stand for VSYNC signal front-porch, back-porch, and active pulse width, respectively.

Offset: 0x1E0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PW_V[4:8]					0	0	FP_V								
W																
Reset	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1

Figure 12-18. VSYN_PARA Register

Table 12-20. VSYN_PARA field descriptions

Field	Description
1–9 BP_V	VSYNC back-porch pulse width (in horizontal line cycles). Pulse width has a minimum value of 1.
12–20 PW_V	VSYNC active pulse width (in horizontal line cycles).
23–31 FP_V	VSYNC front-porch pulse width (in horizontal line cycles). Pulse width has a minimum value of 1.

12.3.4.17 SYN_POL Register

Figure 12-19 represents the SYN_POL register. SYN_POL register selects polarity for corresponding synchronize signals (HSYNC, VSYNC, CSYNC), and controls the bypass of HSYNC or VSYNC with CSYNC signal.

Offset: 0x1E4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0											
W						INV_PDI_DE	INV_PDI_HS	INV_PDI_VS	INV_PDI_CLK	INV_PXCK	NEG	BP_VS	BP_HS	INV_CS	INV_VS	INV_HS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-19. SYN_POL Register

Table 12-21. SYN_POL field descriptions

Field	Description
21 INV_PDI_DE	Polarity change of PDI input data Enable. 0 DE is active high 1 DE is active low
22 INV_PDI_HS	Polarity change of PDI input HSYNC. 0 HSYNC is active high 1 HSYNC is active low
23 INV_PDI_VS	Polarity change of PDI input VSYNC. 0 VSYNC is active high 1 VSYNC is active low
24 INV_PDI_CLK	Polarity change of PDI input Clock. 0 DCU samples data on the rising edge 1 DCU samples data on the falling edge
25 INV_PXCK	Polarity change of Pixel Clock. 0 Display samples data on the falling edge 1 Display samples data on the rising edge
26 NEG	Indicates if value at the output (pixel data output) needs to be negated. 0 Output is to remain same 1 Output to be negated
27 BP_VS	Bypass Vertical Synchronize Signal (internal pin muxing). 0 Do not bypass VSYNC signal output 1 CSYNC bypass VSYNC signal, output CSYNC instead of VSYNC
28 BP_HS	Bypass Horizontal Synchronize Signal (internal pin muxing). 0 Do not bypass HSYNC signal output 1 CSYNC bypass HSYNC signal, output CSYNC instead of HSYNC
29 INV_CS	Invert Composite Synchronize Signal. 0 Do not invert CSYNC signal, active HIGH 1 Invert CSYNC signal, active LOW
30 INV_VS	Invert Vertical Synchronize Signal 0 Do not invert VSYNC signal, active HIGH 1 Invert VSYNC signal, active LOW
31 INV_HS	Invert Horizontal Synchronize Signal. 0 Do not invert HSYNC signal, active HIGH 1 Invert HSYNC signal, active LOW

12.3.4.18 Threshold Register

Figure 12-20 represents the Threshold Register.

Offset: 0x1E8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUT_BUF_HIGH								OUT_BUF_LOW							
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0

Figure 12-20. Threshold Register

Table 12-22. Threshold Register field descriptions

Field	Description
6–15 LS_BF_VS	Lines before VSYNC threshold value. The LS_BF_VS status flag (in INT_STATUS) is set this number of lines before the VSYNC signal is asserted.
16–23 OUT_BUF_HIGH	Output buffer high threshold (in pixels). When the output buffer exceeds this value the datapath clock is suspended.
24–31 OUT_BUF_LOW	Output buffer filling low Threshold (in pixels).This value is used to generate the underrun exception (UNDRUN in INT_STATUS).

12.3.4.19 Interrupt Status Register (INT_STATUS)

Figure 12-21 indicates the interrupt status register. See Section 12.5.4, Interrupt generation, for a description of how the DCU collects interrupt events into different source groups.

Offset: 0x1EC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	P4_FIFO_HI_FLAG	P4_FIFO_LO_FLAG	P3_FIFO_HI_FLAG	P3_FIFO_LO_FLAG
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	DMA_TRANS_FINISH	0	0	IPM_ERROR	PROG_END	P2_FIFO_HI_FLAG	P2_FIFO_LO_FLAG	P1_FIFO_HI_FLAG	P1_FIFO_LO_FLAG	CRC_OVERFLOW	CRC_READY	VS_BLANK	LS_BF_VS	UNDRUN	VSYNC
W		w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-21. Interrupt Status Register (INT_STATUS)

Table 12-23. INT_STATUS field descriptions

Field	Description
12 P4_FIFO_HI_FLAG	Interrupt signal that indicates the High threshold has been reached for plane 4(FG2plane) input buffer
13 P4_FIFO_LO_FLAG	Interrupt signal that indicates the Low threshold has been reached for plane 4(FG2plane) input buffer
14 P3_FIFO_HI_FLAG	Interrupt signal that indicates the High threshold has been reached for plane 3(FG1plane) input buffer
15 P3_FIFO_LO_FLAG	Interrupt signal that indicates the Low threshold has been reached for plane 3(FG1plane) input buffer
17 DMA_TRANS_FINISH	Interrupt signal that indicates that the DCU DMA has fetched the last pixel of data from the memory
20 IPM_ERROR	Interrupt signal that indicates that an error has occurred in the Magenta line transaction
21 PROG_END	Interrupt signal that indicates that the duration for programming of DCU registers and internal memories is finished
22 P2_FIFO_HI_FLAG	Interrupt signal that indicates the High threshold has been reached for plane 2 (FGplane) input buffer

Table 12-23. INT_STATUS field descriptions (continued)

Field	Description
23 P2_FIFO_LO_FLAG	Interrupt signal that indicates the Low threshold has been reached for plane 2 (FGplane) input buffer
24 P1_FIFO_HI_FLAG	Interrupt signal that indicates the High threshold has been reached for plane 1 (BGplane) input buffer
25 P1_FIFO_LO_FLAG	Interrupt signal that indicates the Low threshold has been reached for plane 1 (BGplane) input buffer
26 CRC_OVERFLOW	Interrupt signal that indicates the CRC_ready has not been serviced and CRC has been calculated for the next frame
27 CRC_READY	Interrupt signal to indicate CRC calculation is done and ready to be compared with precomputed CRC value by the software
28 VS_BLANK	Interrupt signal to indicate vertical blanking period. This is the period in which all the registers that affect the visible state of the layers need to be latched. This is needed so that CPU writes to the register while the display is being updated does not cause any errors.
29 LS_BF_VS	Lines Before Vsync interrupt. It is generated threshold LS_BF_VS number of lines ahead of the vertical front porch (FP_V) if enabled. The CPU can program the registers after LS_BF_VS interrupt.
30 UNDRUN	Under Run Exception Interrupt. Asserted when display needs data and output buffer filling is lower than or equal to the OUT_BUF_LOW threshold. Interrupt is cleared when the data in the output buffer is greater than threshold and CPU writes 1 to this bit.
31 VSYNC	Vertical Synchronize Interrupt. If enabled, an interrupt is generated at the beginning of a frame.

12.3.4.20 Interrupt Mask Register (INT_MASK)

Figure 12-22 represents the interrupt mask register.This register enables or masks corresponding interrupt.

Offset: 0x1F0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	M_P4_FIFO_HI_FLAG	M_P4_FIFO_LO_FLAG	M_P3_FIFO_HI_FLAG	M_P3_FIFO_LO_FLAG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M_DMA_TRANS_FINISH	0	0	M_IPM_ERROR	M_PROG_END	M_P2_FIFO_HI_FLAG	M_P2_FIFO_LO_FLAG	M_P1_FIFO_HI_FLAG	M_P1_FIFO_LO_FLAG	M_CRC_OVERFLOW	M_CRC_READY	M_VS_BLANK	M_LS_BF_VS	M_UNDRUN	M_VSYNC
W																
Reset	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 12-22. Interrupt Mask Register (INT_MASK)

Table 12-24. INT_MASK field descriptions

Field	Description
12 M_P4_FIFO_HI_FLAG	P4_FIFO_HI_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
13 M_P4_FIFO_LO_FLAG	P4_FIFO_LO_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
14 M_P3_FIFO_HI_FLAG	P3_FIFO_HI_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
15 M_P3_FIFO_LO_FLAG	P3_FIFO_LO_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
17 M_DMA_TRANS_FINISH	DMA_TRANS_FINISH interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
20 M_IPM_ERROR	IPM_ERROR interrupt mask 0 Interrupt is not masked 1 Interrupt is masked

Table 12-24. INT_MASK field descriptions (continued)

Field	Description
21 M_PROG_END	PROG_END interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
22 M_P2_FIFO_HI_FLAG	P2_FIFO_HI_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
23 M_P2_FIFO_LO_FLAG	P2_FIFO_LO_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
24 M_P1_FIFO_HI_FLAG	P1_FIFO_HI_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
25 M_P1_FIFO_LO_FLAG	P1_FIFO_LO_FLAG interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
26 M_CRC_OVERFLOW	CRC_OVERFLOW interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
27 M_CRC_READY	CRC_READY interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
28 M_VS_BLANK	VS_BLANK interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
29 M_LS_BF_VS	LS_BF_VS interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
30 M_UNDRUN	UNDRUN interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
31 M_VSYNC	VSYNC interrupt mask 0 Interrupt is not masked 1 Interrupt is masked

12.3.4.21 COLBAR Registers

The COLBAR registers are used to generate color bars in functional test mode. Eight different pixel values are taken as input data, to display 8 color bars on the display.

Table 12-25. COLBAR_*n* register field descriptions

Field Name	Description
COLBAR_*n*_R	Red component value

Table 12-25. COLBAR_*n* register field descriptions

Field Name	Description
COLBAR_*n*_G	Green component value
COLBAR_*n*_B	Blue component value

12.3.4.21.1 COLBAR_1 register

Offset: 0x1F4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_1_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_1_G								COLBAR_1_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-23. COLBAR_1 Register (Black)

12.3.4.21.2 COLBAR_2 Register

Offset: 0x1F8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_2_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_2_G								COLBAR_2_B							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 12-24. COLBAR_2 Register (Blue)

12.3.4.21.3 COLBAR_3 Register

Offset: 0x1FC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_3_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_3_G								COLBAR_3_B							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 12-25. COLBAR_3 Register (Cyan)

12.3.4.21.4 COLBAR_4 Register

Offset: 0x200

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_4_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_4_G								COLBAR_4_B							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 12-26. COLBAR_4 Register (Green)

12.3.4.21.5 COLBAR_5 Register

Offset: 0x204 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_5_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_5_G								COLBAR_5_B							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 12-27. COLBAR_5 Register (Yellow)

12.3.4.21.6 COLBAR_6 Register

Offset: 0x208 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_6_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_6_G								COLBAR_6_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-28. COLBAR_6 Register (Red)

12.3.4.21.7 COLBAR_7 Register

Offset: 0x20C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_7_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_7_G								COLBAR_7_B							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 12-29. COLBAR_7 Register (Purple)

12.3.4.21.8 COLBAR_8 Register

Offset: 0x210

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_8_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_8_G								COLBAR_8_B							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 12-30. COLBAR_8 Register (White)

12.3.4.22 Divide Ratio (DIV_RATIO) register

Figure 12-31 shows the Divide Ratio register.

Offset: 0x214 Access: User read/write

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

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

Figure 12-31. Divide Ratio (DIV_RATIO) register

Table 12-26. DIV_RATIO field descriptions

Field	Description
24–31 DIV_RATIO	Specifies the divide value for the input clock. Used to generate the pixel clock to support different types of displays. To divide by N, set the DIV_RATIO to (N-1).

12.3.4.23 SIGN_CALC_1 Register

Figure 12-32 presents the register for vertical/horizontal size of the area for CRC calculation.

Offset: 0x218 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	SIG_HOR_SIZE									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-32. SIGN_CALC_1 Register

Table 12-27. SIGN_CALC_1 field descriptions

Field	Description
6–15 SIG_VER_SIZE	Vertical size of the window of interest of pixels for CRC calculation (in pixels)
22–31 SIG_HOR_SIZE	Horizontal size of window of interest of pixels for CRC calculations (in pixels)

12.3.4.24 SIGN_CALC_2 Register

Figure 12-33 represents the register for position of the window of interest for CRC calculation.

Offset: 0x21C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	SIG_HOR_POS									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-33. SIGN_CALC_2 Register

Table 12-28. SIGN_CALC_2 field descriptions

Field	Description
6–15 SIG_VER_POS	Vertical position of the window of interest of pixels for CRC calculation (in pixels)
22–31 SIG_HOR_POS	Horizontal position of window of interest of pixels for CRC calculation (in pixels)

12.3.4.25 CRC_VAL Register

Figure 12-34 represents the register presenting the CRC value to the software for comparison.

Offset: 0x220 Access: User read/write

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

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

Figure 12-34. CRC_VAL Register

Table 12-29. CRC_VAL field descriptions

Field	Description
0–31 CRC_VAL	CRC value calculated for safety enabled layers to be presented to the software for comparison.

12.3.4.26 PDI Status Register

Figure 12-35 represents the PDI status register.

Offset: 0x224 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	PDI_BLANKING_ERR	PDI_ECC_ERR2	PDI_ECC_ERR1	PDI_LOCK_LOST	PDI_LOCK_DET	PDI_VSYNC_DET	PDI_HSYNC_DET	PDI_DE_DET	PDI_CLK_LOST	PDI_CLK_DET
W							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-35. PDI Status Register

Table 12-30. PDI Status Register field descriptions

Field	Description
22 pdi_blanking_err	Status bit to inform the software that 80h,10h sequence is not present during the blanking period in internal sync mode. 0 Correct data sequence present in blanking period 1 Correct data sequence not present in blanking period
23 pdi_ecc_err2	Status bit to inform the software about multibit bit error that is detected. 0 Multibit ECC error is not detected 1 Multibit ECC error detected
24 pdi_ecc_err1	Status bit to inform the software about one bit error is detected. 0 One bit ECC error is not detected 1 One bit ECC error detected
25 pdi_lock_lost	Status bit to inform the software that frame lock is lost. 0 Frame is locked 1 Frame lock is lost

Table 12-30. PDI Status Register field descriptions (continued)

Field	Description
26 pdi_lock_det	Status bit to inform the software PDI is frame locked to the camera interface. 0 Waiting for frame to lock 1 Frame lock is detected
27 pdi_vsync_det	Status bit to inform the software that vsync for the camera data has been detected. 0 pdi_vsync not detected 1 pdi_vsync is detected
28 pdi_hsync_det	Status bit to inform the software that hsync for the camera data has been detected. 0 pdi_hsync not detected 1 pdi_hsync is detected
29 pdi_de_det	Status bit to inform the software that data Enable for the camera data has been detected. 0 pdi_de not detected 1 pdi_de is detected
30 pdi_clk_lost	Status bit to inform the software that pdi_clk is lost 0 pdi_clk is present 1 pdi_clk is lost
31 pdi_clk_det	Status bit to inform the software that clock for the camera data has been detected. 0 pdi_clk not detected 1 pdi_clk is detected

12.3.4.27 PDI Status Mask Register

Figure 12-36 represents the Mask PDI status register

Offset: 0x228 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0										
W							m_pdi_blanking_err	m_pdi_ecc_err2	m_pdi_ecc_err1	m_pdi_lock_lost	m_pdi_lock_det	m_pdi_vsync_det	m_pdi_hsync_det	m_pdi_de_det	m_pdi_clk_lost	m_pdi_clk_det
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Figure 12-36. PDI status mask register

Table 12-31. PDI Status Mask Register field descriptions

Field	Description
22 m_pdi_blanking_err	pdi_blanking_err interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
23 m_pdi_ecc_err2	pdi_ecc_err2 interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
24 m_pdi_ecc_err1	pdi_ecc_err1 interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
25 m_pdi_lock_lost	pdi_lock_lost interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
26 m_pdi_lock_det	pdi_lock_det interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
27 m_pdi_vsync_det	pdi_vsync_det interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
28 m_pdi_hsync_det	pdi_hsync_det interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
29 m_pdi_de_det	pdi_de_det interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
30 m_pdi_clk_lost	pdi_clk_lost interrupt mask 0 Interrupt is not masked 1 Interrupt is masked
31 m_pdi_clk_det	pdi_clk_det interrupt mask 0 Interrupt is not masked 1 Interrupt is masked

12.3.4.28 Parameter Error Status (PARR_ERR) register

Figure 12-37 shows the parameter error status register.

An error in a layer can occur under the following conditions:

a) Number of pixels in a tile > maximum tile memory size in case of Tile bandwidth optimized mode (when in internal memory mode)

b) There is an automatic error checking mechanism when a layer is enabled that detects a non-valid horizontal size and color format combination. See Section 12.4.5.3, Layer size and positioning, for details.

These errors are grouped into a single bit error for each layer. The parameter error specific to each layer is signaled only when the layer is enabled.

DISP_ERR occurs when the size of display (height or width) is set to zero or when the pulse width of hsync/vsync is programmed as zero.

SIG_ERR occurs when the area of interest for calculating CRC value is programmed with values which are outside the display.

HWC_ERR occurs if size of cursor programmed is greater than memory size(256x32).See Section 12.4.6, Hardware cursor, for further details on how cursor can be programmed.

Offset: 0x22C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	HWC_ERR	SIG_ERR	DISP_ERR
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	L15_PARR_ERR	L14_PARR_ERR	L13_PARR_ERR	L12_PARR_ERR	L11_PARR_ERR	L10_PARR_ERR	L9_PARR_ERR	L8_PARR_ERR	L7_PARR_ERR	L6_PARR_ERR	L5_PARR_ERR	L4_PARR_ERR	L3_PARR_ERR	L2_PARR_ERR	L1_PARR_ERR	L0_PARR_ERR
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-37. Parameter Error Status Register

Table 12-32. Parameter Error Status Register field descriptions

Field	Description
13 HWC_ERR	Interrupt signal to indicate HWC error. This can occur if HWC position is out of display area or cursor memory is bigger than the HWC size. When this occurs, the HWC is disabled.
14 SIG_ERR	Interrupt occurs whenever the area of interest specified by SIG_CALC register is outside the display size. 0 SIG_ERR is not set 1 SIG_ERR is set
15 DISP_ERR	Interrupt occurs whenever width and height of display, pulse width (both vertical and horizontal sync) value is 0. 0 DISP_ERR is not set 1 DISP_ERR is set
16 L15_PARR_ERR	Interrupt occurs whenever there is an error in layer 15. 0 Parameter error is not set 1 Parameter error is set

Table 12-32. Parameter Error Status Register field descriptions (continued)

Field	Description
17 L14_PARR_ERR	Interrupt occurs whenever there is an error in layer 14. 0 Parameter error is not set 1 Parameter error is set
18 L13_PARR_ERR	Interrupt occurs whenever there is an error in layer 13. 0 Parameter error is not set 1 Parameter error is set
19 L12_PARR_ERR	Interrupt occurs whenever there is an error in layer 12. 0 Parameter error is not set 1 Parameter error is set
20 L11_PARR_ERR	Interrupt occurs whenever there is an error in layer 11. 0 Parameter error is not set 1 Parameter error is set
21 L10_PARR_ERR	Interrupt occurs whenever there is an error in layer 10. 0 Parameter error is not set 1 Parameter error is set
22 L9_PARR_ERR	Interrupt occurs whenever there is an error in layer 9 0 Parameter error is not set 1 Parameter error is set
23 L08_PARR_ERR	Interrupt occurs whenever there is an error in layer 8. 0 Parameter error is not set 1 Parameter error is set
24 L7_PARR_ERR	Interrupt occurs whenever there is an error in layer 7 0 Parameter error is not set 1 Parameter error is set
25 L6_PARR_ERR	Interrupt occurs whenever there is an error in layer 6. 0 Parameter error is not set 1 Parameter error is set
26 L5_PARR_ERR	Interrupt occurs whenever there is an error in layer 5. 0 Parameter error is not set 1 Parameter error is set
27 L4_PARR_ERR	Interrupt occurs whenever there is an error in layer 4 0 Parameter error is not set 1 Parameter error is set
28 L3_PARR_ERR	Interrupt occurs whenever there is an error in layer 3. 0 Parameter error is not set 1 Parameter error is set
29 L2_PARR_ERR	Interrupt occurs whenever there is an error in layer 2. 0 Parameter error is not set 1 Parameter error is set

Table 12-32. Parameter Error Status Register field descriptions (continued)

Field	Description
30 L1_PARR_ERR	Interrupt occurs whenever there is an error in layer 1. 0 Parameter error is not set 1 Parameter error is set
31 L0_PARR_ERR	Interrupt occurs whenever there is an error in layer 0. 0 Parameter error is not set 1 Parameter error is set

12.3.4.29 Mask PARR_ERR Status register

Figure 12-38 shows the mask register for parameter error status register.

Offset: 0x230 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_L15_parr_err	M_L14_parr_err	M_L13_parr_err	M_L12_parr_err	M_L11_parr_err	M_L10_parr_err	M_L9_parr_err	M_L8_parr_err	M_L7_parr_err	M_L6_parr_err	M_L5_parr_err	M_L4_parr_err	M_L3_parr_err	M_L2_parr_err	M_L1_parr_err	M_L0_parr_err
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 12-38. Mask parameter error status register

Table 12-33. Mask parameter error status register field descriptions

Field	Description
13 M_HWC_ERR	M_HWC_ERR interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
14 M_SIG_ERR	M_SIG_ERR interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
15 M_DISP_ERR	M_DISP_ERR interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
16 M_L15_parr_err	M_L15_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt

Table 12-33. Mask parameter error status register field descriptions (continued)

Field	Description
17 M_L14_parr_err	M_L14_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
18 M_L13_parr_err	M_L13_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
19 M_L12_parr_err	M_L12_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
20 M_L11_parr_err	M_L11_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
21 M_L10_parr_err	M_L10_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
22 M_L9_parr_err	M_L9_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
23 M_L8_parr_err	M_L8_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
24 M_L7_parr_err	M_L7_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
25 M_L6_parr_err	M_L6_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
26 M_L5_parr_err	M_L5_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
27 M_L4_parr_err	M_L4_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
28 M_L3_parr_err	M_L3_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
29 M_L2_parr_err	M_L2_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt

Table 12-33. Mask parameter error status register field descriptions (continued)

Field	Description
30 M_L1_parr_err	M_L1_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt
31 M_L0_parr_err	M_L0_parr_err interrupt mask 0 Do not mask the interrupt 1 Mask the interrupt

12.3.4.30 THRESHOLD_INP_BUF_1 Register

Figure 12-39 shows the threshold register for input buffer.

Offset: 0x234 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INP_BUF_p2_hi								INP_BUF_p2_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP_BUF_p1_hi								INP_BUF_p1_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 12-39. Threshold input buffer 1 Register (THRESHOLD_INP_BUF_1)

Table 12-34. THRESHOLD_INP_BUF_1 field descriptions

Field	Description
0–7 INP_BUF_p2_hi	High Threshold for input buffer for blend stage 2.
8–15 INP_BUF_p2_lo	Low Threshold for input buffer for blend stage 2.
16–23 INP_BUF_p1_hi	High Threshold for input buffer for blend stage 1 (background).
24–31 INP_BUF_p1_lo	Low Threshold for input buffer for blend stage 1 (background plane).

12.3.4.31 THRESHOLD_INP_BUF_2 Register

Figure 12-40 represents the threshold register for input buffer for plane 3 and plane 4.

Offset: 0x238 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INP_BUF_p4_hi								INP_BUF_p4_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP_BUF_p3_hi								INP_BUF_p3_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 12-40. THRESHOLD_INP_BUF_2 Register

Table 12-35. THRESHOLD_INP_BUF_2 field descriptions

Field	Description
0–7 INP_BUF_p4_hi	High Threshold for input buffer for blend stage 4.
8–15 INP_BUF_p4_lo	Low Threshold for input buffer for blend stage 4.
16–23 INP_BUF_p3_hi	High Threshold for input buffer for blend stage 3.
24–31 INP_BUF_p3_lo	Low Threshold for input buffer for blend stage 3.

12.3.4.32 LUMA Component Register

Figure 12-41 represents the LUMA component register.

Offset: 0x23C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Y_GREEN[5:9]					0	Y_BLUE									
W																
Reset	1	0	1	0	0	0	1	0	0	1	0	1	0	1	0	0

Figure 12-41. LUMA Component Register

Table 12-36. LUMA Component Register field descriptions

Field	Description
0–9 Y_RED	Luminance Coefficient for Red Matrix
11–20 Y_GREEN	Luminance Coefficient for Green Matrix
22–31 Y_BLUE	Luminance Coefficient for Blue Matrix

12.3.4.33 Red Chroma Components

Figure 12-42 represents the Red Chroma component register.

Offset: 0x240 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	Cb_RED											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-42. Red Chroma Component Register

Table 12-37. Red Chroma Component Register field descriptions

Field	Description
5–15 Cr_RED	Cr Coefficient for Red Matrix
20–31 Cb_GREEN	Cb Coefficient for Red Matrix

12.3.4.34 Green Chroma Component Register

Figure 12-43 represents the Green Chroma component register

Offset: 0x244 Access: User read/write

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

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

Figure 12-43. Green Chroma Component Register

Table 12-38. Green Chroma Component Register field descriptions

Field	Description
5–15 Cr_GREEN	Cr Coefficient for Green Matrix
20–31 Cb_GREEN	Cb Coefficient for Green Matrix

12.3.4.35 Blue Chroma Component Register

Figure 12-44 represents the Blue Chroma component register.

Offset: 0x248 Access: User read/write

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

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

Figure 12-44. BLUE Chroma component register

Table 12-39. BLUE Chroma component Register field descriptions

Field	Description
5–15 Cr_BLUE	Cr Coefficient for Blue Matrix
20–31 Cb_BLUE	Cb Coefficient for Blue Matrix

12.3.4.36 CRC_POS Register

Figure 12-45 represents the CRC_POS register.

Offset: 0x24C Access: User read/write

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

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

Figure 12-45. CRC_POS Register

Table 12-40. CRC_POS field descriptions

Field	Description
0–31 CRC_POS	CRC position value calculated for safety enabled layers to be presented to the software for comparison

12.3.4.37 FG0_FCOLOR Register

Figure 12-46 represents the FG0_fcolor register.

Offset:

0x250 (FG0_FCOLOR)	0x290 (FG8_FCOLOR)
0x258 (FG1_FCOLOR)	0x298 (FG9_FCOLOR)
0x260 (FG2_FCOLOR)	0x2A0 (FG10_FCOLOR)
0x268 (FG3_FCOLOR)	0x2A8 (FG11_FCOLOR)
0x270 (FG4_FCOLOR)	0x2B0 (FG12_FCOLOR)
0x278 (FG5_FCOLOR))	0x2B8 (FG13_FCOLOR)
0x280 (FG6_FCOLOR)	0x2C0 (FG14_FCOLOR)
0x288 (FG7_FCOLOR)	0x2C8 (FG15_FCOLOR)

Access: User read/write

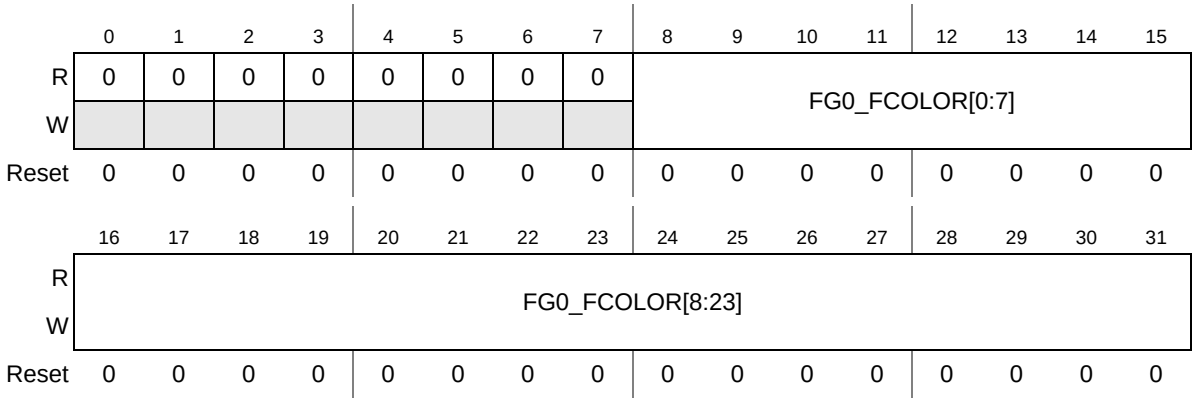


Figure 12-46. FG0_fcolor Register

Table 12-41. FG0_fcolor field descriptions

Field	Description
8–31 FG0_FCOLOR	Foreground color for layer FG0 for pre-blending engine

12.3.4.38 FG0_bcolor

Figure 12-47 represents the FG0_bcolor register.

Offset:

0x254 (FG0_BCOLOR)	0x294 (FG8_BCOLOR)
0x25C (FG1_BCOLOR)	0x29C (FG9_BCOLOR)
0x264 (FG2_BCOLOR)	0x2A4 (FG10_BCOLOR)
0x26C (FG3_BCOLOR)	0x2AC (FG11_BCOLOR)
0x274 (FG4_BCOLOR)	0x2B4 (FG12_BCOLOR)
0x27C (FG5_BCOLOR))	0x2BC (FG13_BCOLOR)
0x284 (FG6_BCOLOR)	0x2C4 (FG14_BCOLOR)
0x28C (FG7_BCOLOR)	0x2CC (FG15_BCOLOR)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	FG0_BCOLOR[0:7]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FG0_BCOLOR[8:23]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-47. FG0_bcolor Register

Table 12-42. FG0_bcolor field descriptions

Field	Description
8–31 FG0_BCOLOR	Background color for layer FG0 for pre-blending engine

12.3.4.39 Global Protection Register

Figure 12-48 represents the Global Protection register.

Offset: 0x300

Access: User read/write

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

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

Figure 12-48. Global Protection Register

Table 12-43. Global Protection Register field descriptions

Field	Description
0 HLB	Hard Lock Bit. This bit cannot be cleared once it is set by software. It can only be cleared by a system reset. 0 All SLBs are accessible and can be modified 1 All SLBs are write-protected and cannot be modified

12.3.4.40 Soft Lock Bit Register L0

Figure 12-49 represents the Soft Lock Bit Register for Layer0. This is used to protect the seven control descriptor layer registers for Layer0.

Offset: 0x304 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SLB_L0_1	SLB_L0_2	SLB_L0_3	SLB_L0_4	0	0	0	0	SLB_L0_5	SLB_L0_6	SLB_L0_7	0
W	WEN_LO_1	WEN_LO_2	WEN_LO_3	WEN_LO_4					WEN_LO_5	WEN_LO_6	WEN_LO_7					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 12-49. Soft Lock Register L0

Table 12-44. Soft Lock Register L0 field descriptions

Field	Description
0 WEN_L0_1	Write Enable for Soft Lock Bit SLB_L0_1 0 SLB is not modified 1 Value is written to SLB
1 WEN_L0_2	Write Enable for Soft Lock Bit SLB_L0_2 0 SLB is not modified 1 Value is written to SLB
2 WEN_L0_3	Write Enable for Soft Lock Bit SLB_L0_3 0 SLB is not modified 1 Value is written to SLB
3 WEN_L0_4	Write Enable for Soft Lock Bit SLB_L0_4 0 SLB is not modified 1 Value is written to SLB
4 SLB_L0_1	Soft Lock Bit for Control Desc L0_1 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

Table 12-44. Soft Lock Register L0 field descriptions (continued)

Field	Description
5 SLB_L0_2	Soft Lock Bit for Control Desc L0_2 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
6 SLB_L0_3	Soft Lock Bit for Control Desc L0_3 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
7 SLB_L0_4	Soft Lock Bit for Control Desc L0_4 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
8 WEN_L0_5	Write Enable for Soft Lock Bit SLB_L0_5 0 SLB is not modified 1 Value is written to SLB
9 WEN_L0_6	Write Enable for Soft Lock Bit SLB_L0_6 0 SLB is not modified 1 Value is written to SLB
10 WEN_L0_7	Write Enable for Soft Lock Bit SLB_L0_7 0 SLB is not modified 1 Value is written to SLB
12 SLB_L0_5	Soft Lock Bit for Control Desc L0_5 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
13 SLB_L0_6	Soft Lock Bit for Control Desc L0_6 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
14 SLB_L0_7	Soft Lock Bit for Control Desc L0_7 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

12.3.4.41 Soft Lock Bit Register L1

Figure 12-50 represents the Soft Lock Bit Register for Layer1. This is used to protect the seven control descriptor layer registers for Layer1.

Offset: 0x308 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SLB_L1_1	SLB_L1_2	SLB_L1_3	SLB_L1_4	0	0	0	0	SLB_L1_5	SLB_L1_6	SLB_L1_7	0
W	WEN_L1_1	WEN_L1_2	WEN_L1_3	WEN_L1_4					WEN_L1_5	WEN_L1_6	WEN_L1_7					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 12-50. Soft Lock Register L1

Table 12-45. Soft Lock Register L1 field descriptions

Field	Description
0 WEN_L1_1	Write Enable for Soft Lock Bit SLB_L1_1 0 SLB is not modified 1 Value is written to SLB
1 WEN_L1_2	Write Enable for Soft Lock Bit SLB_L1_2 0 SLB is not modified 1 Value is written to SLB
2 WEN_L1_3	Write Enable for Soft Lock Bit SLB_L1_3 0 SLB is not modified 1 Value is written to SLB
3 WEN_L1_4	Write Enable for Soft Lock Bit SLB_L1_4 0 SLB is not modified 1 Value is written to SLB
4 SLB_L1_1	Soft Lock Bit for Control Desc L1_1 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
5 SLB_L1_2	Soft Lock Bit for Control Desc L1_2 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
6 SLB_L1_3	Soft Lock Bit for Control Desc L1_3 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
7 SLB_L1_4	Soft Lock Bit for Control Desc L1_4 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
8 WEN_L1_5	Write Enable for Soft Lock Bit SLB_L1_5 0 SLB is not modified 1 Value is written to SLB

Table 12-45. Soft Lock Register L1 field descriptions (continued)

Field	Description
9 WEN_L1_6	Write Enable for Soft Lock Bit SLB_L1_6 0 SLB is not modified 1 Value is written to SLB
10 WEN_L1_7	Write Enable for Soft Lock Bit SLB_L1_7 0 SLB is not modified 1 Value is written to SLB
12 SLB_L1_5	Soft Lock Bit for Control Desc L1_5 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
13 SLB_L1_6	Soft Lock Bit for Control Desc L1_6 Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
14 SLB_L1_7	Soft Lock Bit for Control Desc L1_7 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

12.3.4.42 Soft Lock DISP_SIZE Register

Figure 12-51 represents the Soft Lock DISP_SIZE register.

Offset: 0x30C Access: User read/write

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

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

Figure 12-51. Soft Lock DISP_SIZE Register

Table 12-46. Soft Lock DISP_SIZE Register field descriptions

Field	Description
0 WEN_DISP	Write Enable for Soft Lock Bit SLB_DISP 0 SLB is not modified 1 Value is written to SLB
4 SLB_DISP	Soft Lock Bit for DISP_SIZE Register. This bit cannot be cleared once set by software. Can only be cleared by system reset. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

12.3.4.43 Soft Lock HSYNC/VSYNC PARA Register

Figure 12-52 represents the Soft Lock HSYNC/VSYNC register.

Offset: 0x310 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0			0	0	0	0	0	0	0	0	0	0
W	WEN_HSYNC	WEN_VSYNC			SLB_HSYNC	SLB_VSYNC										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 12-52. Soft Lock HSYNC/VSYNC PARA Register

Table 12-47. Soft Lock HSYNC/VSYNC PARA Register field descriptions

Field	Description
0 WEN_HSYNC	Write Enable for Soft Lock Bit SLB_HSYNC 0 SLB is not modified 1 Value is written to SLB
1 WEN_VSYNC	Write Enable for Soft Lock Bit SLB_VSYNC 0 SLB is not modified 1 Value is written to SLB
4 SLB_HSYNC	Soft Lock Bit for HSYNC Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
5 SLB_VSYNC	Soft Lock Bit for VSYNC Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

12.3.4.44 Soft Lock POL Register

Figure 12-53 represents the Soft Lock POL Register.

Offset: 0x314 Access: User read/write

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

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

Figure 12-53. Soft Lock POL Register

Table 12-48. Soft Lock POL Register field descriptions

Field	Description
0 WEN_POL	Write Enable for Soft Lock Bit SLB_POL 0 SLB is not modified 1 Value is written to SLB
4 SLB_POL	Soft Lock Bit for SYN_POL Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

12.3.4.45 Soft Lock L0_TRANSP Register

Figure 12-54 represents the Soft Lock L0_TRANSP register.

Offset: 0x318 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0			0	0	0	0	0	0	0	0	0	0
W	WEN_L0_FCOLOR	WEN_L0_BCOLOR			SLB_L0_FCOLOR	SLB_L0_BCOLOR										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 12-54. Soft Lock L0_TRANSP Register

Table 12-49. Soft Lock L0_TRANSP Register field descriptions

Field	Description
0 WEN_L0_FCOLOR	Write Enable for Soft Lock Bit SLB_L0_FCOLOR 0 SLB is not modified 1 Value is written to SLB
1 WEN_L0_BCOLOR	Write Enable for Soft Lock Bit SLB_L0_BCOLOR 0 SLB is not modified 1 Value is written to SLB
4 SLB_L0_FCOLOR	Soft Lock Bit for L0_FCOLOR Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
5 SLB_L0_BCOLOR	Soft Lock Bit for L0_BCOLOR Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

12.3.4.46 Soft Lock L1_TRANSP Register

Figure 12-55 represents the Soft Lock L1_TRANSP register.

Offset: 0x31C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0			0	0	0	0	0	0	0	0	0	0
W	WEN_L1_FCOLOR	WEN_L1_BCOLOR			SLB_L1_FCOLOR	SLB_L1_BCOLOR										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 12-55. Soft Lock L1_TRANSP Register

Table 12-50. Soft Lock L0_TRANSP Register field descriptions

Field	Description
0 WEN_L1_FCOLOR	Write Enable for Soft Lock Bit SLB_L1_FCOLOR 0 SLB is not modified. 1 Value is written to SLB
1 WEN_L1_BCOLOR	Write Enable for Soft Lock Bit SLB_L1_BCOLOR 0 SLB is not modified. 1 Value is written to SLB
4 SLB_L1_FCOLOR	Soft Lock Bit for L1_FCOLOR Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access
5 SLB_L1_BCOLOR	Soft Lock Bit for L1_BCOLOR Register. 0 Associated protected register is not locked and writeable 1 Associated protected register is locked for write access

12.4 Functional description

The DCU is a master on the crossbar switch; it fetches graphic source information directly from memory and dynamically performs blending and bit-blitting operations before delivering data to a TFT LCD panel.

12.4.1 Graphic sources

As the DCU is a master on the crossbar switch, it can access directly any memory or device connected to the crossbar switch as a slave. This includes all on-chip flash, all on-chip RAM, and any slave capable of providing high enough data rates, such as, for example an expanded bus interface or a QuadSPI module.

Therefore, any compatible graphic stored anywhere on-chip or in an accessible interface can be displayed on the connected TFT LCD panel with no further intervention from the CPU, except to program the DCU to fetch and place it. The DCU also includes a dedicated memory to store the graphic for its cursor layer.

12.4.2 TFT LCD panel configuration

The nature and timing of the signals required by TFT LCD panels vary greatly between manufacturers. Therefore, the DCU allows highly flexible and detailed configuration of these signals.

Timing diagrams for TFT LCD panels are typically divided into a horizontal timing chart and a vertical timing chart. See Figure 12-56 for details.

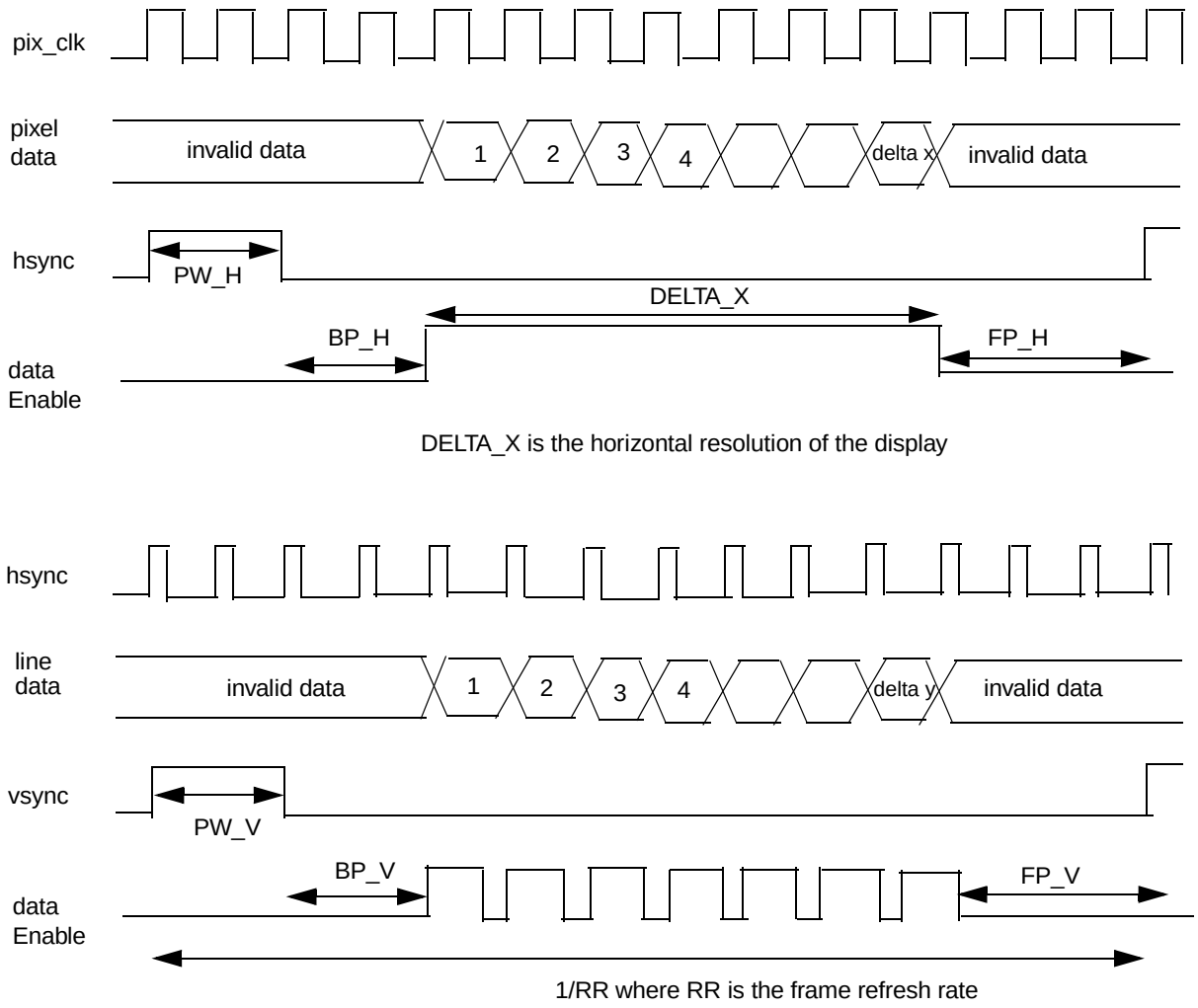


Figure 12-56. HSYNC and VSYNC timing diagram

The number of pixel data slots in the horizontal timing diagram is defined by the width of the panel. The number of line data slots is defined by the height of the panel. Both of these values are defined in the DISP_SIZE register (DELTA_X, DELTA_Y). The width of the panel must always be defined as a multiple of 16.

The timing of the pixel clock is defined by the DIV_RATIO register and the frequency of the clock supplied to the DCU.

In addition to defining the number and timing of pixels in each line and the number of lines, it is normal for TFT LCD panel manufacturers to define other timing signals in terms of pixel clock periods or of the number of horizontal lines. The DCU also follows this convention.

If the TFT LCD panel requires a horizontal synchronizing signal (HSYNC) and/or a data enable signal, then these can be configured using the fields in the HSYN_PARA register. HSYNC provides a pulse to give the panel notice that the next line of pixel data is about to start, and the data enable signal indicates when that data is present. The PW_H bit field indicates the width of the HSYNC pulse, in pixel data clock periods. The BP_H bit field defines the delay between the end of the HSYNC pulse and the start of the data enable signal (and pixel data delivery), in pixel clock periods. The FP_H bit field defines the delay between the end of the data enable signal (and pixel data delivery) and the next HSYNC pulse, in pixel clock periods. FP_H and BP_H have minimum values of 1.

If the TFT LCD panel requires a vertical synchronizing signal (VSYNC), then this can be configured using the fields in the VSYN_PARA register. VSYNC provides a pulse to give the panel notice that the next frame of pixel data lines is about to start, and the panel defines delays before and after this pulse, in terms of pixel clock periods. The PW_V bit field indicates the width of the VSYNC pulse in horizontal line periods. The BP_V bit field defines the delay between the end of the VSYNC pulse and the start of the next pixel data (data enable signal), in horizontal line periods. The FP_V bit field defines the delay between the end of the last pixel data (data enable signal) and the next VSYNC pulse, in horizontal line periods. FP_V and BP_V have minimum values of 1.

The polarity of all these signals, including the pixel data itself, may be inverted by using the control bits in the SYN_POL register.

The refresh rate for the panel can be calculated using Equation 12-1 and Equation 12-2 below.

$$RR = \frac{pix\_clk}{(DELTA\_X + FP\_H + PW\_H + BP\_H) \times (DELTA\_Y + FP\_V + PW\_V + BP\_V)} \qquad \textit{Eqn. 12-1}$$

where: pix_clk is the pixel clock

DELTA_X is the horizontal resolution (in pixels)

DELTA_Y is the vertical resolution (in pixels)

FP_H is the hsync front porch pulse width (in pixel clock cycles)

BP_H is the hsync back porch pulse width (in pixel clock cycles)

PW_H is the hsync active pulse width (in pixel clock cycles)

FP_V is the vsync front porch pulse width (in pixel clock cycles)

BP_V is the vsync back porch pulse width (in pixel clock cycles)

PW_V is the vsync active pulse width (in pixel clock cycles)

Pixel Clock = (DCU Clock) / PRESCALE VALUE *Eqn. 12-2*

where PRESCALE VALUE is an integer value that can range from 2–32.

12.4.3 DCU mode selection and background color

Once the DCU is configured for use with a particular TFT LCD panel, it can be enabled for use. There are five modes to choose from, as shown in Table 12-51.

Table 12-51. List of DCU operating modes

Mode	DCU_MODE[1:0]	PDI_EN	Description
Off	00	X	DCU disabled; the TFT LCD panel is not driven.
Color bar	11	X	DCU displays a test pattern consisting of vertical bands of programmable color.
Normal	01	0	DCU blends layers and displays result on TFT LCD panel.
PDI normal	01	1	As normal mode, except that the panel timing is defined by the input on the PDI interface, and the background color is replaced by the content provided on the PDI interface.
PDI slave	01	0	The DCU synchronizes its timing to an external signal when PDI_SLAVE_MODE is enabled.

The DCU_MODE, PDI_EN and PDI_SLAVE_MODE control bits are in the DCU_MODE register. The DCU has an interface enable bit for the TFT LCD panel interface called RASTER_EN, also in the DCU_MODE register. When RASTER_EN is 0 the raster scanning of pixels to the panel is disabled but the pixel clock continues to run as long as DCU_MODE is in an active state.

Color bar mode is intended for testing the interface between the DCU and the TFT LCD panel. In this mode, the panel is divided into eight vertical strips of equal width, and the strips display a single color whose RGB value is specified in the COLBAR_1 to COLBAR_8 registers. At reset, the colors are set to black, blue, cyan, green, yellow, red, magenta, and white, where positive logic for the RGB values is assumed. The mode can be used to verify correct connection of the interface to the DCU and correct timing configuration of the interface. In this mode, any layer configuration settings are ignored.

In Normal mode, the DCU operates according to the timings specified in Section 12.4.2, TFT LCD panel configuration, and displays graphics according to the configuration of its layers. The BGND register sets the RGB color of the background shown when no other layers are present. This background color is included in the layer blending process but, since it is always the background, it does not include any layer blending settings.

In PDI normal mode, the DCU adopts the timing provided on the PDI interface and replaces the background color by the pixel data coming from the PDI interface. The timing values set in the DCU are ignored in this mode, and the pixel clock and synchronization signals are taken from the PDI interface and passed to the TFT LCD panel. The content of the panel is a combination of the incoming pixel stream and layers generated by the DCU.

PDI Slave mode allows the DCU to synchronize with the external timing signals on the PDI input.

12.4.4 Proper sequence for enabling and disabling the DCU

It is important to follow a correct sequence when enabling and disabling the DCU.

To enable the panel it is possible to set DCU_MODE to be active and then set RASTER_EN. It is possible to set RASTER_EN and then set DCU_MODE to be active. It is also possible to set both in the same write.

It is not possible to set DCU_MODE to be active and then set it to 0 before the RASTER_EN bit has been set.

To disable the panel MCU_MODE must be set to 0 in the same write as or before RASTER_EN can be set to 0.

It is not possible to set RASTER_EN to 0 (disable raster) before disabling the pixel clock.

12.4.5 Layer configuration and blending

Users control the graphical content of the TFT panel by manipulating the configuration of elements in the DCU called layers. Each layer has a control descriptor that defines the size, position, memory encoding, blending, and memory location of the graphic to be displayed. The DCU provides 16 independent layers that are identical except that they have a fixed priority with respect to each other, and this affects how individual pixels are blended when layers overlap. The blending setting on each layer allows the pixels on that layer to be opaque, partially transparent, or fully transparent, which allows them to combine with pixels on other layers that they overlap.

12.4.5.1 Blending priority of layers

The 16 layers available in the DCU are each fixed in priority order, with layer 0 being the highest priority, layer 1 being the second highest priority, and so on until layer 15, which is the lowest priority. The priority is used by the DCU to define how to blend individual pixels within the layers. For example, if layer 0 is defined as not being blended with other layers and a pixel on layer 0 overlaps a pixel on layer 1 then the pixel on layer 0 will be visible on the panel unchanged by the pixel on layer 1. However, if layer 0 is defined as being partially transparent, then the DCU will blend the overlapping pixel such that the result is a combination of the pixel on layer 0 and the pixel on layer 1. It is possible to blend up to four layers at each pixel position.

As there is a maximum number of layers that can blended together, then any pixel on a layer that is lower than the threshold priority will not be included in any blend. If a pixel is on a layer that has the lowest priority in any blending scheme, then the blending settings for that pixel are ignored and the pixel is treated as a background pixel. This means that a lower priority layer may have some pixels completely obscured by those on higher priority layers on one part of the panel, and some other pixels visible or blended on other parts of the panel.

Figure 12-57 shows how the pixel blend takes place inside the DCU. The priority of the layers determines at which stage of the blend the pixel enters. Any pixels lower than the threshold priority are ignored and, as can be seen, the blend settings for the lowest priority pixel is also ignored. The maximum number of pixels in the blend is configured by the BLEND_ITER bit field in the DCU_MODE register. As can be seen in the figure, the blending process is iterative so that four-pixel blending takes more DCU clock cycles than three-pixel blending, and three-pixel blending takes more DCU clock cycles than two-pixel blending.

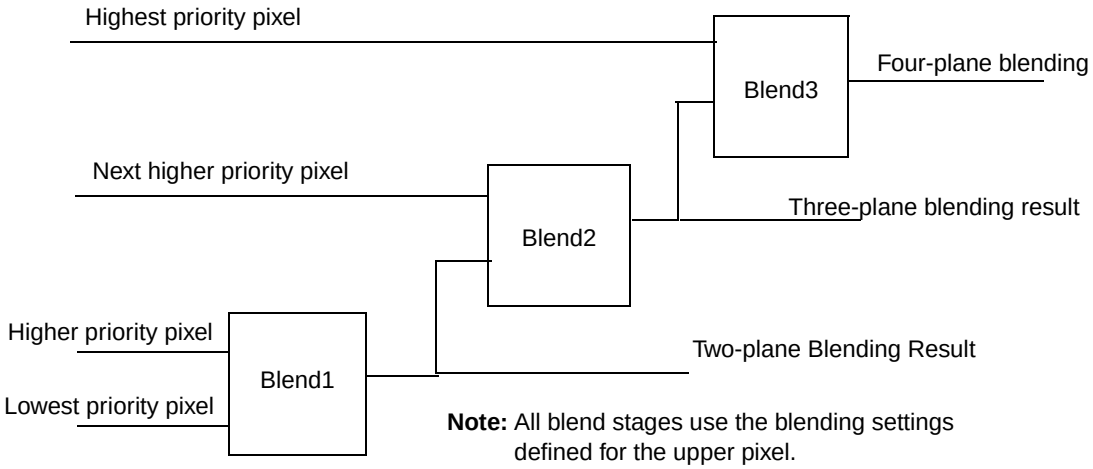


Figure 12-57. Pixel blending stack

The blending algorithm used for each color component is shown in Equation 12-3.

This priority concept is illustrated in Figure 12-58 and Figure 12-59. In this case, there are five layers enabled, and each contains a graphic that is a solid rectangular block of a single color. The size and shape of each layer is different. The background color of the panel is set to grey and layers have been placed such that they overlap each other.

Figure 12-58 shows the individual source graphics and the case where no layer has any blending enabled. Here, the highest priority layer (in this case layer 0) is fully visible. Layer 1 is visible where layer 0 does not overlap it. Layer 2 is visible where layer 1 does not overlap it. Layer 3 is overlapped by layers 0 and 1 and so is only partially visible. Layer 4 is partially obscured by all of the other layers. Note that layer 4 is higher priority than the background color.

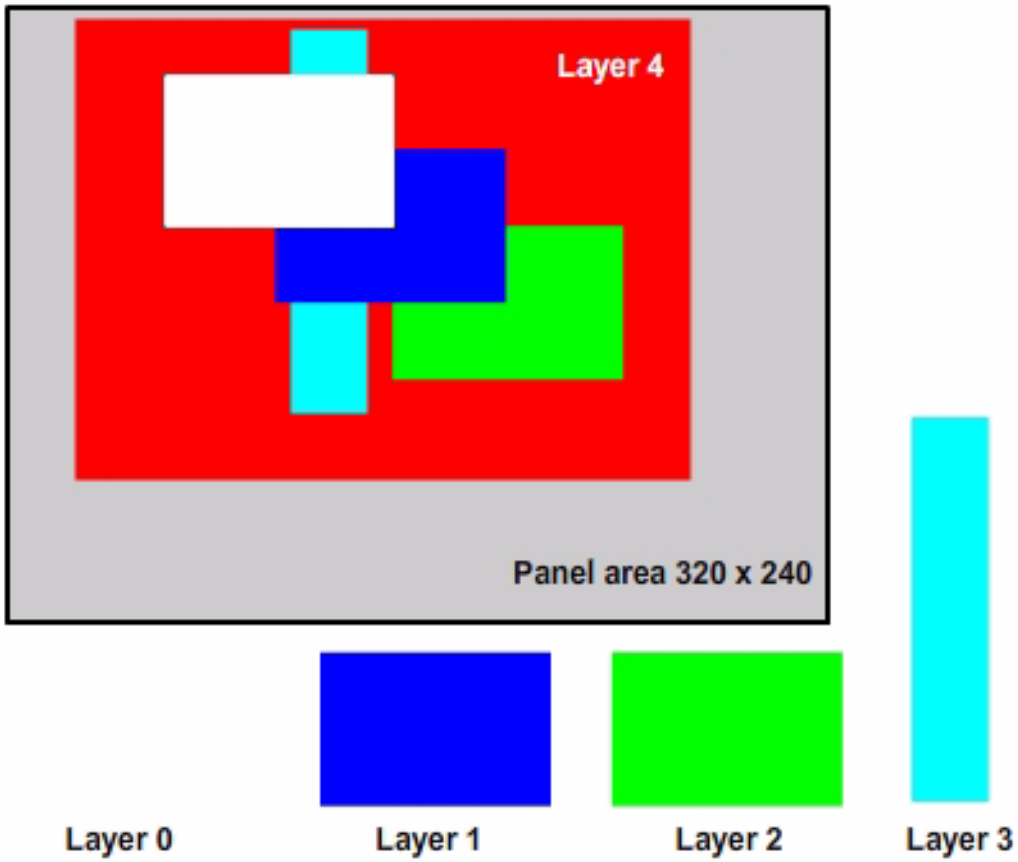


Figure 12-58. Example of layer placement with no blending

Figure 12-59 shows the same layer configuration except that, in this case, the layers have been made 50% transparent and the depth of the pixel blend is set to 3. The pixels in layer 0 are now blended with pixels in the underlying layers. In particular, note region A where layer 0 is blended with layer 4 and the background color. This blending effect is repeated across all of the layers; however, note the pixels in region B. In this region the pixels from layer 0 are blended with those on layer 1 and layer 2; however, the pixels from layer 4 and the background color are not included in this blend. This is because the DCU is configured to blend three layers only, and so the blend setting for layer 2 is ignored for those pixels in region B.

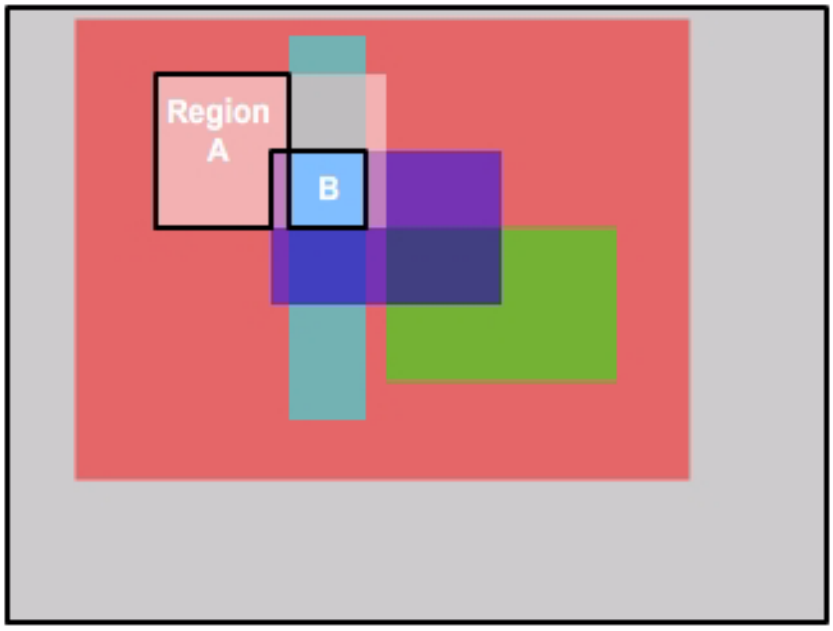


Figure 12-59. Example of layer placement with 3-layer blending

All blending is performed using full 8-bits-per-component colors. The DCU automatically performs a color promotion on source data that is stored in less than RGB888 color.

12.4.5.2 Control Descriptors

The control descriptor for each layer consists of seven registers, and all 16 control descriptors are identical except the two highest priority layers, which have additional control bits for the safety mode.

The control descriptors may be written to at any time, and the value present in the registers at the start of the next frame refresh cycle defines the content of the panel for that frame. To avoid coherency issues, ensure all control descriptor changes are made before the PROG_END bit in the INT_STATUS register is asserted.

12.4.5.3 Layer size and positioning

The size of each layer is defined by register 1 in the control descriptor for the layer (CTRLDESCLn_1, where n is the layer number). The register contains two bit fields, HEIGHT and WIDTH, which determine the size and shape of the layer. Both fields are expressed in terms of the number of pixels in each dimension.

The HEIGHT bit field may take any value; however, it may not be useful to define a value larger than the height of the panel.

The WIDTH field has a restriction on the value it can take, depending on the data format of the graphic specified by the layer. This field must always be an integer multiple of the number of pixels that are represented by a 32-bit word except in the special case of 1 bit per pixel where the multiple is 16. The data format can range from 1 bit per pixel to 32 bits per pixel and so there is a range of multiples from 1 to 32. Figure 12-52 shows the multiples for the WIDTH bit field and some correct values.

Table 12-52. Example of WIDTH multiples for different graphic data formats

Data format	WIDTH multiples	Example values
1 bpp	16	16, 32, 48, 64, …
2 bpp	16	16, 32, 48, 64, …
4 bpp	8	8, 16, 24, 32, …
8 bpp	4	4, 8, 12, 16, …
16 bpp	2	2, 4, 6, 8, …
24 bpp	4 (= 3 whole 32-bit words)	4, 8, 12, 16
32 bpp	1	1, 2, 3, 4, …

If the WIDTH bit field is set to an invalid multiple, then the layer configuration is invalid, the layer cannot be made visible, and an error flag is set in the layer parameter error register (PARR_ERR).

The position of each layer on the panel is defined by register 2 in the control descriptor for the layer (CTRLDESCLn_2, where n is the layer number). The register contains two bit fields, POSY and POSX, which determine the location of the upper left pixel of the layer in the x and y axes. Both fields are expressed in terms of the number of pixels in each axis.

There are no restrictions on layer placement. Any layer can be placed and moved to any panel position. If a layer is placed so that pixels would appear beyond the dimensions of the panel, then the DCU displays the pixels on the panel and ignores the pixels off the panel.

12.4.5.4 Graphics and data format

The memory location of the graphic that is displayed on the layer is defined by register 3 in the control descriptor for the layer (CTRLDESCLn_3, where n is the layer number). This 32-bit value can contain the address of any memory location in the memory map of the MCU.

The format of the data that describes the graphic is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 12.4.5.3, Layer size and positioning). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

There are five formats where the RGB values of the pixels are stored directly in the graphic. In these formats, the DCU treats the data as describing a true RGB color. The formats are:

- ARGB8888, where the data defines 8-bit values for the red, green, blue, and alpha components of the image.
- RGB888, where the data defines 8-bit values for the red, green, and blue components of the image.
- RGB565 where the data defines 5-bit values for the red and blue components, and 6-bit values for the green component of the image.
- ARGB1555 where the data defines 5-bit values for the red, green, and blue components, and a 1-bit value for the alpha channel of the image.
- ARGB4444, where the data defines 4-bit values for the red, green, blue, and alpha components of the image.

The three 16-bit formats (RGB565, ARGB1555, and ARGB4444) are promoted to full 8 bit per component format by shifting the bits left so that the MSB of the component in the 16-bit format becomes the MSB of the 24/32 bpp (bit per pixel) format, and the LSB is filled with the value of the MSB. For example, an RGB565 value of 10000:010000:11111 becomes 10000111:0100000000:11111111.

There are four indexed color formats where the data in the graphic does not define the RGB color to display. Instead, the data defines the entry in a color look-up table (CLUT) that contains a palette of RGB colors. The maximum number of colors in the CLUT is defined by the size of the data stored in the graphic. For 1 bpp graphics, there is a maximum of two colors in the CLUT. For 2 bpp, there is a maximum of four colors. For 4 bpp and 8 bpp data, the maximums are 16 and 256 colors, respectively. The address of the first value in the CLUT is defined in the LUOFFS bit field of register 4 and the CLUT is the RAM block dedicated to the DCU which is described in Section 12.4.7, CLUT/Tile RAM. Since the RGB values stored in the CLUT are 24-bit RGB, there is no need for further adjustment before blending.

There are four additional formats defined by the BPP bit field. These configure the graphic in transparency mode and luminance mode (see Section 12.4.5.6, Transparency mode and blending, and Section 12.4.5.7, Luminance mode, respectively).

There is a set storage format for each data format provided by the DCU. These formats can be seen in Table 12-53 to Table 12-59.

For 32 bpp format, data expected in the memory is in the form ARGB8888.

Table 12-53. Data layout for 32 bpp

Address offset	[7:0]	[15:8]	[23:16]	[31:24]
0x00	B0	G0	R0	A0
0x04	B1	G1	R1	A1
0x08	B2	G2	R2	A2

For 24 bpp format, data expected in the memory is in the form RGB888.

Table 12-54. Data Layout for 24 bpp

Address offset	[7:0]	[15:8]	[23:16]	[31:24]
0x00	B0	G0	R0	B1
0x04	G1	R1	B2	G2
0x08	R2	B3	G3	R3

For 16 bpp, data expected is in the form of RGB565, ARGB1555 or ARGB4444.

Table 12-55. Data Layout for 16 bpp

Address offset	[7:0]	[15:8]	[23:16]	[31:24]
0x00	pixel0[15:8]	pixel0[7:0]	pixel1[15:8]	pixel1[7:0]
0x04	pixel2[15:8]	pixel2[7:0]	pixel3[15:8]	pixel3[7:0]
0x08	pixel4[15:8]	pixel4[7:0]	pixel5[15:8]	pixel5[7:0]

For 8 bpp, data is in the form shown in Table 12-56.

Table 12-56. Data Layout for 8 bpp

Address offset	[7:0]	[15:8]	[23:16]	[31:24]
0x00	pixel0[7:0]	pixel1[7:0]	pixel2[7:0]	pixel3[7:0]
0x04	pixel4[7:0]	pixel5[7:0]	pixel6[7:0]	pixel7[7:0]
0x08	pixe8[7:0]	pixel9[7:0]	pixel10[7:0]	pixel11[7:0]

For 4 bpp, data is in the form shown in Table 12-57.

Table 12-57. Data Layout for 4 bpp

Address offset	[3:0]	[7:4]	[11:8]	[15:12]	[19:16]	[23:20]	[27:24]	[31:28]
0x00	pixel1[3:0]	pixel0[3:0]	pixel3[3:0]	pixel2[3:0]	pixel5[3:0]	pixel4[3:0]	pixel7[3:0]	pixel6[3:0]
0x04	pixel9[3:0]	pixel8[3:0]	pixel11[3:0]	pixel10[3:0]	pixel13[3:0]	pixel12[3:0]	pixel15[3:0]	pixel14[3:0]
0x08	pixel17[3:0]	pixel16[3:0]	pixel19[3:0]	pixel18[3:0]	pixel21[3:0]	pixel20[3:0]	pixel23	pixel22

For 2 bpp, data is in the form shown in Table 12-58.

Table 12-58. Data Layout for 2bpp

Address offset	[3:0]	[7:4]	[11:8]	[15:12]	[19:16]	[23:20]	[27:24]	[31:28]
0x00	[pix3,pix2]	[pix1,pix0]	[pix7,pix6]	[pix5,pix4]	[pix11,pix10]	[pix9,pix8]	[pix15,pix14]	[pix13,pix12]
0x04	[pix19,pix18]	pix17,pix16]	[pix23,pix22]	[pix21,pix20]	[pix27,pix26]	[pix25,pix24]	pix31,pix30]	[pix29,pix28]

For 1 bpp, data is in the form shown in Table 12-59.

Table 12-59. Data Layout for 1 bpp

Address offset	[7:0]	[15:8]	[23:16]	[31:24]
0x00	pixel7-pixel0	pixel15-pixel8	pixel23-pixel16	pixel31-pixel24
0x04	pixel39-pixel32	pixel47-pixel40	pixel55-pixel48	pixel63-pixel56

The DCU includes a flag that indicates when it has completed fetching graphics from memory for the current frame refresh. If required, this flag (DMA_TRANS_FINISH in the INT_STATUS register) can be used to determine when changes can be made to the source graphic content.

12.4.5.5 Alpha and Chroma-key blending

The blending configuration of each layer is defined by the BB, AB, and TRANS bit fields in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). The pixels affected by the blending configuration can be further selected by registers 5 and 6 in the control descriptor (CTRLDESCLn_4 and CTRLDESCLn_5). Depending on the priority and placement of the layer (see

Section 12.4.5.1, Blending priority of layers), these bit fields and registers define how pixels from different layers are blended together.

The AB and BB bit fields define whether blending is active and whether the whole graphic or a selected portion is blended. Registers 5 and 6 define the range of RGB colors that define the selected pixels. The TRANS bit field defines the transparency of the selected pixels.

The BB bit field defines whether the whole graphic, or only certain pixels, should be blended. When this bit is set, pixels that have an RGB value that falls into the range defined by registers 5 and 6 are considered to be selected and treated differently to the non-selected pixels in the graphic. This is a process known as chroma-keying since it is the color of the pixel that defines the selection. The selected pixels must be within the range defined by each color component of registers 5 and 6. See Table 12-60 for examples of pixels that are selected and not selected when the given range is defined as 0x0080C0 to 0x0FB0FF.

Table 12-60. Example of how chroma-key range selects pixels

Source pixel	Red 00–0F	Green 80–B0	Blue C0–FF	Comment
0x000000	P	X	X	Not selected
0x08C0C0	P	X	P	Not selected
0x08A0C0	P	P	P	Pixel is selected

The AB bit field defines how any selected and non-selected pixels are blended. By combining this control with the BB bit field it is possible to define 11 unique ways of blending the pixels on a layer dependent on the type of layer. Depending on the configuration defined by the AB and BB bit fields, the TRANS bit field combines the two pixels in every blend stage using the alpha value of the upper pixel (which has the effect of making this pixel more or less transparent and revealing more or less of the lower pixel).

The result of each blend stage is calculated for all three color components as shown in Equation 12-3.

$$A = (LPixel \times (255 - alpha)) + (HPixel \times alpha) \qquad \textit{Eqn. 12-3}$$

where:

- A is a 16-bit value
- LPixel is the lower priority pixel in the blend
- HPixel is the higher priority pixel in the blend
- alpha is the alpha value of the higher priority pixel

The value of A is then normalized back to an 8-bit value by the approximation shown in Equation 12-4:

$$BPixel = A>>8 + 1 \qquad \textit{Eqn. 12-4}$$

where BPixel is the blended pixel output.

The output value for each of the RGB components is therefore obtained by right-shifting A by 8 and then adding 1 to the result.

The blend can apply to pixels with no alpha channel (RGB) or with an alpha channel (ARGB) in different ways.

Table 12-61 defines how the settings of the BB and AB bit fields affect the pixels in the layer; RGB modes are 1 bpp, 2 bpp, 4 bpp, 8 bpp, RGB565, and RGB888; ARGB modes are ARGB1555, ARGB4444, and ARGB8888.

Table 12-61. Blend options for BB and AB configurations

Case	BB	AB[1:0]	Format	Function
1	0	00	RGB	No blending, underlying pixels are obscured
2	1	00	RGB	Selected pixels are completely removed
3	0	01	RGB	The value in TRANS becomes the alpha channel of all pixels on the layer
4	1	01	RGB	The value in TRANS becomes the alpha channel of the selected pixels on the layer
5	0	10	RGB	Same as case 3
6	1	10	RGB	Selected pixels are completely removed and the value in TRANS becomes the alpha channel of the non-selected pixels on the layer
7	0	11	RGB	Reserved
8	1	11	RGB	Reserved
9	0	00	ARGB	No blending, pixel alpha is ignored and underlying pixels are obscured
10	1	00	ARGB	Selected pixels are completely removed, pixel alpha is ignored
11	0	01	ARGB	Pixel alpha is used to blend layer with underlying pixels. Value in TRANS is ignored.
12	1	01	ARGB	Uses the pixel alpha of the selected pixels only to blend layer with underlying pixels. Value in TRANS is ignored.
13	0	10	ARGB	The value in TRANS is multiplied with the pixel alpha value and the resultant alpha is used to blend all the pixels
14	1	10	ARGB	Selected pixels are completely removed, the value in TRANS is multiplied with the pixel alpha value and the resultant alpha is used to blend the non-selected pixels on the layer
15	0	11	ARGB	Reserved
16	1	11	ARGB	Reserved

Figure 12-60 to Figure 12-68 illustrate the effect of the cases identified in Table 12-61. In all cases there is a single active layer and a white background color.

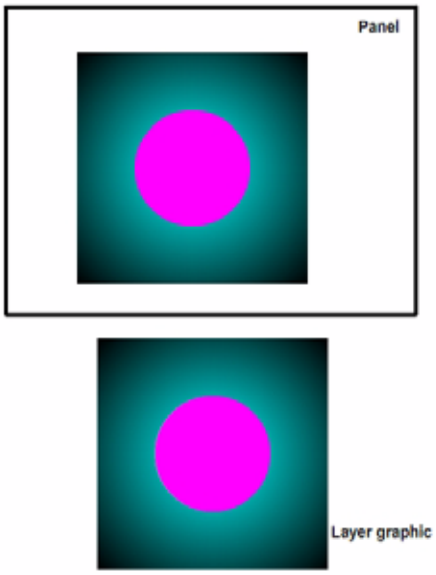


Figure 12-60. Case 1 example (no blend)

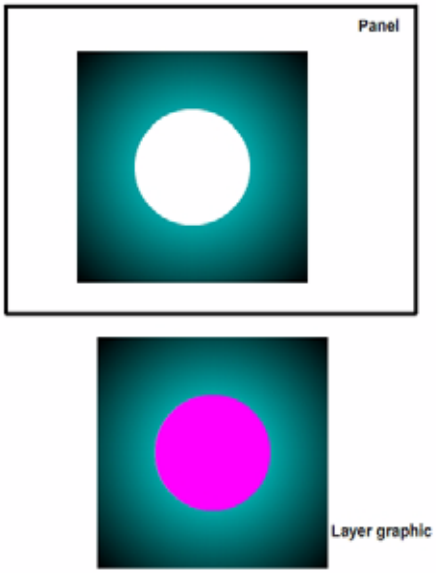


Figure 12-61. Case 2 example (remove selected pixels)

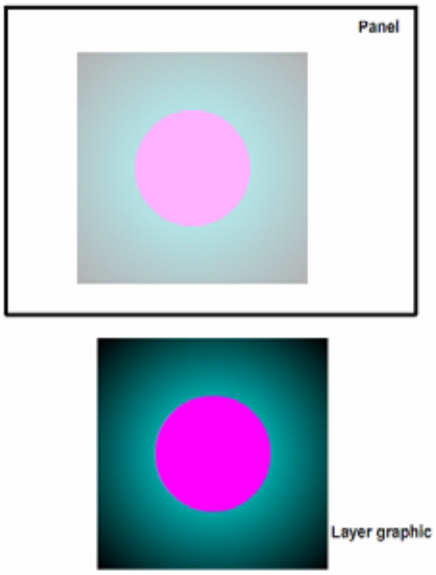


Figure 12-62. Case 3 example (all pixels transparent)

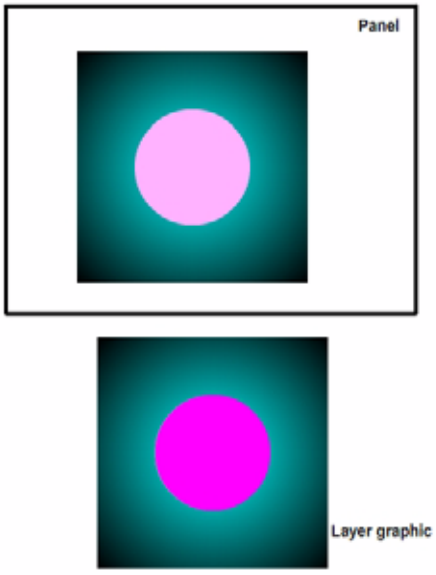


Figure 12-63. Case 4 example (selected pixels transparent)

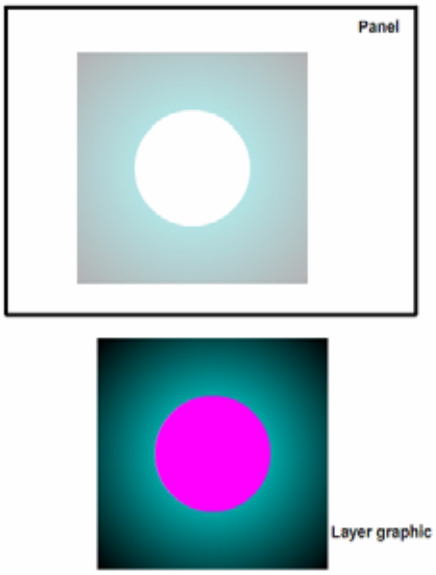


Figure 12-64. Case 6 example (selected pixels removed, others transparent)

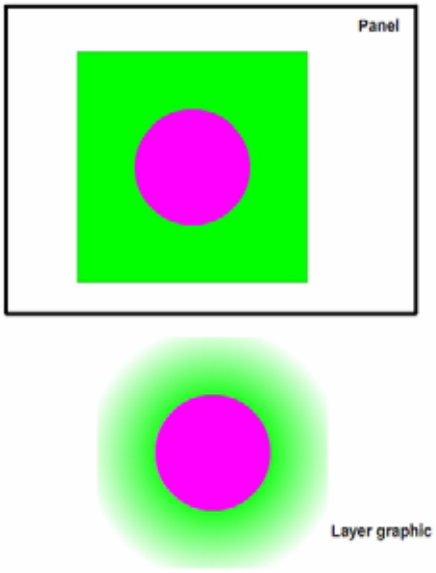


Figure 12-65. Case 9 example (no blend, pixel alpha ignored)

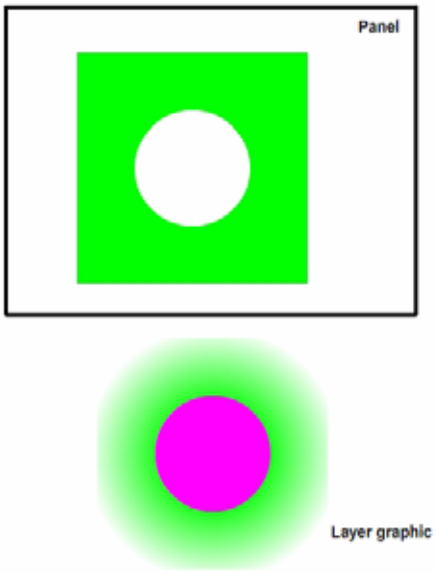


Figure 12-66. Case 10 example (selected pixels removed, pixel alpha ignored)

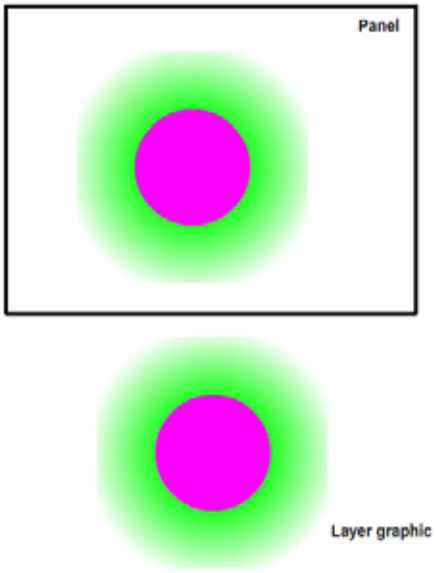


Figure 12-67. Case 13 example (pixel and layer alpha used in blend)

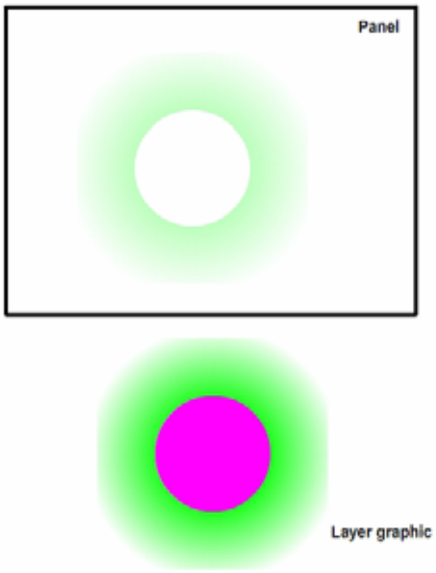


Figure 12-68. Case 14 example (selected pixels removed, pixel and layer alpha used in blend)

12.4.5.6 Transparency mode and blending

Transparency mode is a special case for the graphic data format and is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 12.4.5.3, Layer size and positioning). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

In transparency mode, the source graphic does not contain any direct or indexed color information. Instead, the graphic data represents the alpha channel of the graphic. The DCU creates the final graphic by pre-blending a foreground color and background color using the alpha value of each pixel. The result of this pre-blend can then be blended with pixels on other layers using the normal blending process. Each layer has dedicated registers to contain the foreground and background colors for this mode. These are FGn_fcolor and FGn_bcolor, where n is the layer number.

Transparency mode is typically used when a graphic must blend smoothly into the underlying layers, but where a rich color palette is not required. Examples include text where this mode allows the text to blend smoothly with any background — this is known as anti-aliasing.

There are two transparency modes available: 4 bpp and 8 bpp. The result of the pre-blend can be treated as an RGB888 graphic and blended in a similar way to previously described, or it can be treated as a special case of ARGB with only the foreground color visible in the final blend. Table 12-62 describes the blend options for transparency mode.

Table 12-62. Blend options for transparency mode

Case	BB	AB[1:0]	Mode	Function
1	0	00	Transparency	No blending, underlying pixels are obscured
2	1	00	Transparency	Reserved
3	0	01	Transparency	The value in TRANS becomes the alpha channel of all pixels on the layer
4	1	01	Transparency	The value in TRANS becomes the alpha channel of the selected pixels on the layer
5	0	10	Transparency	Same as case 3
6	1	10	Transparency	Background color is ignored, selected pixels are completely removed, the value in TRANS is multiplied with the graphic data value (alpha) and the resultant alpha is used to blend the non-selected pixels on the layer
7	0	11	Transparency	Reserved
8	1	11	Transparency	Reserved

Figure 12-69–Figure 12-72 illustrate the effect of the cases identified in Table 12-62. In all cases there is a single active transparency layer and a white background color.

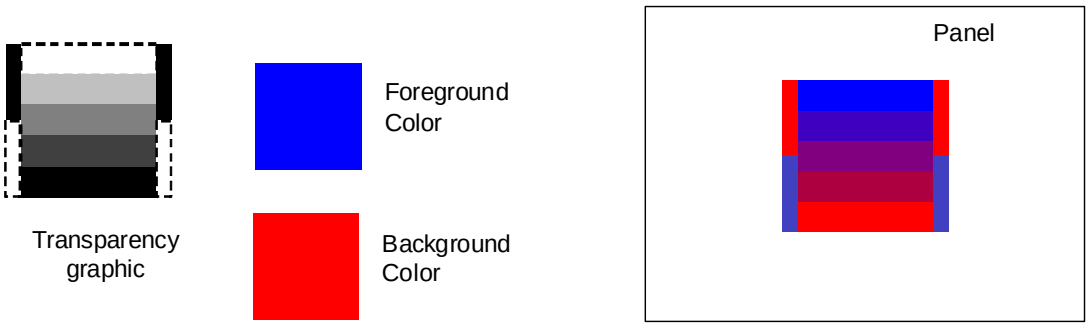


Figure 12-69. Case 1 example (no blend)

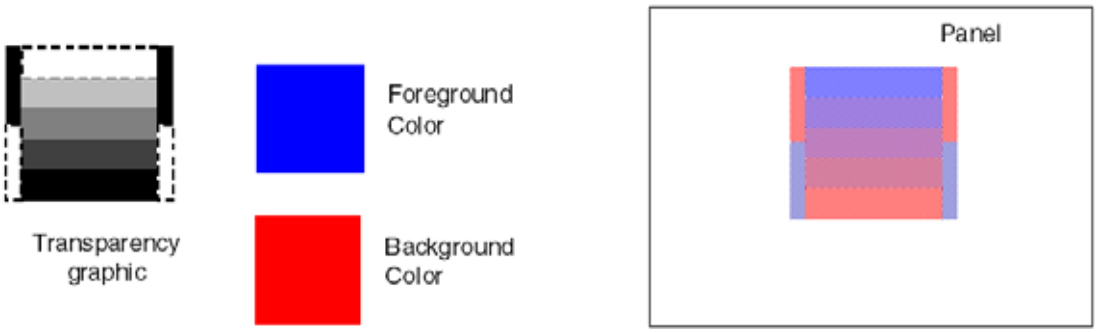


Figure 12-70. Case 3 example (all pixels transparent)

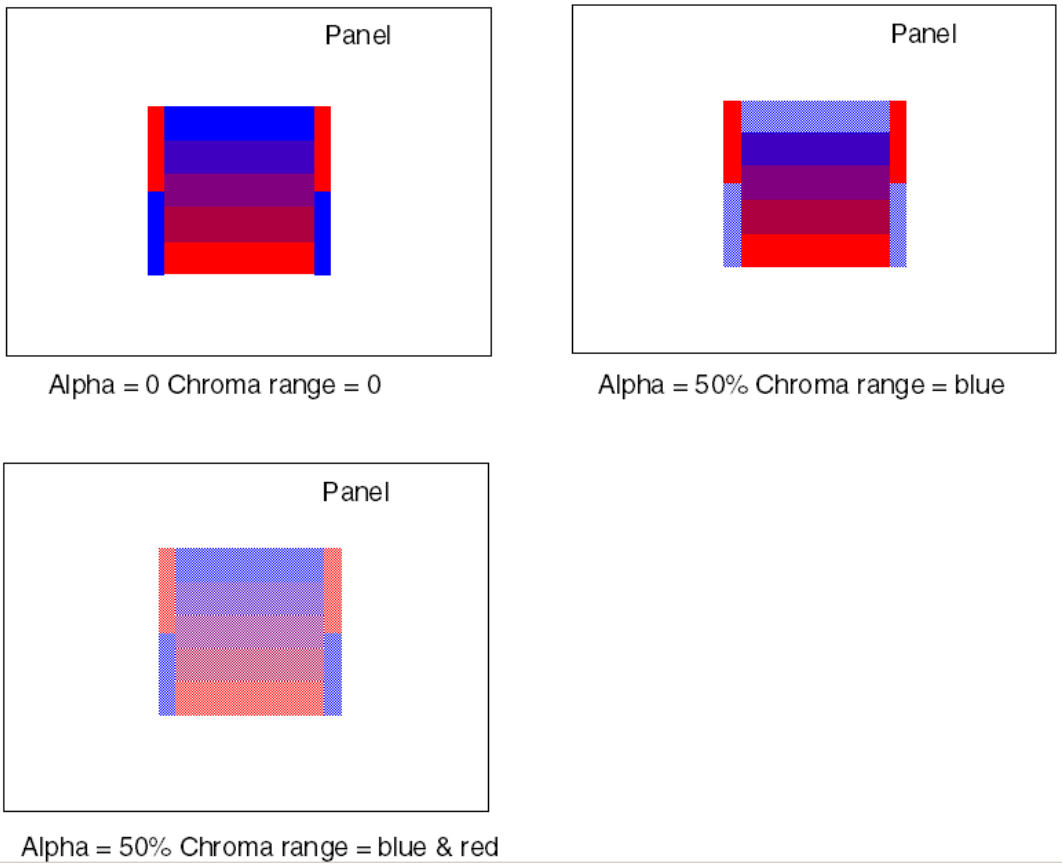


Figure 12-71. Case 4 example (selected pixels transparent)

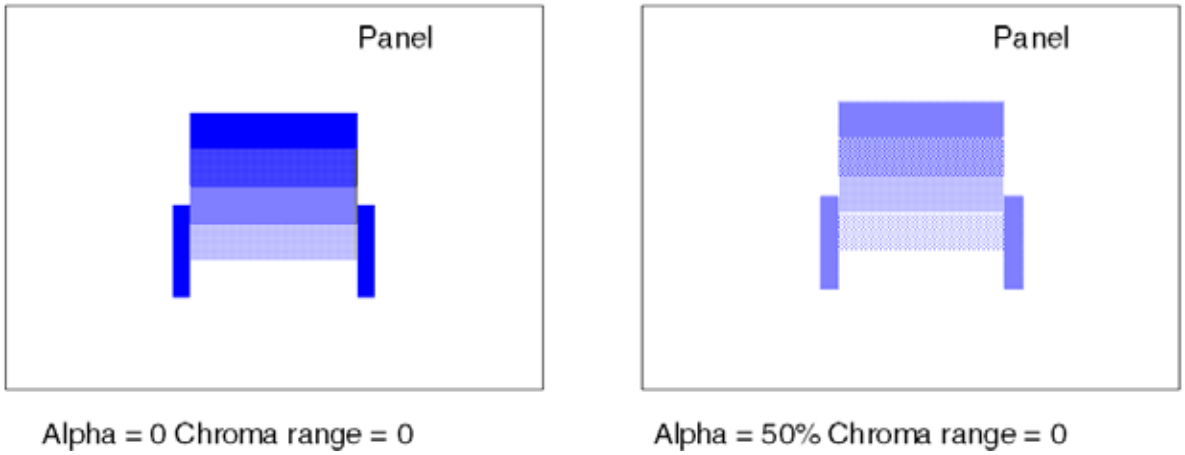


Figure 12-72. Case 6 example (only foreground color blended)

12.4.5.7 Luminance mode

Luminance mode is a special case for the graphic data format and is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 12.4.5.3, Layer size and positioning). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

In luminance mode, the data in the source graphic does not contain any direct or indexed color information or alpha information. The data values in a layer in luminance mode modify the values of the pixels on underlying layers only. There are two luminance modes available: 4 bpp and 8 bpp. In both cases, the data values behave as signed integers that are added to each component of the underlying pixel. The 4 bpp mode is left-shifted to form a signed 8 bpp integer. The results of the addition are prevented from overflowing, so that any result greater than 0xFF is set to 0xFF and any result less than 0x00 is set to 0x00.

The result of a blend with a luminance layer is that the intensity of the underlying pixel's color will be increased or decreased. In this way, luminance mode can be used to highlight or dim pixels on the panel without having to modify the source graphic data. Table 12-63 describes the effect of luminance blends on an underlying pixel.

Table 12-63. Example of a blend with a luminance mode layer

Pixel value	Luminance value	Resultant pixel
0xFF8040	0x40	0xFFC080
0xFF8040	0xC0	0x3F0000

12.4.5.8 Tile mode

Tile mode is a special case for the layer and is enabled by the TILE_EN bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number).

In this mode the layer size register (CTRLDESCLn_1, where n is the layer number) defines the size of the layer; however, the size of the graphic is defined in control register 7 (CTRLDESCLn_7, where n is the

layer number). The size of the graphic must be less than or equal to the size of the layer. When tile mode is enabled, the graphic is repeated horizontally and vertically until it fills the whole layer. The horizontal size of the tile is defined by the TILE_HOR_SIZE bit field and is restricted to be a multiple of 16 pixels. The vertical size of the tile is defined by the TILE_VER_SIZE bit field.

The graphic data for the Tile mode can be fetched either from the system memory or from the internal CLUT/Tile memory. This is defined by the DATA_SEL bit field in register 4 in the control descriptor for the layer. If the graphic is fetched from CLUT/tile memory then it must be in the CLUT/TILE RAM direct color format. Otherwise the graphic can be in any previously described data format. See Figure 12-73 for an example of a layer in tile mode.

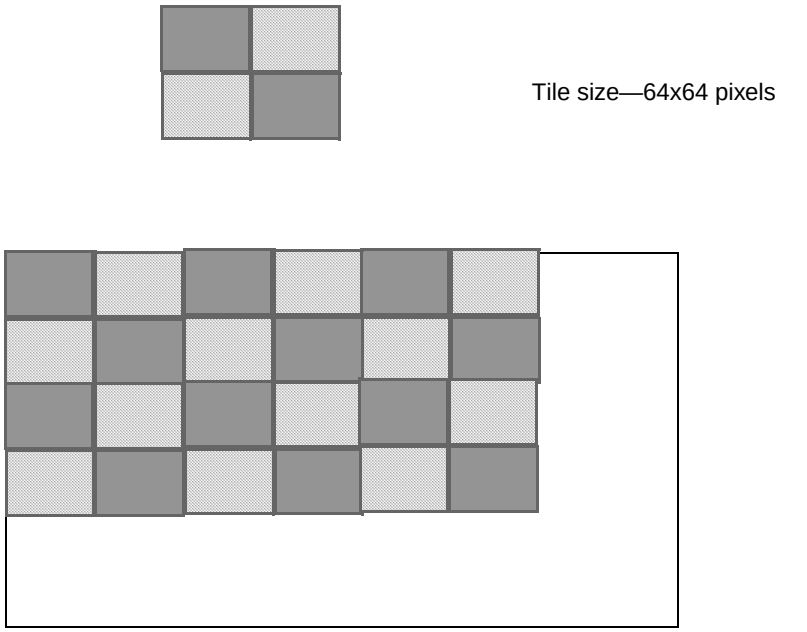


Figure 12-73. Tile mode

When DATA_SEL is set (to use CLUT/TILE RAM) the LUOFFS bitfield defines the start address of the tile graphic.

12.4.6 Hardware cursor

In addition to the 16 layers, the DCU also provides a special layer intended for use as a cursor. This cursor operates in 1 bpp mode and includes its own RAM area to store the graphic. The cursor may be placed at any location on the panel and includes an automatic blink option. The hardware cursor is configured using a dedicated control descriptor and its visible pixels always overlay any other layer on the panel.

The size of the cursor is defined by register 1 in the control descriptor for the cursor (CTRLDESCCURSOR_1). The register contains two bit fields, HEIGHT and WIDTH, which determine the size and shape of the layer. Both fields are expressed in terms of the number of pixels in each

dimension. The HEIGHT is limited to a maximum of 256 pixels, and the total number of pixels cannot exceed the number of bits in the cursor RAM (8192 bits).

Bits in the cursor RAM that are 0 become transparent on the panel. Bits that are 1 become fully opaque in the color defined in register 3 in the control descriptor for the cursor (CTRLDESCCURSOR_3). The DEFAULT_CURSOR_COLOR bit field is in RGB888 format.

There are restrictions on the arrangement of bits in the cursor RAM depending on how the HEIGHT and WIDTH bit fields are configured.

- The rightmost bit in the cursor RAM (bit 31) represents the leftmost pixel on the display.
- When the cursor size is less than 32 bits, each row of the cursor is contained in a single 32-bit word of cursor RAM. The other bits in each row must be filled with zeros.
- When the cursor width is an integer multiple of 32 bits, the pixels in each row roll from one word in the RAM to the next one. The rightmost bit in the first word in the RAM is the top leftmost pixel on the display. The leftmost bit in the word represents a pixel that is adjacent to the rightmost bit in the next word (in the same row). The leftmost pixel on the next row is the rightmost bit in the first word after n words that describe the first row.
- When the cursor is greater than 32 bits but not an integer multiple of 32, the pixels in each row roll from one word into the next one such that the rightmost bit in the first word of the row is the leftmost bit on the display. In the final word of the row there are unused bits.

The position of the cursor on the panel is defined by register 2 in the control descriptor for the cursor (CTRLDESCCURSOR _2). The register contains two bit fields, POSY and POSX, which determine the location of the upper left pixel of the cursor in the x and y axes. Both fields are expressed in terms of the number of pixels in each axis. Placing the cursor beyond the panel area is not allowed.

The cursor can be configured to blink at a particular rate when it is enabled. The EN_BLINK, HWC_BLINK_ON, and HWC_BLINK_OFF bit fields define the blink behavior. These are in register 4 in the control descriptor for the cursor (CTRLDESCCURSOR_4). EN_BLINK enables blinking. The blinking time is based on the frame rate, and the on and off times are independently configurable. HWC_BLINK_ON configures the number of frame refresh cycles for which the cursor is visible. HWC_BLINK_OFF configures the number of frame refresh cycles for which the cursor is not visible. For a frame refresh rate of 64 Hz, the HWC_BLINK_ON and HWC_BLINK_OFF counters give a range of on/off times up to 4 seconds.

The cursor is enabled by setting the CUR_EN bit field in register 3 in the control descriptor for the cursor (CTRLDESCCURSOR_3). Visible pixels in the cursor graphic operate independently from the normal layer blend process and always replace any other pixel at the same panel location. This has the effect of making the cursor the highest priority layer on the panel when it is enabled and where it contains visible pixels.

If the DCU detects an invalid configuration in the cursor control descriptor, then the cursor configuration is invalid and it cannot be made visible. In addition, the error flag HWC_ERR is set in the layer parameter error register (PARR_ERR).

The cursor RAM may be written at any time when the TFT LCD panel is not being driven with data. This means that the RAM can be modified when the DCU is not enabled and during the vertical blanking period.

12.4.7 CLUT/Tile RAM

The internal tile memory and color look up table (CLUT) memory share a common block of RAM internal to the DCU. Color information in this RAM is always stored as aligned 32-bit words where the most-significant byte is always 0x00, the next byte contains the red component, the next the green component and the least significant byte the blue component (0x00RRGGBB).

This memory block can be used to store either color look-up tables or graphics for use as a tile on a layer. The content of the RAM at a specific address is defined by the control descriptor of a layer. The LUOFFS bit field in the layer control descriptor defines the starting address of the area, and the BPP and TILE_EN bit fields define what type of use the RAM area has.

In Figure 12-74 three areas of the RAM are defined for different purposes. Area A is used by layer 1 as a CLUT for its 4 bpp graphic. Area B is use by layer 5 as a store for its tile graphic. Area C is used by layers 2, 7, and 9 as a CLUT for their 8 bpp graphics.

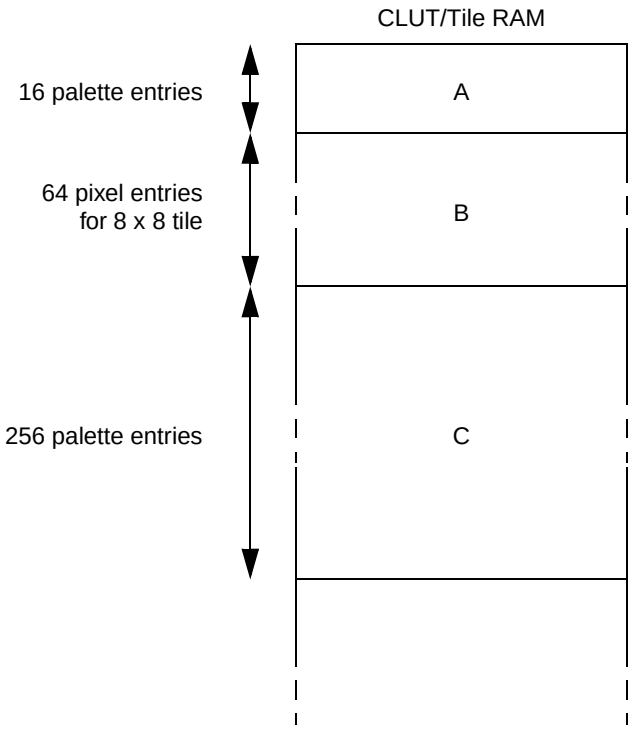


Figure 12-74. An example of use for the CLUT/Tile RAM

The CLUT/Tile RAM is mapped in the DCU 16K memory space from address 0x2000 to 0x3FFF. This gives 2048 entries, which provides up to eight full CLUTs for 8 bpp layers.

The CLUT/Tile RAM may be written at any time when the TFT LCD panel is not being driven with data. This means that the RAM can be modified when the DCU is not enabled and during the vertical blanking period.

12.4.8 Gamma correction

The gamma table allows the user to define an arbitrary transfer function at the output of each color component. The function (Equation 12-5) is applied to each pixel after all blending is complete and before the data is driven to the TFT LCD panel. Gamma correction is optional and can be used to adjust the color output values to match the gamut of a particular TFT LCD panel, or to perform data inversion or data length reduction on each component.

output_color_component = gamma_table[input_color_component] ***Eqn. 12-5***

The table is arranged as three separate memory blocks within the DCU memory map; one for each of the three color components. Each memory block has one entry for every possible 8-bit value and the entries are stored at 32-bit aligned addresses. This means that the upper 24 bits are not used while reading/writing the gamma memories. See Figure 12-75 for details of the memory arrangement.

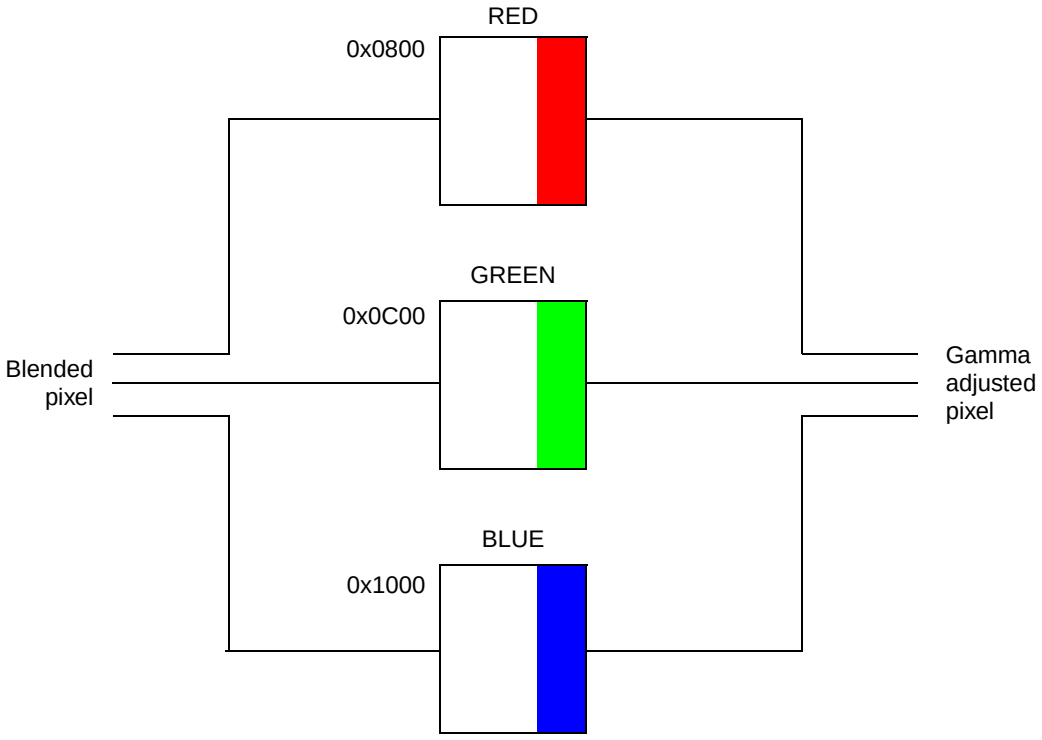


Figure 12-75. Gamma correction table organization

The gamma table can only be read or written when the DCU is not enabled or during the vertical blanking period.

12.5 Timing, error and interrupt management

The DCU can detect and raise status and error flags when the status of the system changes and when configuration or operational errors are detected.

12.5.1 Synchronizing to panel frame rate

Since the DCU fetches data directly from memory independently of the CPU, there is the possibility that changes to the DCU layer configuration or content can create incoherent content on the panel. To help avoid this situation there are five timing control flags that define when the DCU recognizes and locks changes to its configuration. These can be used to manage changes to control descriptors, CLUT or tile memory, or source graphics and so avoid coherency problems on the panel. All the timing flags are in the INT_STATUS register and can be used to generate interrupts from the DCU.

The DCU configuration is completely open during the vertical blanking period and control descriptors and some other registers may also be programmed at any time. The configuration present one HSYNC before the end of the vertical blanking period is the configuration used by the DCU for the panel refresh phase.

The VS_BLANK and LS_BF_VS flags give indication of the start of the vertical blanking period. The VS_BLANK flag is set at the beginning of the vertical blanking period. The LS_BF_VS flag is set a given number of horizontal lines before the start of the vertical blanking period; the given number of lines is defined by the LS_BF_VS bit field in the THRESHOLD register.

The PROG_END flag indicates that the DCU has locked the contents of its configuration registers for the new panel refresh period. No further changes are accepted to the DCU configuration after this flag is set (until the next vertical blanking period).

The DMA_TRANS_FINISH flag indicates that the DCU has completed fetching all data from memory in the current panel refresh cycle. This normally precedes the vertical blanking period and indicates that it is possible to change the contents of a memory that contains graphics used by the DCU.

The VSYNC flag indicates that the DCU has begun the next panel refresh period.

12.5.2 Managing the DCU FIFOs and DMA activity

The DCU fetches graphic data directly from internal and external memory using a dedicated DMA system and manages the output of data to the TFT LCD panel such that the panel always receives the pixel information when expected. Since the panel is sharing access to memory with the system DMA and CPU it cannot depend on the required data always being available at all times. It therefore it uses input and output FIFOs to temporarily store incoming and outgoing data until required and thus reduces the opportunity for the panel to be starved of pixel data.

The DCU manages the supply of graphic data to its format conversion and blending stages using input FIFOs that are 256 × 32 bits in size. The data that is driven to panel is managed using an output FIFO that is 128 pixels in size. See Figure 12-1 for a diagram of the input FIFO and output FIFO operation in the DCU

The input FIFOs are not accessible to the user but it is possible to set thresholds that control the DCU behavior when the FIFOs are becoming full or empty and observe when the lower and higher thresholds are reached. This can help detect and avert situations where the DCU is running out of data to send to the panel.

The FIFO thresholds are set in the THRESHOLD_INPUT_BUF_1/2 registers. The upper thresholds are set by the INP_BUF_Pm_HI bit fields (where m is the position of the pixel in the blend stack) and these

set the point at which the DCU pauses fetching data from memory. The maximum size of any DMA burst is fixed to 16 pixels and so is dependent on the graphic encoding. The lower thresholds are set by the INP_BUF_Pm_LO bit fields.

Each of the four input FIFOs has two flags that indicate whether the FIFO has reached its upper or lower threshold. The Pm_FIFO_HI_FLAG flags (where m is the position of the pixel in the blend stack) indicates that the input FIFO has reached the upper threshold. The Pm_FIFO_LO_FLAG indicates that the input FIFO has less data than its low threshold. Depending on when the low threshold is reached this may indicate a number of scenarios

- The expected graphical data is not available for the DCU to load
- The DCU is reaching the end of a frame and does not need to load any more data
- The blend stack does not need pixels of this priority

In the situation where the data is not available to the DCU then there may or may not be an impact to the data visible on the panel. In the situation where the output FIFO is full then it is possible for the DCU to accept a delay before it requires to use the incoming data.

The output FIFO is not accessible to the user but it is possible to set thresholds that control the DCU behavior when the FIFO is becoming full or empty and observe the lower threshold. This can help detect and avert situations where the DCU is running out of data to send to the panel.

The buffer thresholds are set in the THRESHOLD register. The upper threshold is set by the OUT_BUF_HIGH bit field and this indicates that sufficient data exists in the output buffer and processing should stop until the DCU uses some of the values in the FIFO. If this value is set too low then the possibility of the DCU running out of data to drive the panel is increased. The lower threshold is set by the OUT_BUF_LOW bit field.

When the output FIFO has emptied below its low threshold (OUT_BUF_LOW bit field) it sets the UNDRUN bit. In an under run situation there may or may not be an impact to the data visible on the panel. The impact depends on whether the DCU is reaching the end of a frame and how close to running out the threshold is set.

The best guide to indicate whether the DCU is able to supply the required pixel information to the panel is the output buffer. If the output is indicating that it is running out of data then the input FIFOs may help identify the areas of memory that are restricting the supply of data. Using these indicators can help to set the DCU thresholds and ensure that the data throughput on the MCU is balanced correctly for all master devices.

Finally, note that the number of DCU clock cycles to fetch and blend each pixel increases with the depth of the blend stack. However, the time taken to process the pixel data is fixed by the timing requirements of the panel. Therefore, for full performance across all color encodings the ratio between the DCU clock and the pixel clock must increase as the blend stack depth increases. For two-pixel blending, the minimum DCU clock must be twice the TFT pixel clock. For three-pixel blending, the minimum DCU clock must be three times the TFT pixel clock. For four-pixel blending, the minimum DCU clock must be four times the TFT pixel clock.

12.5.3 Error detection

The DCU asserts error flags when errors are detected in its configuration or when the user attempts to modify the configuration at an invalid point in the panel refresh period or when it is unable to access the required source data. The error flags may raise an interrupt if enabled to do so by the related mask bit in the corresponding mask register.

Error flags are stored in the PARR_ERR_STATUS and INT_STATUS registers.

Errors in the DCU configuration are collected in the PARR_ERR_STATUS register. The flags Ln_PARR_ERR (where n is the layer number) indicate an error in the configuration of the layer which can be either an invalid tile mode size or a layer with a horizontal dimension that is smaller than the minimum size defined by the layer encoding (see Section 12.4.5.3, Layer size and positioning). The DISP_ERR flag indicates that the VSYNC and HSYNC pulse widths are configured to the invalid value of 0. The HWC_ERR flag indicates that the hardware cursor is either larger than the available memory or is placed in an off-panel position. The SIG_ERR indicates that the signature calculation specifies an area that extends beyond the panel size.

Reads of CLUT/Tile RAM during the period when the TFT LCD panel is being updated do not return the CLUT/Tile RAM content.

Errors caused when the DCU is unable to access its required source data are collected in the INT_STATUS register. These errors are indicated by the UNDRUN flag and the Pm_FIFO_LO_FLAG flags (where m is the position in the blend stack)

12.5.4 Interrupt generation

The DCU generates interrupt through four lines that are controlled by the contents of six registers:

- INT_STATUS
- INT_MASK
- PDI_STATUS
- MASK_PDI_STATUS
- PARR_ERR STATUS
- MASK_PARR_ERR STATUS

There are four interrupt status lines defined.These lines are grouped as follows

- Timing based interrupts:
  - VSYNC
  - LS_BF_VS
  - VS_BLANK
  - PROG_END
  - DMA_TRANS_FINISH
- Functional interrupts:
  - UNDRUN
  - CRC_READY

  - CRC_OVERFLOW
  - P1_FIFO_HI_FLAG
  - P1_FIFO_LO_FLAG
  - P2_FIFO_LO_FLAG
  - P2_FIFO_HI_FLAG
  - P3_FIFO_HI_FLAG
  - P3_FIFO_LOW_FLAG
  - P4_FIFO_HI_FLAG
  - P4_FIFO_LOW_FLAG
  - IPM_ERROR
- Parameter error interrupts
  - Layer Error
  - Signature Calculator Error
  - Display Error
  - HWC_error
- PDI-related interrupts (pdi_int)
  - This includes PDI related interrupts. See Section 12.8.2.7, PDI-related interrupts, for a description.

When any interrupt occurs, the host can identify which type of interrupt has occurred by reading the interrupt status register/PDI status register/PARR_ERR status register.

12.6 Register protection

There is a customized register protection scheme on the DCU that is different to the protection scheme implemented elsewhere on the MCU. The scheme provides a mechanism to protect certain registers in the DCU from being written.

12.6.1 Operation of scheme

The register protection scheme provides a two-step protection scheme for the protected register.

Firstly, each register has an associated soft lock bit (SLB) that prevents further writes to the register when it is set. Each SLB has a corresponding write enable (WEN) bit that must be set in the same write operation as the SLB. The SLB can be set or cleared by writing a '1' or '0' to it while its WEN bit is set. The SLB bits are in the Soft Lock Registers L0 and L1, DISP_SIZE, HSYNC/VSYNC_PARA, POL, L0_TRANSP and L1_TRANSP registers.

Secondly, there is a hard lock bit (HLB) in the Global Protection Register which prevents all changes to soft lock bits. The HLB can only be cleared by a system reset.

If a write is made to a register whose SLB is set then a transfer error occurs that generates an IVOR2 exception on the CPU. Similarly if the HLB is set then any write to the SLB registers causes a transfer error.

12.6.2 List of protected registers

The register protection scheme applies to the following registers:

- All Layer 0 control descriptors CTRLDESCL0_1 to CTRLDESCL0_7
- All Layer 1 control descriptors CTRLDESCL1_1 to CTRLDESCL1_7
- Layer 0 foreground and background registers for transparency mode FG0_FCOLOR and FG0_BCOLOR
- Layer 1 foreground and background registers for transparency mode FG1_FCOLOR and FG1_BCOLOR
- all Control Descriptors and Transparency Registers for Layer1
- DISP_SIZE
- HSYNC_PARA
- VSYNC_PARA
- SYN_POL

12.7 Safety mode

Safety layers are used in a multi-layer DCU environment for the purpose of guaranteeing that the content is visible on the display regardless of the setting of remaining layers and the pixel manipulation algorithms of the DCU.Safety Feature is a requirement from qualification institutes to be able to reach a safety level of SIL2 or ASILB.The safety layers (Layer 0 and Layer 1) have the highest priority and use chroma key for complex area description.When a layer has safety mode enabled and layer format is 32 bpp or luminance mode, the layer is disabled as this is a wrong setting.Also alpha blending for the layer is disabled if layer has safety mode enabled.

A signature calculator module is implemented in the DCU that calculates the signature (value and position) for a predefined area of the frame.

DCU specifies a set of registers which define the window/area of the pixels for which CRC needs to be calculated. The application sets these registers. The signature calculator starts to calculate the signature after the first vertical (frame) sync after activation and when first pixel in the selected area is activeness tagged). It is also possible to calculate the CRC value for all pixels if the TAG_EN bit in the register is set to 0.

The pixels in the region of the layer on which signature needs to be calculated are tagged by the DCU. All the pixels for the safety layer enabled layer are tagged and then depending on the chroma keying operation, pixels with tagged values are passed along with the original data (24 bpp RGB + 1 bit safety pixel tag). The CRC value for these tagged pixels are then calculated by the Signature calculator.

Once CRC calculation is done on all the pixels in the window of interest, an interrupt CRC_Ready is generated for the processor to compare the CRC value with a precomputed value. Signature calculator would continue to calculate CRC for the next frame. If the interrupt is not processed within one frame time period, then CRC overflow interrupt is issued. The signature calculator calculates the content signature and position signature.

If the user has set NEG bit for DCU which indicates that the pixels fed to the display are inverted of output pixels, value CRC would be calculated on non inverted values. The position CRC however remains as it is.

In case a pixel has FG0 and FG1 as two highest priority layers with safety mode enabled in layer 1, normal arbitration takes place.

The polynomial used for CRC calculation is

(x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1)

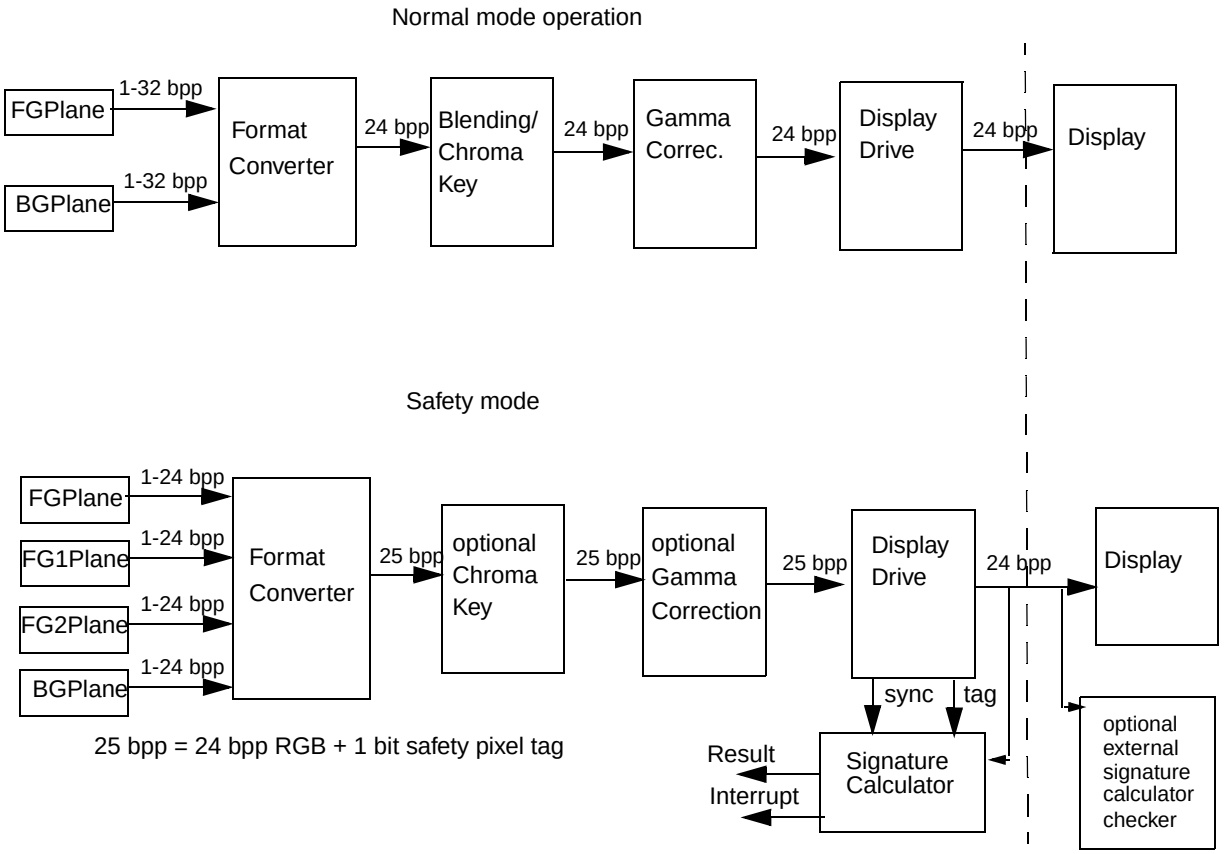


Figure 12-76. Safety mode block diagram

12.7.1 CRC area description

12.7.1.1 Relationship between various input signals

1. CRC is calculated when Safety mode is enabled.
2. CRC can be calculated over full screen as shown in Figure 12-77. Here tag bit (generated internal to DCU and flow along the data path) is set low. Data enable is coming for green color portion given in Figure 12-77. Data enable is not affected by chroma keying. TAG_EN register bit is low in this case.

This scenario can also be used in debug mode to calculate the data of the single layer. For such case CPU has to just enable the single concerned layer and disable all the mathematical function on the same.

3. CRC can be calculated for the part of the screen as shown in Figure 12-77. Here tag bit is set low.Data enable is coming for green color portion given in Figure 12-77. Data enable is not effected by chroma keying. TAG_EN register is low for this case. This can also be used in debug mode.

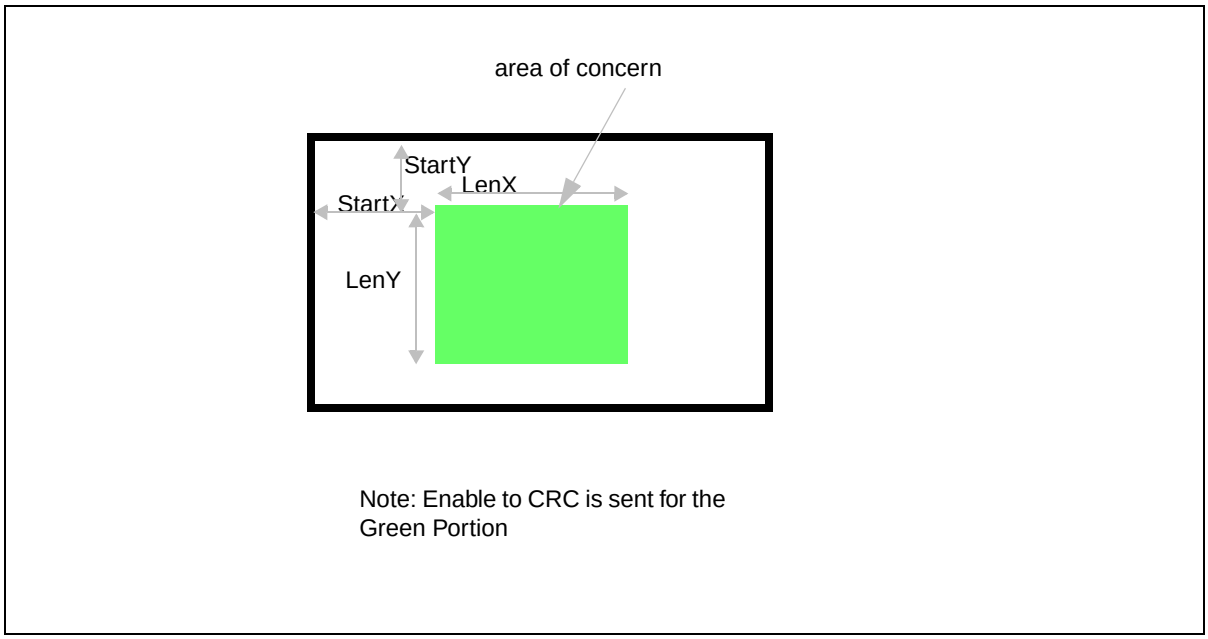


Figure 12-77. Safety mode enabled for part of the screen

4. CRC can be calculated only for the safety layers. Layer 0 and 1 can only act as a safety layer. Both of them have a separate control bits. In calculates the CRC over the intersection of both as shown (dark pink) in Figure 12-78. Here tag bit set high for all the bits corresponding to the layer 0 and 1 as shown by pink color (both dark and light) as shown in Figure 12-78. Also data is affected by chroma keying.Tag bit is cleared for the pixel removed by chroma keying.As shown in Figure 12-78 Data enable is controlled by the region shown in green color and tag is controlled by the pink region. CRC is calculated over region in dark pink.

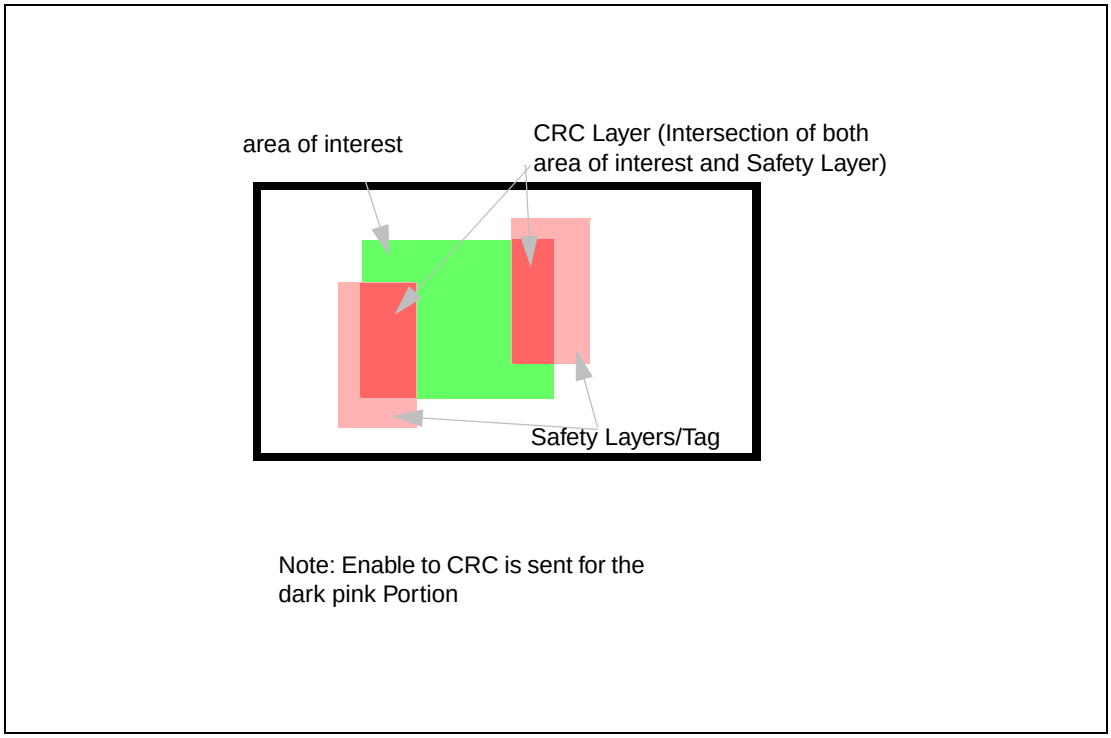


Figure 12-78. Safety mode with tag bit high

5. Layer 0 and 1 (i.e., safety layer) in safety mode does not supports blending or luminance offset.

12.7.2 Features

- SC support area modes mentioned in Table 12-64
- Supports calculation of CRC on the pixel value only
- CRC is calculated with the initial value as 32'h000000000
- CRC does not support:
  - Any modification of input bytes
  - Any modification of the output CRC value before reporting.
- When PDI is enabled with the Layer 0 and 1 (safety enabled) in a single sector, then PDI would act as BG layer and Layer 0 as a FG Layer.CRC would be calculated over a single L0 Layer.

Table 12-64. Supported Area

Area	Tag Value	Note
Full	0	StartX = 0 StartY = 0 LenX = Screen Size LenY = Screen Height

Table 12-64. Supported Area (continued)

Area	Tag Value	Note
Part	0	All the parameter have value other the one mentioned above as shown in Figure 12-77
Safety Layer (Layer 0 and 1 only)	1	Part of the safety Layer would depend on • Part lying within the area of concern defined by the StartX startY LenX LenY • Part deleted by the chroma keying functionality

12.7.3 Programming for Debug mode

1. Program the only debug layer as FG layer. No layer is programmed as BG layer.
2. Set startX, Starty, WIDTH, and LENGTH as full layer parameters.
3. Set Safety mode to 1.
4. Check the CRC when CRC calculation interrupt is raised for a particular screen.

12.7.4 Programming of Tag mode

1. Program the safety layers.
2. Set startX, Starty, WIDTH, and LENGTH as full layer parameters.
3. Set Tag bit to 1.
4. Set Program Safety mode to 1.
5. Check the CRC when CRC calculation interrupt is raised fora particular screen.

12.8 Parallel Data Interface (Camera Interface)

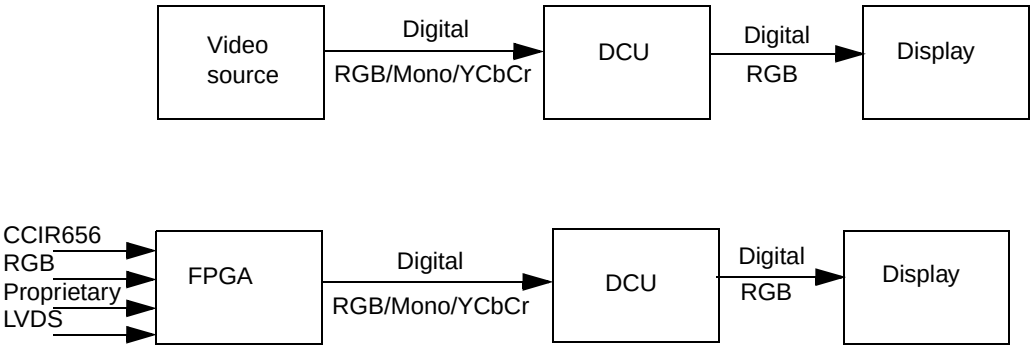


Figure 12-79. Camera interface block diagram

12.8.1 ITU-R BT.656 sync information extraction

According to ITU-R BT.656 recommendation, the incoming digital video will have a pdi_clk signal and 8 data bits. The data bits can contain both the video data and the timing reference signals (VSYNC and HSYNC).

The timing signals are encoded at the start and end of each line by timing reference codes known as Start of Active Video (SAV) and End of Active Video (EAV). The SAV and EAV codes are identified by their preamble of three bytes (0xFF, 0x00, 0x00). Due to this, neither 0x00 or 0xFF can be used during the Active Video Data. The preamble is followed by the XY status word which contains a field bit (F), a vertical blanking bit (V), a horizontal blanking bit (H), and four protection bits for single-bit error correction and detection.

The H bit is set to one to denote an EAV. That is the end of a line, and the beginning of the horizontal blanking period. The V bit is set to 1 to denote the beginning of the vertical blanking period. The F bit is used for interlaced video to denote if the forthcoming line is odd or even.

The remaining 4 bits make up the protection bits for single-bit error correction and detection. F and V fields are only allowed to change as part of EAV sequences i.e during transitions from H=0 to H=1.

An entire line of video comprises Active Video + Horizontal Blanking (from the start of the EAV code until the end of the SAV code) and Vertical Blanking (the space where V = 1).

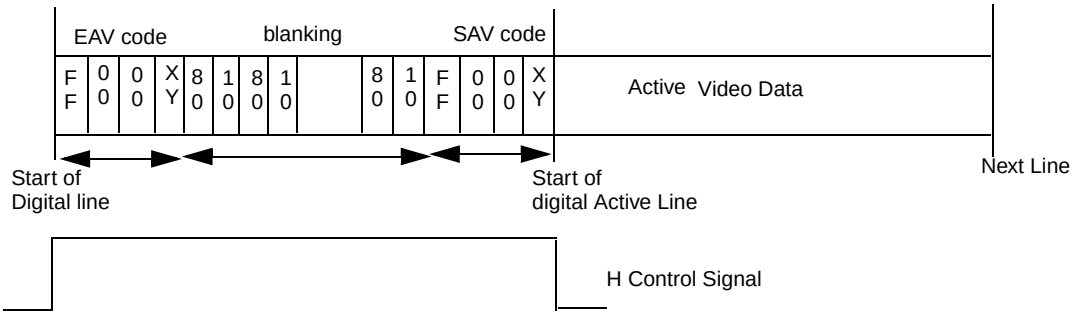


Figure 12-80. ITU-R BT.656 8-bit parallel data format for 525 video system

NOTE

Only 8-bit video is supported on this device.

Only non-interlaced video is supported on this device. The Field (F) value is ignored. For more information, please see *EB736, Limitations of the PDI Module on the MPC560xS*.

Table 12-65. Control byte sequence for 8-bit video

Data Bit	First word (FF)	Second word (00)	Third word (00)	Fourth word (XY
D9(MSB)	1	0	0	1
D8	1	0	0	F
D7	1	0	0	V

Table 12-65. Control byte sequence for 8-bit video (continued)

Data Bit	First word (FF)	Second word (00)	Third word (00)	Fourth word (XY
D6	1	0	0	H
D5	1	0	0	P3
D4	1	0	0	P2
D3	1	0	0	P1
D2	1	0	0	P0
D1	1	0	0	0
D0	1	0	0	0

The bit definitions for the status word XY are given in Table 12-68.

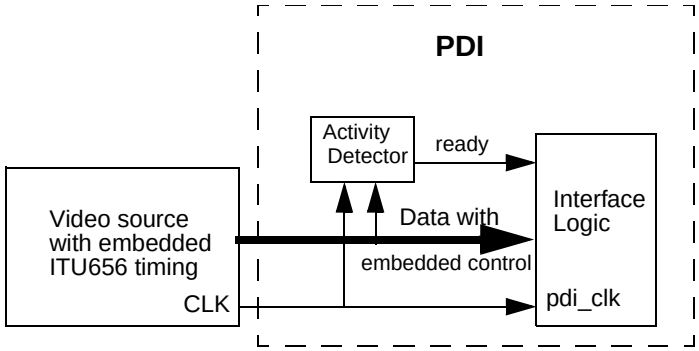


Figure 12-81. PDI Input data mode

Figure 12-81 represents the scenario in which data from an ITU-R BT.656 compliant video source is fed into the PDI interface. The incoming data includes codes that trigger the start and end of the active video and blanking fields. An activity detector checks for the transitions on the PDI bus. It samples the values on the PDI bus and once it has detected valid activity, sets a flag in the status register and can optionally trigger an interrupt. The PDI interface has a state machine which extracts the control information from the video data. The machine checks the video data for the Preamble Field (0xFF,0x00,0x00) and then depending on the status bits XY decides if it has received a valid control signal.

12.8.2 PDI interface description

12.8.2.1 Introduction

This block takes pixel information from the video source and passes to the DCU block to display the pixel information to the TFT/LCD screen. It can also be used in the Slave mode; that is, it takes only timing info from external chip/FPGA and display pixel info from memory to TFT screen at the timing provided.

12.8.2.2 PDI interaction with other modules

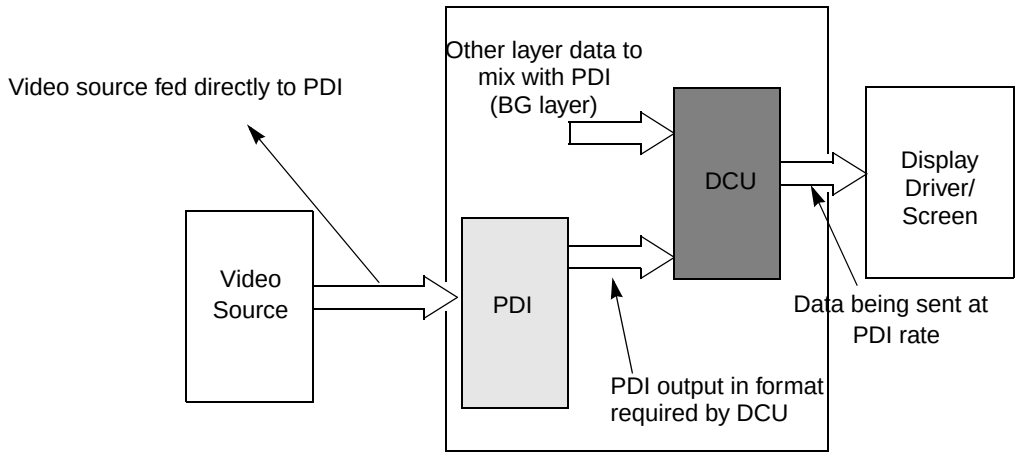


Figure 12-82. PDI interacting directly with the external sensor

In Figure 12-82, PDI directly accepts the data from the external video source. External device must support the interface mentioned in the document.

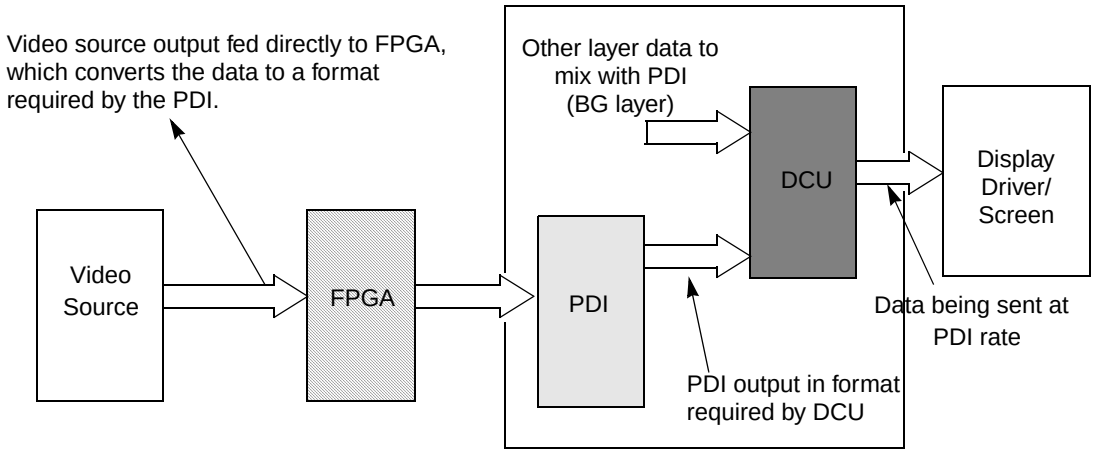


Figure 12-83. PDI interacting with FPGA in-between

As shown in Figure 12-83, PDI would not directly interface to the video source or any existing image processor chip. FPGA sits in between.

In normal mode, PDI clock frequency should be equal to pixel clock frequency desired for the TFT display driver. In narrow mode, PDI clock frequency is double the desired pixel clock frequency. The incoming clock does not have to bear any relation to the DCU clock. Prior to the lock condition the DCU will run on the internal DCU clock. After lock has been achieved, the DCU will switch to the clock from the PDI stream.

PDI clock would be used to send data and timing signal to TFT/LCD display driver.

In all cases, the resolution of the incoming stream and the Hsync and Vsync frequency must be the same as that of the TFT screen. All the horizontal parameter (Front Porch width, Back Porch width, Pulse width)

and vertical parameter (Front Porch width, Back Porch width, Pulse width) should be same as that of TFT screen.

If PDI is the background layer, no other layer can be a background layer for that particular frame. One and only background layer is possible i.e. PDI Layer when PDI is enabled.

12.8.2.3 Features

The PDI supports the following:

- RGB565, RGB666, 8-bit monochrome format,YCbCr422 mode
- Max input frequency of 32 MHz in 8/16/18 normal mode input
- Max input frequency of 64 MHz in 8-bit muxed (Narrow) mode.
- External Synchronization using Hsync, Vsync, and Pdi_clk
- External Synchronization using Hsync, Vsync, DataEn and pdi_clk
- Internal synchronization using pdi_datain and pdi_clk is supported for RGB565 and YCbCr422 muxed mode only.

Table 12-66. Supported RGB format and Sync format

RGB format	Data input bus
8-bit monochrome	8-bit
RGB565	16-bit
RGB666	18-bit
RGB565 muxed	8-bit
YCbCr	8-bit
Sync format	**Pin used**
Internal Sync (Valid only for RGB565 muxed mode)	Pclk
External Sync	Hsync, Vsync, Pclk
External sync (with data En)	Hsync, Vsync, Pclk, DE

12.8.2.4 Normal and Narrow mode

In normal mode, PDI support maximum input frequency of 32 MHz. In narrow mode, PDI supports maximum input frequency of 64 MHz.

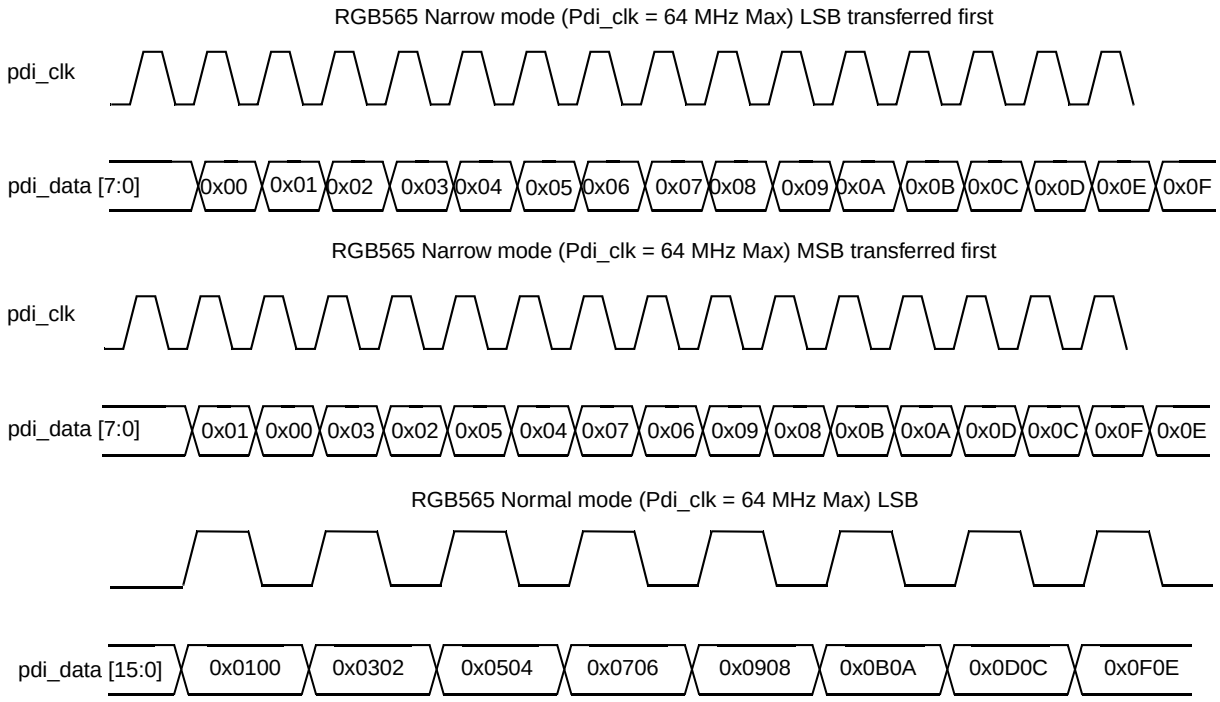


Figure 12-84. Data transfer in Normal and Narrow mode

The byte transferred first (MSB or LSB) depends on the configuration register as shown in Figure 12-84. This would not effect the sync preamble sequence in case internal sync mode.

On this device, the incoming RGB data is mapped onto the PDI pins as described in Table 12-67.

Table 12-67. Mapping of RGB data onto PDI pins

Mode	Mapping
Normal (full 18-bit PDI interface)	PDI[17:12] = R[5:0] PDI[11:6] = G[5:0] PDI[5:0] = B[5:0]
Normal (RGB565 16-bit PDI interface)	PDI[15:11] = R[4:0] PDI[[10:5] = G[5:0] PDI[4:0] = B[4:0]
Narrow (8-bit PDI interface)	RGB565: In first clock cycle, PDI[7:0] = { R[4:0], G[5:3] } In second clock cycle, PDI[7:0] = { G[2:0], B[4:0] } YCbCr: In first clock cycle, PDI[7:0] = { Cb[7:0] } In second clock cycle, PDI[7:0] = { Y0[7:0] } In third clock cycle, PDI[7:0] = { Cr[7:0] } In forth clock cycle, PDI[7:0] = { Y1[7:0] }

12.8.2.5 Modes of operation based on sync extraction

12.8.2.5.1 PDI Input Data (External Sync mode)

In External sync mode the timing signals (HSYNC, VSYNC and, optionally, Date Enable) are provided the timing pins by the external video source.

External sync mode can be used in both normal mode and 8-bit narrow mode. In the instance that external sync and narrow mode is selected, the external signals are used, and any timing information (EAV/SAV) embedded in the data stream is ignored.

As in Figure 12-86, PDI data enable (pdi_de) should be low during Vsync and Hsync pulse, Vsync front porch (FP_v) and back porch (BP_v), Hsync front porch (FP_h) and back porch (BP_h). This is valid for Data Enable mode when the PDI_DE_EN bit is set in the DCU_Mode register (i.e. mode with hsync, vsync, data enable and pdi_clk as pin signals).

Pulse width, Front and back porch values should be picked from those programmed in DCU registers. In order to achieve lock, it must have same value as that of TFT screen. Front porch and back porch value can be zero. Pulse width and TFT screen size parameters cannot be zero. In case they are programmed as zero, it might lead to malfunctioning of the validation state machine.

As in Figure 12-85 Hsync must be coming during the Vsync and V blanking period. Gap between 2 Hsync should be same during Vsync and V Blanking as during active line period. As in Figure 12-85 Positive Edge of the Hsync and Vsync should be aligned. As in Figure 12-85 the positive edge of the Hsync and start of the vertical front/back porch should be aligned. Polarity of hsync and vsync are selectable.

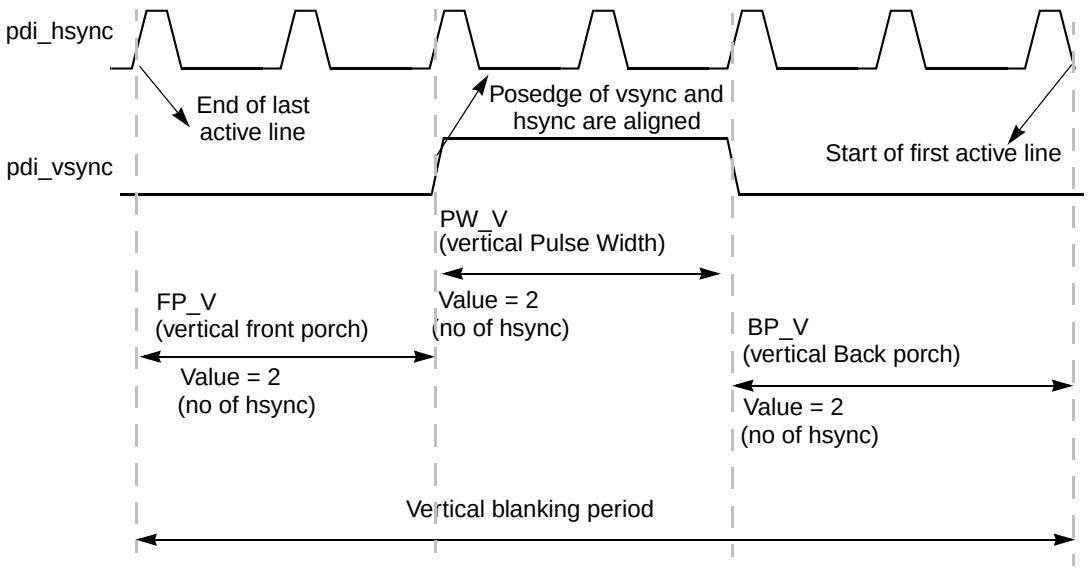


Figure 12-85. Relation between Hsync and Vsync in external synchronization

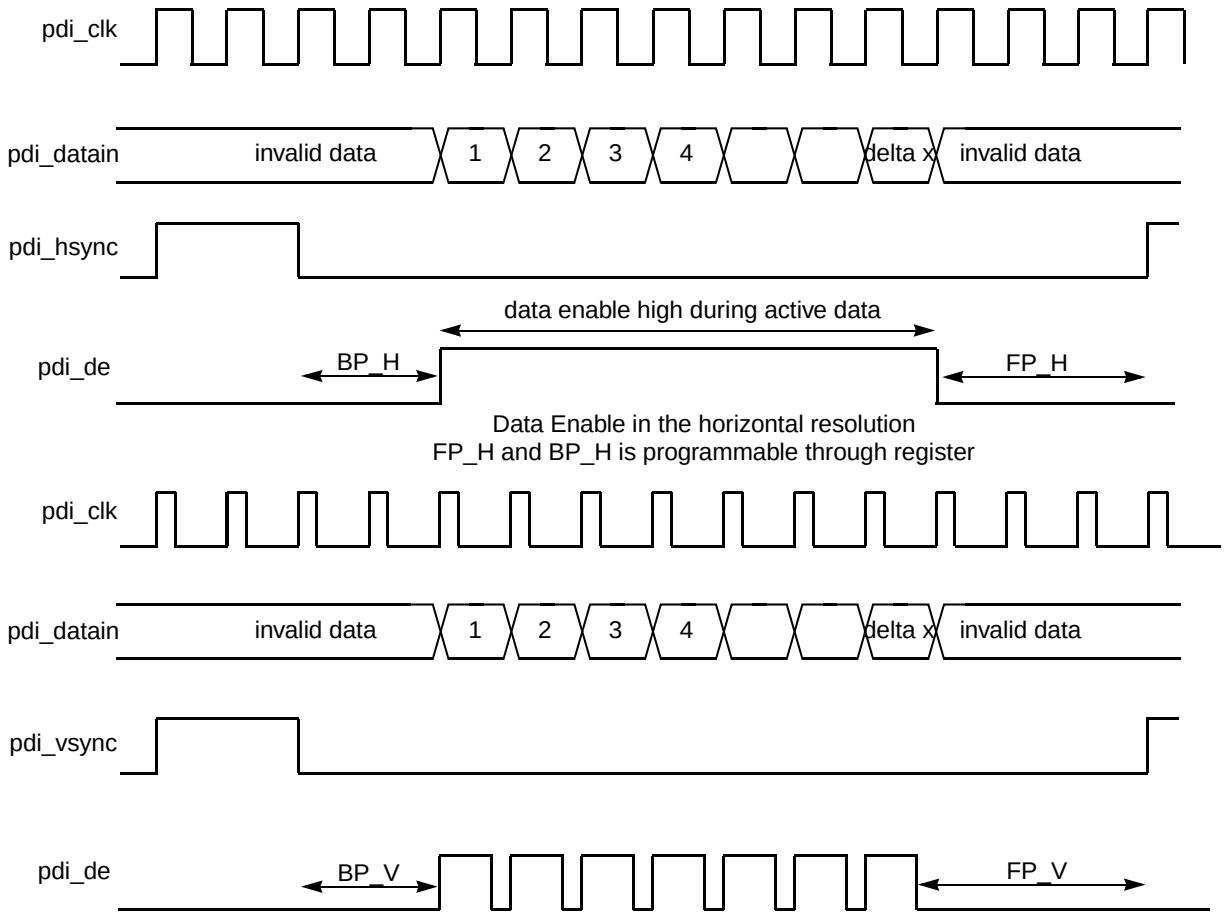


Figure 12-86. Occurrence of Hsync and Vsync and DataEn for the Entire Frame

12.8.2.5.2 PDI Input data (Internal Sync Extraction mode)

In internal sync mode the timing parameters (horizontal and vertical blanking) are encoded into the data stream.

Internal sync mode can only be used in 8-bit narrow mode.

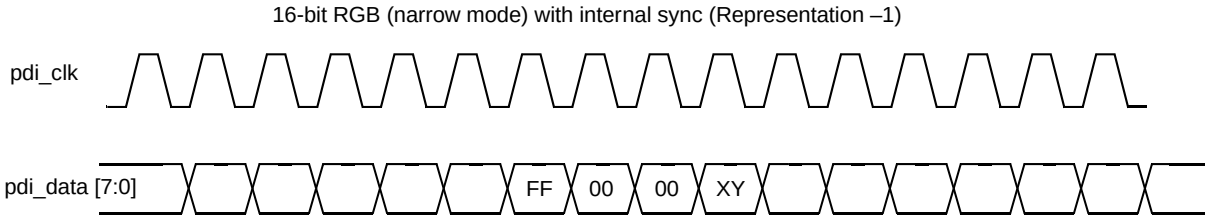
In Figure 12-87, XY is used to decode the vertical and horizontal blanking period.

Table 12-68. XYh Value

Bit	Value	Description
7	1'b1	Always 1'b1. This is checked while decoding sync preamble
6	F	Not considered in the state machine logic
5	V	1'b1 during vertical blanking 1'b0 elsewhere
4	H	1'b0 for start of active video 1'b1 for end of active video

Table 12-68. XYh Value (continued)

Bit	Value	Description
3 2 1 0	P3 P2 P1 P0	Protection bits (used to detect ECC errors). It would not be used for bit correction.



Note: Preamble sequence is independent of data sequence

Figure 12-87. Location of sync preamble in Narrow mode

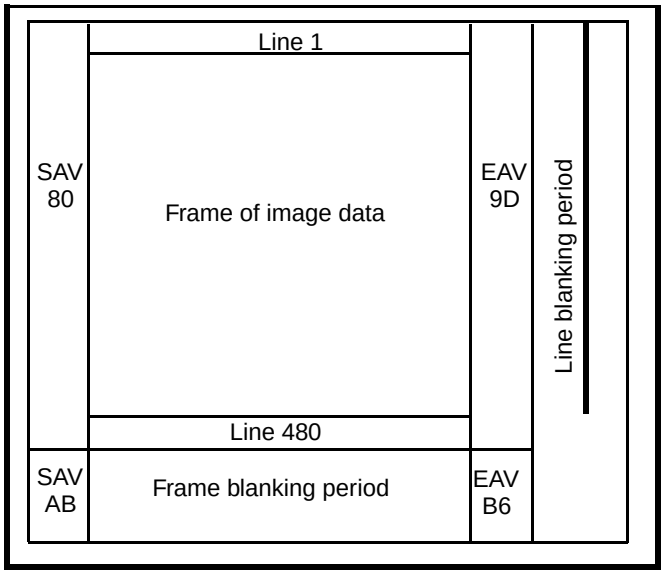
Sync Preamble would come continuously for 4 clock cycles as shown in Figure 12-87. It would not depend upon which byte is coming first in data (MSB or LSB). Sync extraction is done using pdi_datain [7:0]. Sync Extraction identifies the horizontal and vertical blanking period using H and V field of the XYh data as mentioned in Table 12-68.

ITU 656 Sync preamble pattern (FFh 00h 00h) has to be masked out in the RGB and YCbCr data. The data stream must not include FFh 00h 00h as the valid pixel data to avoid malfunction by the validation state machine.

Horizontal blanking period must be coming during the Vertical blanking period. Gap between 2 Horizontal blanking should be same during Vertical Blanking period as during line active. All Vertical and horizontal parameter values are validated against the DCU registers programmed by the user. Polarity of hsync and vsync are selectable. Horizontal blanking and vertical blanking must be aligned as shown in Figure 12-88. During blanking period it would check for the 80h 10h 80h 10h sequence. This sequence would be present both during horizontal (line) blanking and vertical (frame) blanking period.

Framing Bit (F field in XYh) would be ignored during extraction. Extraction is valid for RGB565 and YCbCr422 muxed mode. ECC error is only detected not corrected. It would be calculated using protection bits in Table 12-68

Same as External sync mode, value of front and back porches can be zero but pulse width and TFT screen parameter cannot be zero.



Note: The SAV and EAV bytes are included as part of the blanking period.

Figure 12-88. Relationship Between Hblank and Vblank in Internal Sync

12.8.2.5.3 PDI YCbCr mode

In this mode, the PDI extracts the ITU656 sync (FF-00-00), and sends the video to the processing functions.The first processing function converts the 422 stream to a 444 stream, by providing interpolation on the chroma components of the stream depending on PDI_INTERPOL_EN bit. The second processing function converts the stream to RGB888/RGB565.

The RGB pixel value is computed using following equations:

$$\text{Red} = \frac{(Y-16)\cdot yred}{512} + \frac{(Cr-128)Crred}{512} + \frac{(Cb-128)Cbred}{512} \qquad \textit{Eqn. 12-6}$$

$$\text{Green} = \frac{(Y-16)\cdot Ygreen}{512} + \frac{(Cr-128)Crgreen}{512} + \frac{(Cb-128)Cbgreen}{512} \qquad \textit{Eqn. 12-7}$$

$$\text{Blue} = \frac{(Y-16)\cdot yblue}{512} + \frac{(Cr-128)Crblue}{512} + \frac{(Cb-128)Cbblue}{512} \qquad \textit{Eqn. 12-8}$$

Note that the first multiplication (i.e (y-16)*ycoeff) is unsigned. The other two others signed.

The register values after reset are as follows:

Yred = 10’h254 (596/512 = 1.16)

Crred = 11’h331 (817/512 = 1.6)

Cbred = 12’h000

Ygreen = 10’h254(596/512 = 1.16)

Crgreen = 11’h660(-416/512 = -0.812)

Cbgreen = 12’hf38(-200/512 = -0.39)

Yblue = 10’h254(596/512 = 1.16)

Crblue = 11’h000

Cbblue = 12’h409(1033/512 = 2.017)

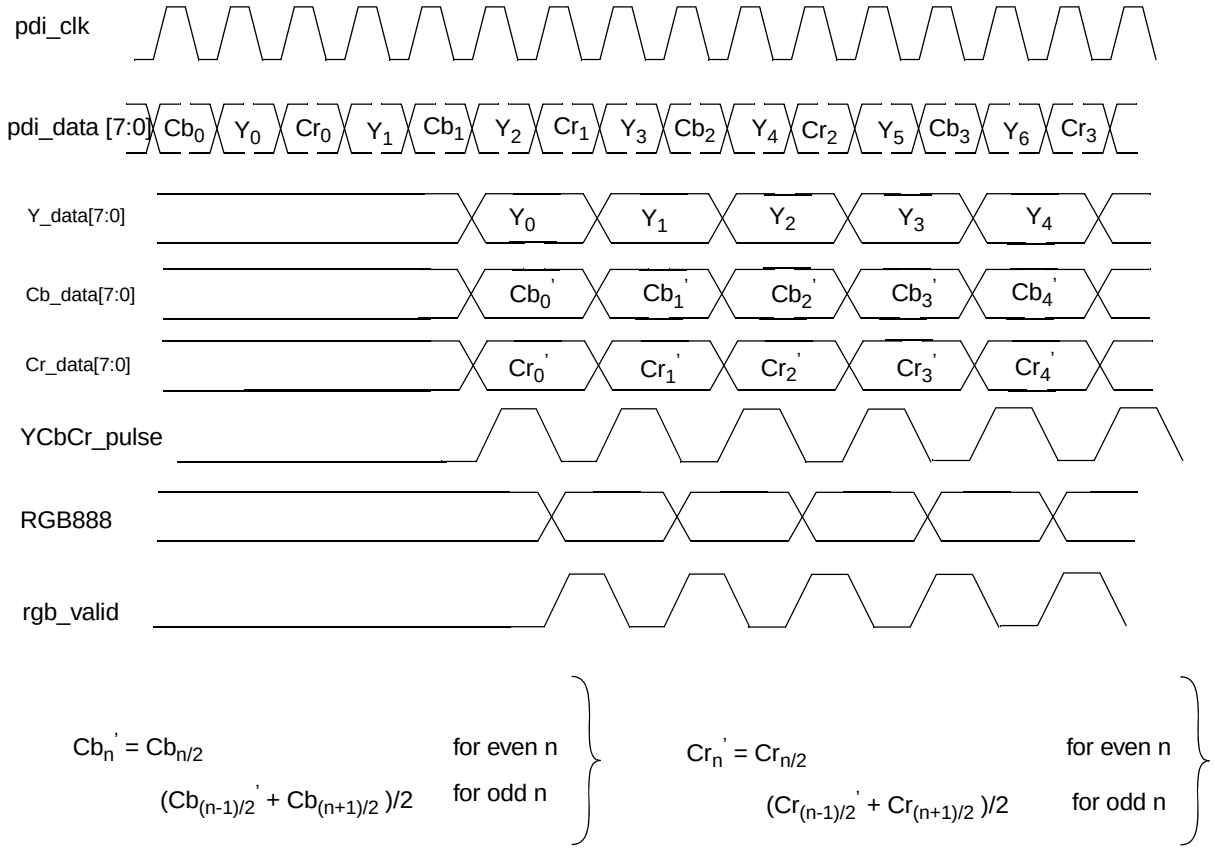


Figure 12-89. YCbCr timing diagram

12.8.2.6 Mode of operation depending on PDI_datain

PDI supports following modes (other than the Slave mode):

- 8-bit monochrome (8-bit input data, each pixel info is coming in 1 clocks)
- 16-bit—RGB565 (16-bit input data, each pixel info is coming in 1 clocks)
- 18-bit—RGB666 (18-bit input data, each pixel info is coming in 1 clocks)
- 16-bit—RGB565 (8-bit input data, each pixel info is coming in 2 clocks)
- 16-bit—YCbCr422 (8-bit input data, info for 2 co-sited pixels coming in 4 clocks)

Data info extraction is given in Table 12-69.

Table 12-69. Data extraction in all possible modes

PDI mode	Narrow mode	Pins	RGB data	Notes
8-bit monochrome mode	1'b0	8-bit	pdi_datain	—
RGB565	1'b0	16-bit	Pdi_datain	—
RGB666	1'b0	18-bit	pdi_datain	—
RGB565 muxed	1'b1	8-bit	pdi_datain	Data from two clocks are combined.
YCbCr422	1	8-bit	pdi_datain	Data from four clocks are combined for two pixels.

The 8-bit monochrome image is equivalent to 8-bit grayscale images. For converting 8-bit monochrome data to RGB data, each of the R/G/B components will have a value equal to the 8-bit monochrome value.

RGB extraction starts when PDI is enabled (from the next falling edge of validated vertical blanking period).

12.8.2.7 PDI-related interrupts

PDI can be configured to trigger an interrupt when synchronization is achieved i.e. it receives the pre-specified numbers of frames without error. PDI would also give an interrupt when synchronization is lost i.e. it receives any error in frame there after. This interrupt is raised when hsync/vsync is lost.

Correctable ECC error interrupts for errors during extraction of preambles. No error correction is done. Non-correctable ECC error interrupt for the preamble errors that cannot be corrected.

Blanking sequence error is received in case 80h 10h is not found in vertical and line blanking period during internal synchronization.

Activity detection interrupt for CLK detection.

Hsync activity detection interrupts. It would be generated from the state machine. Vsync activity detection interrupts. It would also be generated from the state machine.

Activity detection interrupt for data enable. Activity lost interrupt for CLK — for CLK lost interrupt, it is assumed that ipg_clk would be active for entire duration, if pdi_clk_frequency is less than or equal to ipg clock frequency/32, then the interrupt is triggered. (i.e. if ipg_clk freq. max = 64 MHz then the PDI_CLK_LOST flag will be set if pdi clk freq. min < 2 MHz).

All interrupts are RW1C (write one to clear). All interrupts are maskable.

PDI must reset to show the latest status of the clock activity detect interrupt.

12.9 DCU initialization

The following steps describe a typical approach to initializing the DCU for use in an application.

1. After reset configure the DCU peripheral to be active using the mode entry module and configure the DCU clock source in the MC_CGM.
2. Configure the output ports in the SIUL as required.

3. Configure the timing registers to match the TFT LCD panel in use (Section 12.4.2, TFT LCD panel configuration).
4. Set the background color as required.
5. Load the initial tile or palette colors into the CLUT/Tile memory.
6. Configure the control descriptors for the layers and cursor that are to be used initially.
7. Enable the DCU in the appropriate mode (DCU_MODE and RASTER_EN bit fields).

12.10 Glossary

Table 12-70. Glossary

ARGB	A data format where the pixel values are stored using four components: Alpha, Red, Green and Blue. DCU supports different variations of this format where different numbers of bits can be used to represent each of the components
Component	Part of a pixel that contains a single color (red, green or blue)
CLUT	Color Look-up table. The table that contains the palette used by an indexed-color graphic
Direct color	The full 18-bit value actually written to a pixel to create a color
Frame	The collection of all pixels on a panel
Gamut	The set of colors that a panel can display. In most cases a panel cannot display the full gamut of colors visible to the human eye.
Indexed color	An index into a table containing direct-colors. Usually smaller in size than the direct color; the DCU provides 1, 2, 4, and 8 bits per pixel options
Palette	The list of colors used by a graphic when an indexed colors format is used. The palette is stored in a color look up table and can be from one color up to the maximum of the size of the CLUT.
Panel	A TFT LCD containing an array of colored pixels.
Pixel	The basic graphical element on a TFT LCD panel. Can display a range of colors depending on the value of the red, green and blue values written to it. Normally arranged in a rectangular array.
RAM FIFO	A 16-entry buffer that allows writes to the DCU RAMs during the TFT LCD panel refresh period
RGB	A data format where the pixel values are stored using three components: Red, Green and Blue. DCU supports different variations of this format where different numbers of bits can be used to represent each of the components
Vertical blanking period	A time during the TFT LCD panel refresh cycle when no data is being written to the panel

Chapter 13
DMA Channel Mux (DMACHMUX)

13.1 Introduction

13.1.1 Overview

The DMACHMUX controls the routing of multiple DMA peripheral sources (called slots) to 16 DMA channels. This is illustrated in Figure 13-1.

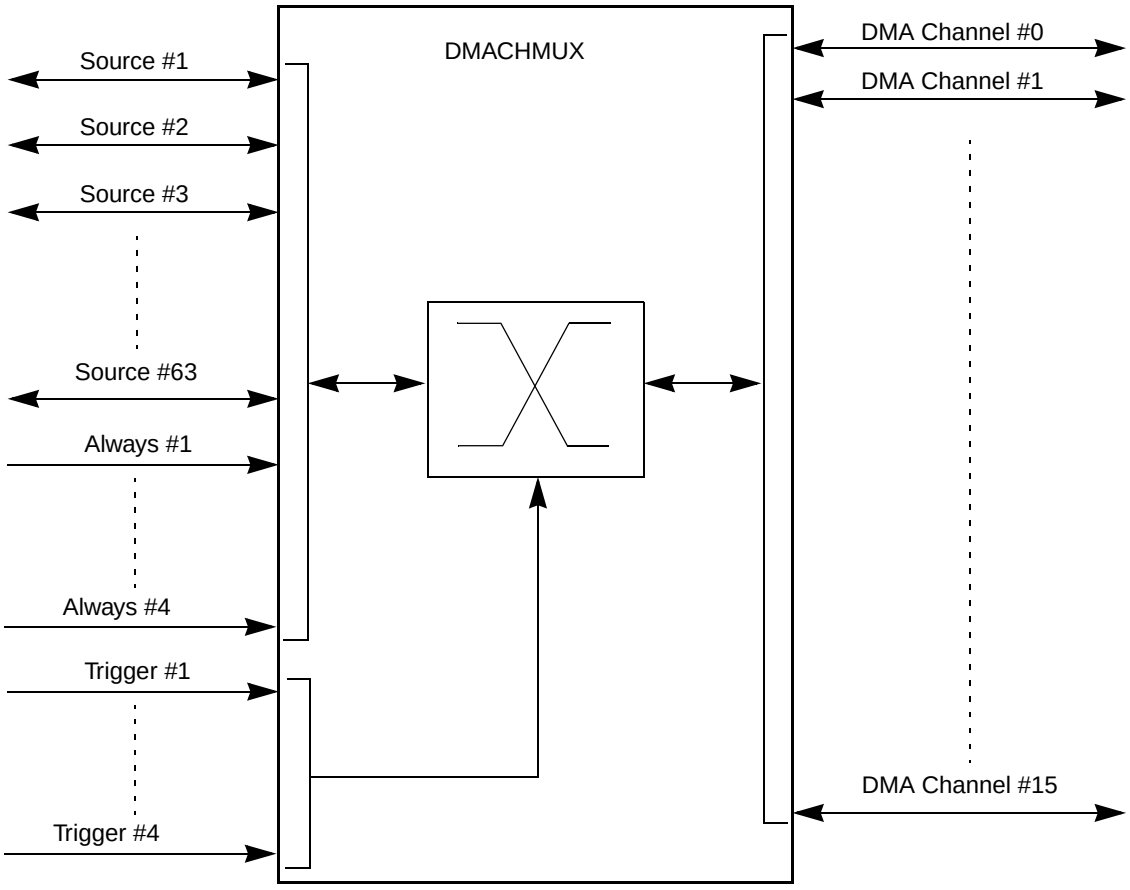


Figure 13-1. DMACHMUX block diagram

13.1.2 Features

The DMACHMUX provides these features:

- 48 peripheral slots + four always-on slots can be routed to 16 channels
- 16 independently selectable DMA channel routers
  - First four channels additionally provide a trigger functionality

- Each channel router can be assigned to one of 48 possible peripheral DMA slots or to one of the four always-on slots

13.1.3 Modes of operation

The following operation modes are available:

- Disabled mode

  In this mode, the DMA channel is disabled. Since disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel mux. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place (for example, changing the period of a DMA trigger).
- Normal mode

  In this mode, a DMA source (such as DSPI transmit or DSPI receive for example) is routed directly to the specified DMA channel. The operation of the DMA Mux in this mode is completely transparent to the system.
- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer (such as when a transmit buffer becomes empty or a receive buffer becomes full) periodically. Configuration of the period is done in the registers of the Periodic Interrupt Timer (PIT). This mode is only available for channels 0–4.

13.2 External signal description

13.2.1 Overview

The DMA channel mux has no external pins.

13.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the DMA channel mux.

Table 13-1 shows the memory map for the DMA channel mux. Note that all addresses are offsets; the absolute address may be computed by adding the specified offset to the base address of the DMA channel mux.

Table 13-1. Module memory map

Address	Use	Access	Location
Base + 0x00	Channel #0 Configuration (CHCONFIG0)	R/W	on page 451
Base + 0x01	Channel #1 Configuration (CHCONFIG1)	R/W	on page 451
..	..	..	..
Base + 0x*#n – 1*	Channel *#n* Configuration (CHCONFIG*#n – 1*)[1]	R/W	on page 451

[1] In the table *n* refers to the *number of channels – 1*

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCONFIG0 through CHCONFIG3 are accessible by a 32-bit read/write to address Base + 0x00, but performing a 32-bit access to address Base + 0x01 is illegal.

13.3.1 Register descriptions

The following memory-mapped registers are available in the DMA channel mux.

13.3.1.1 Channel configuration registers

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

Address: Base + *#n* Access: User read/write

	0	1	2	3	4	5	6	7
R W	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

Figure 13-2. Channel Configuration Registers (CHCONFIG#n)

Table 13-2. CHCONFIGxx field descriptions

Field	Description
ENBL	DMA Channel Enable. ENBL enables the DMA Channel 0 DMA channel is disabled. This mode is primarily used during configuration of the DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or re-configure a DMA channel. 1 DMA channel is enabled.
TRIG	DMA Channel Trigger Enable (for triggered channels only). TRIG enables the periodic trigger capability for the DMA Channel. 0 Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel. 1 Triggering is enabled.
SOURCE	DMA Channel Source (slot). SOURCE specifies which DMA source, if any, is routed to a particular DMA channel. Please check your SoC guide for further details about the peripherals and their slot numbers.

Table 13-3. Channel and trigger enabling

ENBL	TRIG	Function	Mode
0	X	DMA channel is disabled	Disabled mode
1	0	DMA channel is enabled with no triggering (transparent)	Normal mode
1	1	DMA channel is enabled with triggering	Periodic Trigger mode

NOTE

Setting multiple CHCONFIG registers with the same Source value will result in unpredictable behavior.

NOTE

Before changing the trigger or source settings a DMA channel must be disabled via the CHCONFIG[*#n*].ENBL bit.

Table 13-4. DMACHMUX request assignments

DMA requesting module	DMACHMUX source number (ipd_ref_peripher[$N_{periphs}$:1])
Channel Disable[1]	0
DSPI_0 TX	1
DSPI_0 RX	2
DSPI_1 TX	3
DSPI_1 RX	4
QuadSPI_0 TFFF	5
QuadSPI_0 RFDF / RBDF	6
I2C_0_TX	7
I2C_0_RX	8
I2C_1_TX	9
I2C_1_RX	10
I2C_2_TX	11
I2C_2_RX	12
I2C_3_TX	13
I2C_3_RX	14
eMIOS200_0_FLAG_F0	15
eMIOS200_0_FLAG_F1	16
eMIOS200_0_FLAG_F2	17
eMIOS200_0_FLAG_F3	18
eMIOS200_0_FLAG_F4	19
eMIOS200_0_FLAG_F5	20
eMIOS200_0_FLAG_F6	21
eMIOS200_0_FLAG_F7	22
eMIOS200_0_FLAG_F8	23
eMIOS200_0_FLAG_F9	24
eMIOS200_0_FLAG_F10	25

Table 13-4. DMACHMUX request assignments (continued)

DMA requesting module	DMACHMUX source number (ipd_ref_peripher[$N_{periphs}$:1])
eMIOS200_0_FLAG_F11	26
eMIOS200_0_FLAG_F12	27
eMIOS200_0_FLAG_F13	28
eMIOS200_0_FLAG_F14	29
eMIOS200_0_FLAG_F15	30
eMIOS200_1_FLAG_F0	31
eMIOS200_1_FLAG_F1	32
eMIOS200_1_FLAG_F2	33
eMIOS200_1_FLAG_F3	34
eMIOS200_1_FLAG_F4	35
eMIOS200_1_FLAG_F5	36
eMIOS200_1_FLAG_F6	37
eMIOS200_1_FLAG_F7	38
Reserved	39
Reserved	40
Reserved	41
Reserved	42
Reserved	43
Reserved	44
Reserved	45
Reserved	46
SIU_EISR_EIF1	47
SIU_EISR_EIF2	48
SIU_EISR_EIF3	49
SIU_EISR_EIF4	50
ADC	51
Reserved	52
Reserved	53
Reserved	54
Reserved	55
ALWAYS requestors	56
ALWAYS requestors	57

Table 13-4. DMACHMUX request assignments (continued)

DMA requesting module	DMACHMUX source number (ipd_ref_peripher[$N_{periphs}$:1])
ALWAYS requestors	58
ALWAYS requestors	59
ALWAYS requestors	60
ALWAYS requestors	61
ALWAYS requestors	62
ALWAYS requestors	63

[1] Configuring a DMA channel to select source 0 or any reserved sources will disable that DMA channel.

13.4 Functional description

This section provides a functional description of the DMA channel mux. The primary purpose of the DMA channel mux is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMA Mux is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Section 13.5.2, Enabling and configuring sources, is followed, the configuration of the DMA channel mux may be changed during the normal operation of the system.

Functionally, the DMA channel mux channels may be divided into two classes: channels that implement the normal routing functionality plus periodic triggering capability, and channels that implement only the normal routing functionality.

13.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first four channels of the DMA Mux provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. Please refer to Chapter 27, Periodic Interrupt Timer (PIT), for more information on this topic.

Table 13-5 shows the mapping of PIT channels to DMA channels for triggering.

Table 13-5. PIT-DMA channel mapping

PIT channel number	DMACHMUX channel number for triggering
0	0
1	1
2	2
3	3

NOTE

Because of the dynamic nature of the system (for example, DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

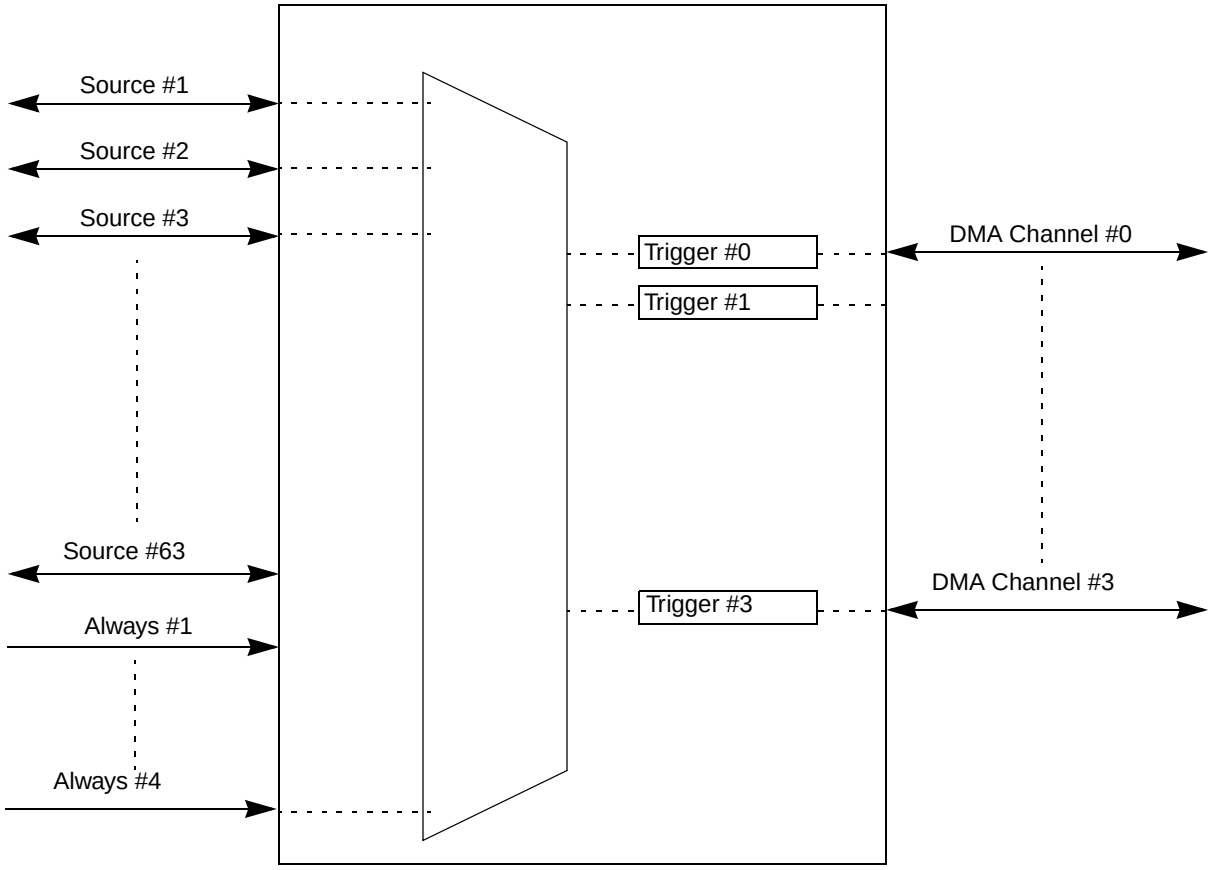


Figure 13-3. DMA mux triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event occurs. This is illustrated in Figure 13-4.

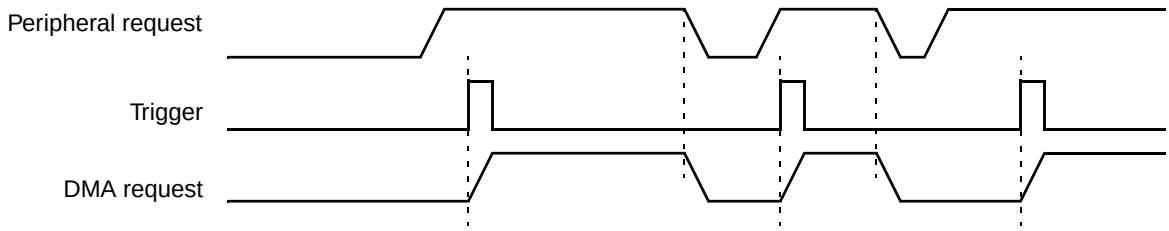


Figure 13-4. DMA mux channel triggering: normal operation

Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that trigger will be ignored. This situation is illustrated in Figure 13-5.

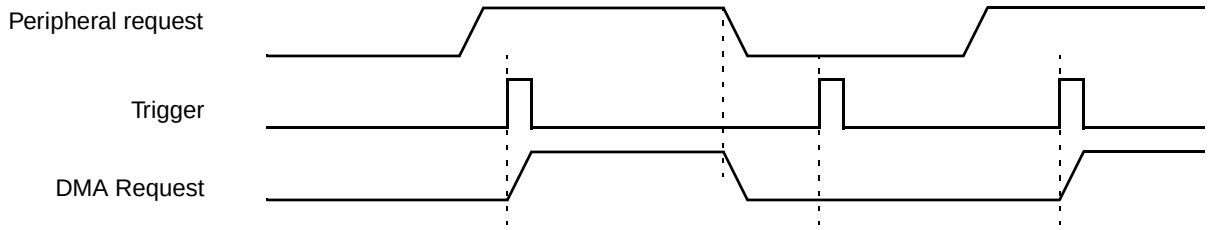


Figure 13-5. DMA mux channel triggering: ignored trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus.
  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once set up, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO ports to drive or sample waveforms.
  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger can be found in Chapter 27, Periodic Interrupt Timer (PIT).

13.4.2 DMA channels with no triggering capability

The other channels of the DMA mux provide the normal routing functionality as described in Section 13.1.3, Modes of operation.

13.4.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are four additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the always enabled sources provide no such throttling of the data transfers. These sources are most useful in the following cases:

- Doing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either un-throttled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Doing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Doing DMA transfers from memory to the external bus (or vice versa)—Similar to memory-to-memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel mux. The options for doing this are:

- Transfer all data in a single minor loop.
  By configuring the DMA to transfer all of the data in a single minor loop (major loop counter = 1), no re-activation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will incur on the system. For this option, the DMA channel should be disabled in the DMA channel mux.
- Use explicit software re-activation.
  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to re-activate the channel (by writing to the DMA registers) *after every minor loop*. For this option, the DMA channel should be disabled in the DMA channel mux.
- Use an always-enabled DMA source.
  In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel mux does the channel re-activation. For this option, the DMA channel should be enabled and pointing to an always-enabled source. Note that the re-activation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

13.5 Initialization/application information

13.5.1 Reset

The reset state of each individual bit is shown within the register description section (See Section 13.3.1, Register descriptions). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

13.5.2 Enabling and configuring sources

13.5.2.1 Enabling a source with periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first four DMA channels have periodic triggering capability.

2. Clear the ENBL and TRIG bits of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set.

Example 13-1. Configure source #5 transmit for use with DMA Channel 2, with periodic triggering capability

1. Write 0x00 to CHCONFIG2 (base address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCONFIG2 (base address + 0x02).

The following code example illustrates steps #1 and #4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
        :
        :
*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0xC5;
```

13.5.2.2 Enabling a source without periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first four DMA channels have periodic triggering capability.
2. Clear the ENBL and TRIG bits of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.

4. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL bit is set and the TRIG bit is cleared.

Example 13-2. Configure source #5 transmit for use with DMA Channel 2, with no periodic triggering capability.

1. Write 0x00 to CHCONFIG2 (base address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Write 0x85 to CHCONFIG2 (base address + 0x02).

The following code example illustrates steps #1 and #3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
         :
         :
*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0x85;
```

13.5.2.3 Disabling a source

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCONFIG registers. Additionally, some module-specific configuration may be necessary. Please refer to the appropriate section for more details.

13.5.2.4 Switching the source of a DMA channel

1. Disable the DMA channel in the DMA and re-configure the channel for the new source.
2. Clear the ENBL and TRIG bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set.

Example 13-3. Switch DMA channel 8 from source #5 transmit to source #7 transmit

1. In the DMA configuration registers, disable DMA channel 8 and re-configure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCONFIG8 (Base Address + 0x08).
3. Write 0x87 to CHCONFIG8 (Base Address + 0x08). (In this example, setting the TRIG bit would have no effect, due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps #2 and #4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
         :
         :
*CHCONFIG8 = 0x00;
*CHCONFIG8 = 0x87;
```

Chapter 14
e200z0h Core

14.1 Overview

The e200 processor family is a set of CPU cores that implement low-cost versions of the Power Architecture® Book E architecture. e200 processors are designed for deeply embedded control applications which require low cost solutions rather than maximum performance.

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

The e200z0h core is a single-issue, 32-bit PowerPC Book E VLE-only design with 32-bit general purpose registers (GPRs). All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs).

Instead of the base PowerPC Book E instruction set support, the e200z0h core only implements the VLE (variable-length encoding) APU, providing improved code density. The VLE APU is further documented in the *PowerPC™ VLE APU Definition*, a separate document.

14.2 Features

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

- 32-bit Power Architecture Book E programmer's model
- Single issue, 32-bit CPU
- Implements the VLE APU for reduced code footprint
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - — Dedicated branch address calculation adder
  - — Branch acceleration using Branch Target Buffer (BTB)
- Supports independent instruction and data accesses to different memory subsystems, such as SRAM and flash memory via independent Instruction and Data bus interface units (BIUs).
- Load/store unit
  - — 1 cycle load latency
  - — Fully pipelined
  - — Big- and Little-endian support
  - — Misaligned access support
  - — Zero load-to-use pipeline bubbles for aligned transfers
- Power management

  - Low-power design
  - Dedicated power saving state: wait
  - Dynamic power management of execution units
- Testability
  - Synthesizeable, full MuxD scan design
  - ABIST/MBIST for optional memory arrays

14.2.1 Microarchitecture summary

The e200z0h processor utilizes a four stage pipeline for instruction execution. The Instruction Fetch (stage 1), Instruction Decode/Register file Read/Effective Address Calculation (stage 2), Execute/Memory Access (stage 3), and Register Writeback (stage 4) stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

The integer execution unit consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), a 32x32 Hardware Multiplier array, result feed-forward hardware, and a hardware divider.

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

The Instruction Unit contains a PC incrementer and a dedicated Branch Address adder to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching from the BTB is performed to accelerate certain taken branches in the e200z0h. Prefetched instructions are placed into an instruction buffer with 4 entries in e200z0h, each capable of holding a single 32-bit instruction or a pair of 16-bit instructions.

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

Memory load and store operations are provided for byte, halfword, and word (32-bit) data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized. Also, a load-to-use dependency does not incur any pipeline bubbles for most cases.

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

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

14.2.1.1 Block diagram

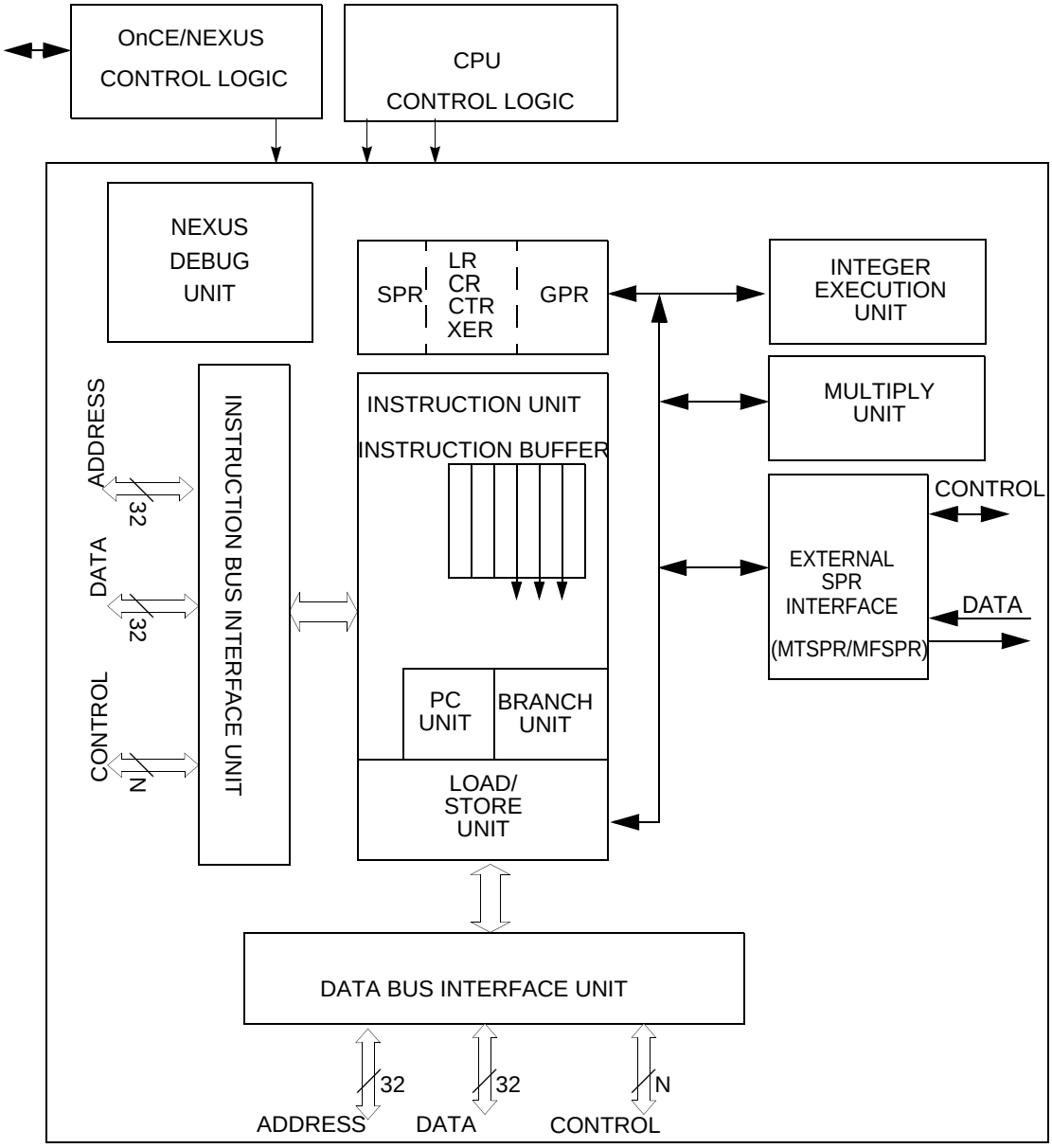


Figure 14-1. e200z0h block diagram

14.2.1.2 Instruction unit features

The features of the e200z0h Instruction unit are:

- 32-bit instruction fetch path supports fetching of one 32-bit instruction per clock, or up to two 16-bit VLE instructions per clock
- Instruction buffer with 4 entries in e200zh0, each holding a single 32-bit instruction, or a pair of 16-bit instructions

- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder, and small branch target buffer logic supporting single cycle of execution of certain branches, two cycles for all others

14.2.1.3 Integer unit features

The e200 integer unit supports single cycle execution of most integer instructions:

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

14.2.1.4 Load/store unit features

The e200 load/store unit supports load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- 32-bit interface to memory (dedicated memory interface on e200z0h)

14.2.1.5 e200z0h system bus features

The features of the e200z0h System Bus interface are as follows:

- Independent Instruction and Data Buses
- AMBA AHB Lite Rev 2.0 Specification with support for ARM v6 AMBA Extensions
  - — Exclusive Access Monitor
  - — Byte Lane Strobes
  - — Cache Allocate Support
- 32-bit address bus plus attributes and control on each bus
- 32-bit read data bus for Instruction Interface
- Separate uni-directional 32-bit read data bus and 32-bit write data bus for Data Interface
- Overlapped, in-order accesses

14.3 Core registers and programmer's model

This section describes the registers implemented in the e200z0h core. It includes an overview of registers defined by the PowerPC Book E architecture, highlighting differences in how these registers are implemented in the e200 core, and provides a detailed description of e200-specific registers. Full descriptions of the architecture-defined register set are provided in the Power Architecture Book E Specification.

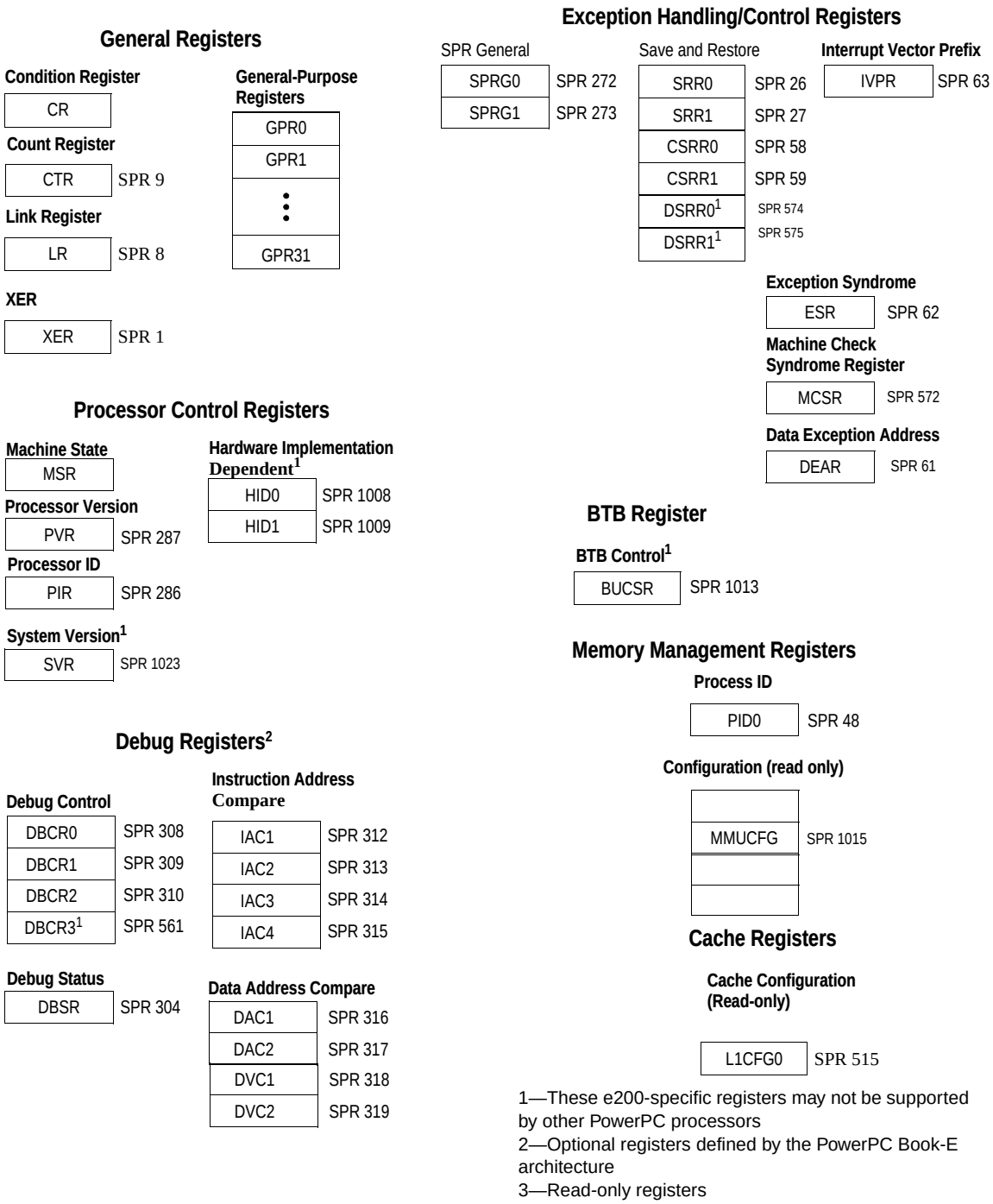
The Power Architecture Book E defines register-to-register operations for all computational instructions. Source data for these instructions are accessed from the on-chip registers or are provided as immediate values embedded in the opcode. The three-register instruction format allows specification of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

Figure 14-2 and Figure 14-3 show the e200 register set including the registers which are accessible while in supervisor mode, and the registers which are accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

NOTE

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

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



1—These e200-specific registers may not be supported by other PowerPC processors
2—Optional registers defined by the PowerPC Book-E architecture
3—Read-only registers

Figure 14-2. e200z0h SUPERVISOR mode program model SPRs

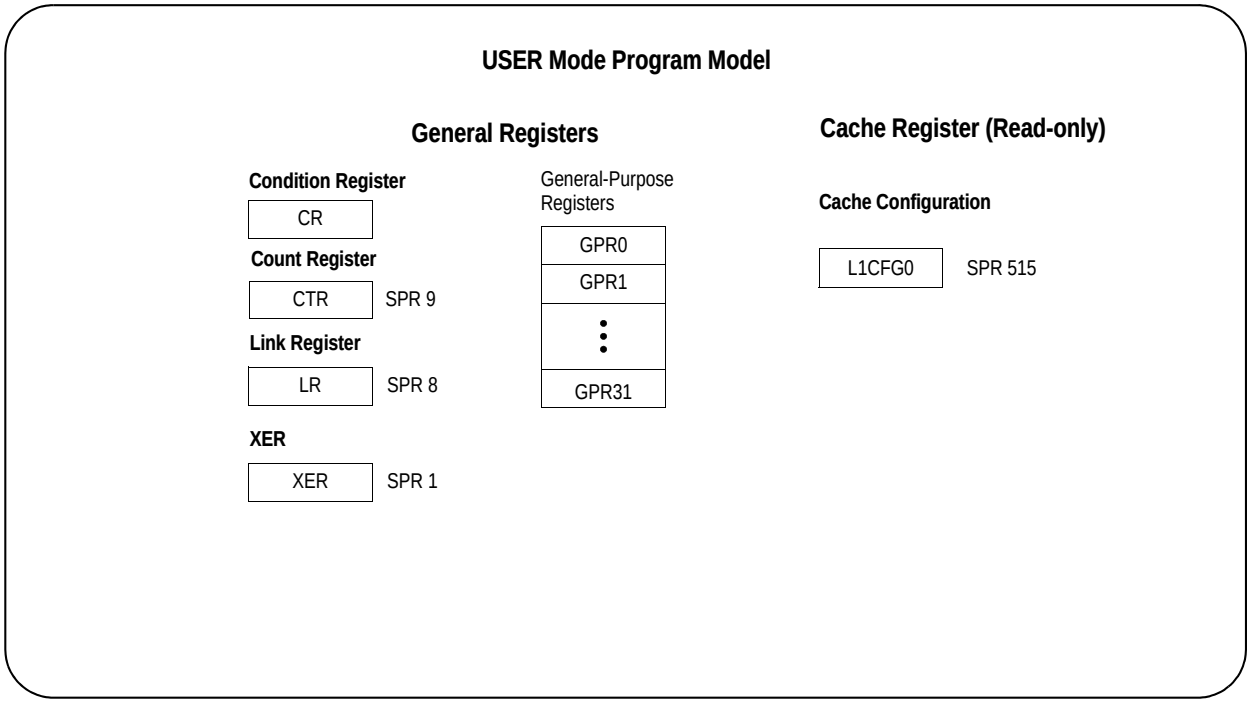


Figure 14-3. e200z0h User mode program model

14.3.1 Unimplemented SPRs and Read-only SPRs

e200z0h fully decodes the SPR field of the **mfspr** and **mtspr** instructions. If the SPR specified is undefined and not privileged, an illegal instruction exception is generated. If the SPR specified is undefined and privileged and the CPU is in user mode (MSR[PR=1]), a privileged instruction exception is generated. If the SPR specified is undefined and privileged and the core is in supervisor mode (MSR[PR=0]), an illegal instruction exception is generated.

For the **mtspr** instruction, if the SPR specified is read-only and not privileged, an illegal instruction exception is generated. If the SPR specified is read-only and privileged and the core is in user mode (MSR[PR=1]), a privileged instruction exception is generated. If the SPR specified is read-only and privileged and the core is in supervisor mode (MSR[PR=0]), an illegal instruction exception is generated.

14.4 Instruction summary

e200z0h supports all VLE instructions described in the *PowerPC™ VLE APU Definition version 1.2 together with the additional instructions for context save/restore.*

Chapter 15
Enhanced Direct Memory Access (eDMA)

15.1 Information specific to this device

This section presents device-specific parameterization, customization, and feature availability information not specifically referenced in the remainder of this chapter.

15.1.1 Device-specific features

- 16 programmable channels

15.2 Introduction

The **DMA** (Direct Memory Access) is a second-generation platform module capable of performing complex data transfers with minimal intervention from a host processor via "*n*" programmable channels. Intended for use as part of the Standard Product Platform (SPP), the hardware microarchitecture includes a DMA engine which performs source and destination address calculations, and the actual data movement operations, along with a local memory containing the *transfer control descriptors* (TCD) for the channels. This SRAM-based implementation is used to minimize the overall module size.

Figure 15-1 is a block diagram of the DMA module.

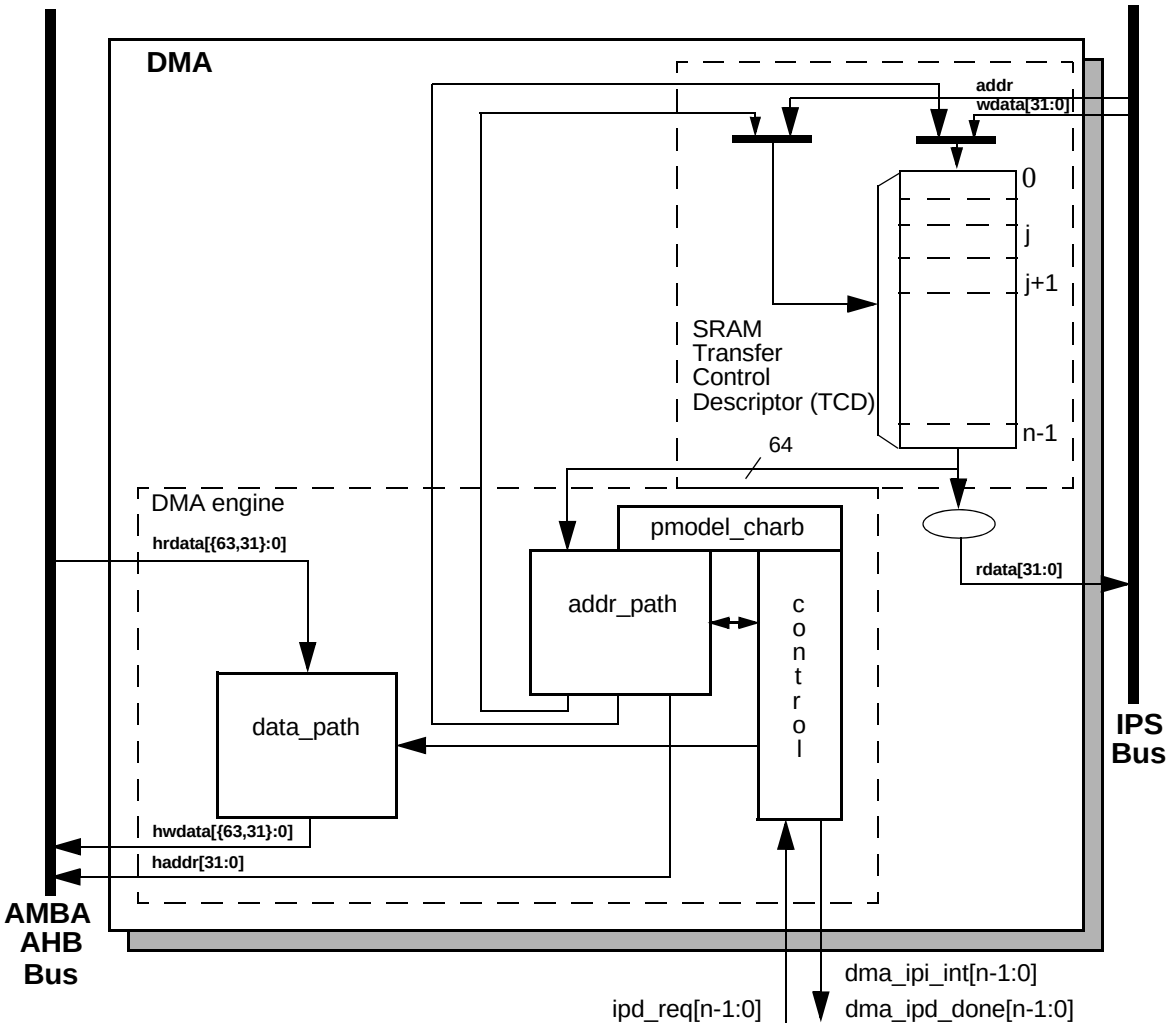


Figure 15-1. DMA block diagram

15.2.1 Overview

The DMA is a highly programmable data transfer engine, which has been optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known, and is *not* defined within the data packet itself. The DMA hardware supports:

- Single design supporting 16-, 32- and 64-channel implementations, dependent on size of the TCD memory and design parameters
- Connections to the AMBA-AHB crossbar switch for bus mastering the data movement, slave bus for programming the module
  - Parameterized support for 32- and 64-bit AMBA-AHB datapath widths
- 32-byte transfer control descriptor per channel stored in local memory
- 32 bytes of data registers, used as temporary storage to support burst transfers

Throughout this document, *n* is used to reference the channel number. Additionally, data sizes are defined as byte (8-bit), halfword (16-bit), word (32-bit) and doubleword (64-bit).

15.2.2 Features

The DMA module supports the following features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source, destination addresses, transfer size, plus support for enhanced addressing modes
- Transfer control descriptor organized to support two-deep, nested transfer operations
  - An *inner* data transfer loop defined by a “minor” byte transfer count
  - An *outer* data transfer loop defined by a “major” iteration count
- Channel service request via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
    – Independent channel linking at end of minor loop and/or major loop
  - Peripheral-paced hardware requests (one per channel)
  - For all three methods, *one service request per execution of the minor loop is required*
- Support for fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Error terminations are optionally enabled per channel, and logically summed together to form a small number of error interrupt outputs
- Support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software or hardware

The structure of the transfer control descriptor is fundamental to the operation of the DMA module. It is defined below in a ‘C’ pseudo-code specification, where `int` refers to a 32-bit variable (unless noted otherwise) and `short` is a 16-bit variable.

NOTE

To compile these structures, change any periods '.' in the variable name to underscores '_'.

```
typedef union {
    struct {              /* citer.e_link = 1 */
        unsigned short citer.linkch:6; /* link channel number, */
        unsigned short citer:9; /* current (“major”) iteration count */
    } minor_link_enabled;                  /* channel link at end of the minor loop */
    struct {              /* citer.e_link = 0 */
        unsigned short citer:15; /* current (“major”) iteration count */
    } minor_link_disabled;                 /* no linking at end of the minor loop */
} t_minor_link_citer;
```

```
typedef union {
    struct {              /* biter.e_link = 1 */
        unsigned short biter.linkch:6; /* link channel number, */
        unsigned short biter:9; /* beginning (“major”) iteration count */
    } init_minor_link_enabled;              /* channel link at end of the minor loop */
    struct {              /* biter.e_link = 0 */
        unsigned short biter:15; /* beginning (“major”) iteration count */
    } init_minor_link_disabled;             /* no linking at end of the minor loop */
} t_minor_link_biter;

typedef struct {
         unsigned intsaddr;/* source address */
         unsigned intsmod:5;/* source address modulo */
         unsigned intssize:3;/* source transfer size */
         unsigned intdmod:5;/* destination address modulo */
         unsigned intdsize:3;/* destination transfer size */
         short    soff;    /* signed source address offset */
         unsigned intnbytes;/* inner (“minor”) byte count */
         int      slast;   /* last source address adjustment */
         unsigned intdaddr;/* destination address */
         unsigned shortciter.e_link:1;/* enable channel linking on minor loop */
         t_minor_link_citerminor_link_citer;/* conditional current iteration count */
         short    doff;    /* signed destination address offset */
         int      dlast_sga;/* last destination address adjustment, or
                            scatter/gather address (if e_sg = 1) */
         unsigned shortbiter.e_link:1;/* beginning channel link enable */
         t_minor_link_biterminor_link_biter;/* beginning (“major”) iteration count */
         unsigned intbwc:2;/* bandwidth control */
         unsigned intmajor.linkch:6;/* link channel number */
         unsigned intdone:1;/* channel done */
         unsigned intactive:1;/* channel executing */
         unsigned intmajor.e_link:1;/* enable channel linking on major loop*/
         unsigned inte_sg:1;/* enable scatter/gather descriptor */
         unsigned intd_req:1;/* disable ipd_req when done */
         unsigned intint_half:1;/* interrupt on citer = (biter >> 1) */
         unsigned intint_maj:1;/* interrupt on major loop completion */
         unsigned intstart:1;/* explicit channel start */
} tcd                         /* transfer_control_descriptor */
```

The basic operation of a channel is defined as:

1. The channel is initialized by software loading the transfer control descriptor into the DMA’s programming model, memory-mapped through the IPS space, and implemented as local memory.

2. The channel requests service; either explicitly by software, a peripheral request or a linkage from another channel.

NOTE

The major loop executes one iteration per service request.

3. The contents of the transfer control descriptor for the activated channel is read from the local memory and loaded into the DMA engine’s internal register file.

4. The DMA engine executes the data transfer defined by the transfer control descriptor, reading from the source and writing to the destination. The number of iterations in the minor loop is automatically calculated by the DMA engine. The number of iterations within the minor loop is a function of the number

of bytes to transfer (nbytes), the source size (ssize) and the destination size (dsize). The completion of the minor loop is equal to one iteration of the major loop.

5. At the conclusion of the minor loop's execution, certain fields of the transfer control descriptor are written back to the local TCD memory.

The process (steps 2-5) is repeated until the outer major loop's iteration count is exhausted. At that time, additional processing steps are completed, e.g., the optional assertion of an interrupt request signaling the transfer's completion, final adjustments to the source and destination addresses, etc. A more detailed description of the channel processing is listed in the pseudo-code below. This simplified example is intended to represent basic data transfers. Detailed processing associated with the error handling is omitted.

```
/* the given DMAchannel is requesting service by the software assertion of the
   tcd[channel].start bit, the assertion of an enabled ipd_req from a device, or
   the implicit assertion of a channel-to-channel link */

/* begin by reading the transfer control descriptor from the local RAM
   into the local dma_engine registers */
dma_engine        = read_from_local_memory [channel];
dma_engine.active = 1;                /* set active flag */
dma_engine.done   = 0;                /* clear done flag */

/* check the transfer control descriptor for consistency */
if (dma_engine.config_error == 0) {

    / * begin execution of the inner “minor” loop transfers */
    {

        /* convert the source transfer size into a byte count */
        switch (dma_engine.ssize) {
        case 0:                        /* 8-bit transfer */
            src_xfr_size = 1;
            break;
        case 1:                        /* 16-bit transfer */
            src_xfr_size = 2;
            break;
        case 2:                        /* 32-bit transfer */
            src_xfr_size = 4;
            break;
        case 3:                        /* 64-bit transfer */
            src_xfr_size = 8;
            break;
        case 4:                        /* 16-byte burst transfer */
            src_xfr_size = 16;
            break;
        case 5:                        /* 32-byte burst transfer */
            src_xfr_size = 32;
            break;
        }

        /* convert the destination transfer size into a byte count */
        switch (dma_engine.dsize) {
        case 0:                        /* 8-bit transfer */
            dest_xfr_size = 1;
            break;
```

```
case 1:                          /* 16-bit transfer */
    dest_xfr_size = 2;
    break;
case 2:                          /* 32-bit transfer */
    dest_xfr_size = 4;
    break;
case 3:                          /* 64-bit transfer */
    dest_xfr_size = 8;
    break;
case 4:                          /* 16-byte burst transfer */
    dest_xfr_size = 16;
    break;
case 5:                          /* 32-byte burst transfer */
    dest_xfr_size = 32;
    break;
}

/* determine the larger of the two transfer sizes, this value reflects */
/* the number of bytes transferred per read->write sequence. */
/* number of iterations of the minor loop = nbytes / xfer_size */
if (dma_engine.ssize < dma_engine.dsize)
    xfr_size = dest_xfer_size;
 else
    xfr_size = src_xfer_size;

/* process the source address, READ data into the buffer*/

/* read “xfr_size” bytes from the source */
/* if the ssize < dsize, do multiple reads to equal the dsize */
/* if the ssize => dsize, do a single read of source data */
number_of_source_reads = xfer_size / src_xfer_size;

for (number_of_source_reads) {
    dma_engine.data = read_from_amba-ahb (dma_engine.saddr, src_xfr_size);

    /* generate the next-state source address */
    /* sum the current saddr with the signed source offset */
    ns_addr = dma_engine.saddr + (int) dma_engine.soff; }

    /* if enabled, apply the power-of-2 modulo to the next-state addr */
    if (dma_engine.smod != 0)
        address_select = (1 << dma_engine.smod) - 1; }
    else
        address_select = 0xffff_ffff;

    dma_engine.saddr = ns_addr          &  address_select
                | dma_engine.saddr & ~address_select; }
}


 /* process the destination address, WRITE data from buffer */

 /* write “xfr_size” bytes to the destination */
 /* if the dsize < ssize, do multiple writes to equal the ssize */
 /* if the dsize => ssize, do a single write of dest data */
 number_of_dest_writes = xfer_size / dest_xfer_size;
```

```
    for (number_of_dest_writes) {
        write_to_amba-ahb (dma_engine.daddr, dest_xfr_size) = dma_engine.data;

        /* generate the next-state destination address */
        /* sum the current daddr with the signed destination offset */
        ns_addr = dma_engine.daddr + (int) dma_engine.doff;

        /* if enabled, apply the power-of-2 modulo to the next-state dest addr */
        if (dma_engine.dmod != 0)
            address_select = (1 << dma_engine.dmod) - 1;
        else
            address_select = 0xffff_ffff;

        dma_engine.daddr = ns_addr           &  address_select
                    | dma_engine.daddr & ~address_select;
    }
    if (cancel_transfer)
        break;

    /* check for a higher priority channel to service if: */
    /* 1) preemption is enabled */
    /* 2) in fixed arbitration mode */
    /* 3) a higher priority channel is requesting service */
    /* 4) not already servicing a preempting channel */
    if ((DCHPRIn.ecp = 1)  &  fixed_arbitration_mode
        higher_pri_request & ~current_channel_is_preempt)
        service_preempt_channel;

    /* the bandwidth control field determines when the next read/write occurs */
    if (dma_engine.bwc > 1)
        stall_dma_engine (1 << dma_engine.bwc);

    /* decrement the minor loop byte count */
    dma_engine.nbytes = dma_engine.nbytes - xfr_size;

}while (dma_engine.nbytes > 0) /* end of minor inner loop */

dma_engine.citer--;            /* decrement major loop iteration count */

/* if the major loop is not yet exhausted, update certain TCD values in the RAM */
if (dma_engine.citer != 0) {
    write_to_local_memory [channel].saddr = dma_engine.saddr;
    write_to_local_memory [channel].daddr = dma_engine.daddr;
    write_to_local_memory [channel].citer = dma_engine.citer;

                    /* if minor loop linking is enabled, make the channel link */
    if (dma_engine.citer.e_link)
        TCD[citer.linkch].start = 1;     /* specified channel service req */

    /* check for interrupt assertion if half of the major iterations are done */
    if (dma_engine.int_half && (dma_engine.citer == (dma_engine.biter >> 1)))
        generate_interrupt (channel);

    dma_engine.active = 0;         /* clear the channel busy flag */

 }
 else { /* major loop is complete, dma_engine.citer == 0 */
```

```
        /* since the major loop is complete, perform the final address adjustments */

        /* sum the current {src,dst} addresses with “last” adjustment */
        write_to_local_memory [channel].saddr = dma_engine.saddr + dma_engine.slast;
        write_to_local_memory [channel].daddr = dma_engine.daddr + dma_engine.dlast;
        /* restore the major iteration count to the beginning value */
        write_to_local_memory [channel].citer = dma_engine.biter;

        /* check for interrupt assertion at completion of the major iteration */
        if (dma_engine.int_maj)
          generate_interrupt (channel);

         /* check if the ipd_req is to be disabled at completion of the major iteration */
        if (dma_engine.d_req)
            DMAERQ [channel] = 0;

        /* check for a scatter/gather transfer control descriptor */
        if (dma_engine.e_sg) {
          /* load new transfer control descriptor from the address defined by dlast_sga */
            write_to_local_memory [channel] =
                                       read_from_amba-ahb(dma_engine.dlast_sga,32);
        }
        if (dma_engine.major.e_link)
            TCD[major.linkch].start = 1;       /* specified channel service req */

        dma_engine.active = 0;                /* clear the channel busy flag */
        dma_engine.done = 1;                  /* set the channel done flag */

}
else { /* configuration error detected, abort the channel */
        dma_engine.error_status = error_type; /* record the error */
        dma_engine.active = 0;                /* clear the channel busy flag */
        /* check for interrupt assertion on error */
        if (dma_engine.int_err)
            generate_interrupt (channel);
}
```

For more details, consult Section 15.3.1, Register descriptions, and Section 15.4, Functional description.

15.3 Memory map/register definition

The DMA’s programming model is partitioned into two sections, both mapped into the slave bus space: the first region defines a number of registers providing control functions, while the second region corresponds to the local transfer control descriptor memory.

Reading an *unimplemented* register bit or memory location will return the value of zero. Write the value of zero to *unimplemented* register bits. Any access to a *reserved* memory location will result in a bus error. *Reserved* memory locations are indicated in the memory map. For 16- and 32-channel implementations, reserved memory also includes the high order “H” registers containing channels 63–32 data (DMAERQH, DMAEEIH, DMAINTH, and DMAERRH).

Many of the control registers have a bit width that matches the number of channels implemented in the module; that is, 16, 32, or 64 bits in size. Registers associated with a 64-channel design are implemented as two 32-bit registers, and include an “H” and “L” suffixes, signaling the “high” and “low” portions of

the control function. The descriptions in this section define the 64-channel implementation. For 16- or 32-channel designs, the unused bits are not implemented: reads return zeroes, and writes are ignored.

The DMA module does not include any logic which provides access control. Rather, this function is supported using the standard access control logic provided by the PBRIDGE controller.

Table 15-1 is a 32-bit view of the DMA's memory map.

Table 15-1. DMA 32-bit memory map

DMA Offset	Register			
0x0000	DMA Control Register (DMACR)			
0x0004	DMA Error Status (DMAES)			
0x0008	DMA Enable Request High (DMAERQH, Channels 63-32)			
0x000C	DMA Enable Request Low (DMAERQL, Channels 31-00)			
0x0010	DMA Enable Error Interrupt High (DMAEEIH, Channels 63-32)			
0x0014	DMA Enable Error Interrupt Low (DMAEEIL, Channels 31-00)			
0x0018	DMA Set Enable Request (DMASERQ)	DMA Clear Enable Request (DMACERQ)	DMA Set Enable Error Interrupt (DMASEEI)	DMA Clear Enable Error Interrupt (DMACEEI)
0x001c	DMA Clear Interrupt Request (DMACINT)	DMA Clear Error (DMACERR)	DMA Set Start Bit (DMASSRT)	DMA Clear Done Status Bit (DMACDNE)
0x0020	DMA Interrupt Request High (DMAINTH, Channels 63-32)			
0x0024	DMA Interrupt Request Low (DMAINTL, Channels 31-00)			
0x0028	DMA Error High (DMAERRH, Channels 63-32)			
0x002C	DMA Error Low (DMAERRL, Channels 31-00)			
0x0030	DMA Hardware Request Status High (DMAHRSH, Channels 63-32)			
0x0034	DMA Hardware Request Status Low (DMAHRSL, Channels 31-00)			
0x0038	DMA General Purpose Output Register (DMAGPOR)			
0x003C-0x00FC	Reserved			
0x0100	DMA Channel 0 Priority (DCHPRI0)	DMA Channel 1 Priority (DCHPRI1)	DMA Channel 2 Priority (DCHPRI2)	DMA Channel 3 Priority (DCHPRI3)
0x0104	DMA Channel 4 Priority (DCHPRI4)	DMA Channel 5 Priority (DCHPRI5)	DMA Channel 6 Priority (DCHPRI6)	DMA Channel 7 Priority (DCHPRI7)
0x0108	DMA Channel 8 Priority (DCHPRI8)	DMA Channel 9 Priority (DCHPRI9)	DMA Channel 10 Priority (DCHPRI10)	DMA Channel 11 Priority (DCHPRI11)
0x010c	DMA Channel 12 Priority (DCHPRI12)	DMA Channel 13 Priority (DCHPRI13)	DMA Channel 14 Priority (DCHPRI14)	DMA Channel 15 Priority (DCHPRI15)
0x0110	DMA Channel 16 Priority (DCHPRI16)	DMA Channel 17 Priority (DCHPRI17)	DMA Channel 18 Priority (DCHPRI18)	DMA Channel 19 Priority (DCHPRI19)
0x0114	DMA Channel 20 Priority (DCHPRI20)	DMA Channel 21 Priority (DCHPRI21)	DMA Channel 22 Priority (DCHPRI22)	DMA Channel 23 Priority (DCHPRI23)
0x0118	DMA Channel 24 Priority (DCHPRI24)	DMA Channel 25 Priority (DCHPRI25)	DMA Channel 26 Priority (DCHPRI26)	DMA Channel 27 Priority (DCHPRI27)
0x011C	DMA Channel 28 Priority (DCHPRI28)	DMA Channel 29 Priority (DCHPRI29)	DMA Channel 30 Priority (DCHPRI30)	DMA Channel 31 Priority (DCHPRI31)
0x0120	DMA Channel 32 Priority (DCHPRI32)	DMA Channel 33 Priority (DCHPRI33)	DMA Channel 34 Priority (DCHPRI34)	DMA Channel 35 Priority (DCHPRI35)

Table 15-1. DMA 32-bit memory map (continued)

DMA Offset	Register			
0x0124	DMA Channel 36 Priority (DCHPRI36)	DMA Channel 37 Priority (DCHPRI37)	DMA Channel 38 Priority (DCHPRI38)	DMA Channel 39 Priority (DCHPRI39)
0x0128	DMA Channel 40 Priority (DCHPRI40)	DMA Channel 41 Priority (DCHPRI41)	DMA Channel 42 Priority (DCHPRI42)	DMA Channel 43 Priority (DCHPRI43)
0x012C	DMA Channel 44 Priority (DCHPRI44)	DMA Channel 45 Priority (DCHPRI45)	DMA Channel 46 Priority (DCHPRI46)	DMA Channel 47 Priority (DCHPRI47)
0x0130	DMA Channel 48 Priority (DCHPRI48)	DMA Channel 49 Priority (DCHPRI49)	DMA Channel 50 Priority (DCHPRI50)	DMA Channel 51 Priority (DCHPRI51)
0x0134	DMA Channel 52 Priority (DCHPRI52)	DMA Channel 53 Priority (DCHPRI53)	DMA Channel 54 Priority (DCHPRI54)	DMA Channel 55 Priority (DCHPRI55)
0x0138	DMA Channel 56 Priority (DCHPRI56)	DMA Channel 57 Priority (DCHPRI57)	DMA Channel 58 Priority (DCHPRI58)	DMA Channel 59 Priority (DCHPRI59)
0x013C	DMA Channel 60 Priority (DCHPRI60)	DMA Channel 61 Priority (DCHPRI61)	DMA Channel 62 Priority (DCHPRI62)	DMA Channel 63 Priority (DCHPRI63)
0x0140–0x0FFC	Reserved			
0x1000–0x11FC	TCD00-TCD15			
0x1200–0x13FC	TCD16-TCD31			
0x1400–0x15FC	TCD32-TCD47			
0x1600–0x17FC	TCD48-TCD63			

15.3.1 Register descriptions

15.3.1.1 DMA Control Register (DMACR) register

The 32-bit DMACR defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests in groups of 16 channels. The 64- and 32-channel configurations have four groups (3,2,1,0) and two groups (1,0), respectively; the 16 channel configuration has only one group (0). Group 3 contains channels 63-48, group 2 contains channels 47-32, group 1 contains channels 31-16, and group 0 contains channels 15-0.

Arbitration within a group can be configured to use either a fixed-priority or a round-robin selection. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The priorities are assigned by the channel priority registers (see Section 15.3.1.17, DMA Channel n Priority (DCHPRIn), n = 0,..., {15,31,63} registers). In round-robin arbitration mode, the channel priorities are ignored and the channels within each group are cycled through without regard to priority.

The group priorities operate in a similar fashion. In group fixed-priority arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPri registers. All group priorities must have unique values prior to any channel service requests occur, otherwise a configuration error will be reported. Unused group priority registers, per configuration, are unimplemented in the DMACR. In group round-robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Minor loop offsets are address offset values added to the final source address (saddr) or destination address (daddr) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (mloff) is added to the final source address (saddr), or the final destination address (daddr), or both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (slast and dlast_sga) are used to compute the next saddr and daddr values.

When minor loop mapping is enabled (DMACR[EMLM] = 1), TCD*n* word2 is redefined. A portion of TCD*n* word2 is used to specify multiple fields: an source enable bit (smloe) to specify the minor loop offset should be applied to the source address (saddr) upon minor loop completion, an destination enable bit (dmloe) to specify the minor loop offset should be applied to the destination address (daddr) upon minor loop completion, and the sign extended minor loop offset value (mloff). The same offset value (mloff) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (smloe set or dmloe set), the nbytes field is reduced to 10 bits. When both minor loop offsets are disabled (smloe cleared and dmloe cleared), the nbytes field is a 30-bit vector.

When minor loop mapping is disabled (DMACR[EMLM] = 0), all 32 bits of TCD*n* word2 are assigned to the nbytes field. See Section 15.3.1.18, Transfer Control Descriptor (TCD), for more details.

See Figure 15-2 and Table 15-2 for the DMACR definition.

Address: Base + 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GRP3PRI		GRP2PRI		GRP1PRI		GRP0PRI		EML M	CLM	HALT	HOE	ERG A	ERC A	EDB G	EBW
W																
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 15-2. DMA Control Register (DMACR)

Table 15-2. DMA Control Register (DMACR) field descriptions

Field	Description
CX	Cancel Transfer 0 Normal operation. 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
ECX	Error Cancel Transfer 0 Normal operation. 1 Cancel the remaining data transfer in the same fashion as the CX cancel transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel has been honored. In addition to cancelling the transfer, the ECX treats the cancel as an error condition; thus updating the DMAES register and generating an optional error interrupt (see Section 15.3.1.2, DMA Error Status (DMAES) register).

Table 15-2. DMA Control Register (DMACR) field descriptions

Field	Description
GRP3PRI	Channel Group 3 Priority Group 3 priority level when fixed priority group arbitration is enabled.
GRP2PRI	Channel Group 2 Priority Group 2 priority level when fixed priority group arbitration is enabled.
GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
EMLM	Enable Minor Loop Mapping 0 Minor loop mapping disabled. TCDn.word2 is defined as a 32-bit nbytes field. 1 Minor loop mapping enabled. When set, TCDn.word2 is redefined to include individual enable fields, an offset field and the nbytes field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The nbytes field is reduced when either offset is enabled.
CLM	Continuous Link Mode 0 A minor loop channel link made to itself will go through channel arbitration before being activated again. 1 A minor loop channel link made to itself will not go through channel arbitration before being activated again. Upon minor loop completion the channel will active again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
HALT	Halt DMA Operations 0 Normal operation. 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution will resume when the HALT bit is cleared.
HOE	Halt On Error 0 Normal operation. 1 Any error will cause the HALT bit to be set. Subsequently, all service requests will be ignored until the HALT bit is cleared.
ERGA	Enable Round Robin Group Arbitration 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
EDBG	Enable Debug 0 The assertion of the ipg_debug input is ignored. 1 The assertion of the ipg_debug input causes the DMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution will resume when either the ipg_debug input is negated or the EDBG bit is cleared.
EBW	Enable Buffered Writes 0 The bufferable write signal (hprot[2]) is not asserted during AMBA AHB writes. 1 The bufferable write signal (hprot[2]) is asserted on all AMBA AHB writes except for the last write sequence.

15.3.1.2 DMA Error Status (DMAES) register

The DMAES register provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer control descriptor or an illegal priority register setting in fixed arbitration mode) or an error termination to a bus master read or write cycle.

A configuration error is caused when the starting source or destination address, source or destination offsets, minor loop byte count and the transfer size represent an inconsistent state. The addresses and offsets must be aligned on 0-modulo-transfer_size boundaries, and the minor loop byte count must be a multiple of the source and destination transfer sizes. All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively. In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled. If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (dlast_sga) is not aligned on a 32-byte boundary. If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD.citer.e_link bit does not equal the TCD.biter.e_link bit. All configuration error conditions except scatter/gather and minor loop link error are reported as the channel is activated and assert an error interrupt request, if enabled. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the DMA engine with the current source address, destination address and current iteration count at the point of the fault. When a system bus error occurs, the channel is terminated after the read or write transaction which is already pipelined after errant access, has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write will execute using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence will execute before the channel is terminated due to the destination bus error.

A transfer may be cancelled by software via the DMACR[CX] bit or hardware via the **dma_cancel_xfer** input signal. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the DMAES register is updated with the cancelled channel number and error cancel bit is set. The TCD of a cancelled channel has the source address and destination address of the last transfer saved in the TCD. It is the responsibility of the user to initialize the TCD again should the channel need to be restarted because the aforementioned fields have been modified by the DMA engine and no longer represent the original parameters. When a transfer is cancelled via the error cancel transfer mechanism (setting the DMACR[ECX] or asserting the **dma_err_cancel_xfer** input), the channel number is loaded into the ERRCHN field and the ECX and VLD bits are set are set in the DMAES register. In addition, an error interrupt may be generated if enabled. See Section 15.3.1.14, DMA Error (DMAERRH, DMAERRL) registers, for error interrupt details.

The occurrence of any type of error causes the DMA engine to immediately stop, and the appropriate channel bit in the DMA Error register to be asserted. At the same time, the details of the error condition are loaded into the DMAES register. The major loop complete indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are *not* affected when an error is detected. See Figure 15-3 and Table 15-3 for the DMAES definition.

Address: Base + 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	0	ERRCHN					SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-3. DMA Error Status (DMAES) register

Table 15-3. DMA Error Status (DMAES) field descriptions

Name	Description
VLD	Logical OR of all DMAERRH and DMAERRL status bits. 0 No DMAERR bits are set. 1 At least one DMAERR bit is set indicating a valid error exists that has not been cleared.
ECX	Transfer cancelled 0 No cancelled transfers. 1 The last recorded entry was a cancelled transfer via the error cancel transfer input.
GPE	Group Priority Error 0 No group priority error. 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
CPE	Channel Priority Error 0 No channel priority error. 1 The last recorded error was a configuration error in the channel priorities within a group. All channel priorities within a group are not unique.
ERRCHN[0:5]	Error Channel Number or Cancelled Channel Number The channel number of the last recorded error (excluding GPE and CPE errors) or last recorded transfer that was error cancelled.
SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD.saddr field. TCD.saddr is inconsistent with TCD.ssize.
SOE	Source Offset Error 0 No source offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.soff field. TCD.soff is inconsistent with TCD.ssize.
DAE	Destination Address Error 0 No destination address configuration error. 1 The last recorded error was a configuration error detected in the TCD.daddr field. TCD.daddr is inconsistent with TCD.dsize.

Table 15-3. DMA Error Status (DMAES) field descriptions (continued)

Name	Description
DOE	Destination Offset Error 0 No destination offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.doff field. TCD.doff is inconsistent with TCD.dsize.
NCE	Nbytes/Citer Configuration Error 0 No nbytes/citer configuration error. 1 The last recorded error was a configuration error detected in the TCD.nbytes or TCD.citer fields. TCD.nbytes is not a multiple of TCD.ssize and TCD.dsize, or TCD.citer is equal to zero, or TCD.citer.e_link is not equal to TCD.biter.e_link.
SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error. 1 The last recorded error was a configuration error detected in the TCD.dlast_sga field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD.e_sg is enabled. TCD.dlast_sga is not on a 32-byte boundary.
SBE	Source Bus Error 0 No source bus error. 1 The last recorded error was a bus error on a source read.
DBE	Destination Bus Error 0 No destination bus error. 1 The last recorded error was a bus error on a destination write.

15.3.1.3 DMA Enable Request (DMAERQH, DMAERQL) registers

The DMAERQ{H,L} registers provide a bit map for the implemented channels {16,32,64} to enable the request signal for each channel. DMAERQH supports channels 63-32, while DMAEQRL covers channels 31-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the DMASERQ and DMACERQ registers. The DMA{S,C}ERQ registers are provided so that the request enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the DMAERQ{H,L} registers.

Both the DMA request input signal and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does *not* affect a channel service request made explicitly through software or a linked channel request. See Figure 15-4, Figure 15-5, and Table 15-4 for the DMAERQ definition.

Address: Base + 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ 63	ERQ 62	ERQ 61	ERQ 60	ERQ 59	ERQ 58	ERQ 57	ERQ 56	ERQ 55	ERQ 54	ERQ 53	ERQ 52	ERQ 51	ERQ 50	ERQ 49	ERQ 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 47	ERQ 46	ERQ 45	ERQ 44	ERQ 43	ERQ 42	ERQ 41	ERQ 40	ERQ 39	ERQ 38	ERQ 37	ERQ 36	ERQ 35	ERQ 34	ERQ 33	ERQ 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-4. DMA Enable Request High (DMAERQH) register

Address: Base + 0x000C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 15	ERQ 14	ERQ 13	ERQ 12	ERQ 11	ERQ 10	ERQ 09	ERQ 08	ERQ 07	ERQ 06	ERQ 05	ERQ 04	ERQ 03	ERQ 02	ERQ 01	ERQ 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-5. DMA Enable Request Low (DMAERQL) register

Table 15-4. DMA Enable Request (DMAERQH, DMAERQL) field descriptions

Name	Description
ERQ*n*, n = 0,... 15 n = 0,... 31 n = 0,... 63	Enable DMA Request n 0 The DMA request signal for channel n is disabled. 1 The DMA request signal for channel n is enabled.

As a given channel completes the processing of its major iteration count, there is a flag in the transfer control descriptor that may affect the ending state of the DMAERQ bit for that channel. If the TCD.d_req bit is set, then the corresponding DMAERQ bit is cleared, disabling the DMA request; else if the d_req bit is cleared, the state of the DMAERQ bit is unaffected.

15.3.1.4 DMA Enable Error Interrupt (DMAEEIH, DMAEEIL) registers

The DMAEEI{H,L} registers provide a bit map for the implemented channels {16,32,64} to enable the error interrupt signal for each channel. DMAEEIH supports channels 63-32, while DMAEEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the DMASEEI and DMACEEI registers. The DMA{S,C}EEI registers are provided so that the error interrupt enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the DMAEEI{H,L} registers.

Both the DMA error indicator and this error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted. See Figure 15-6, Figure 15-7, and Table 15-5 for the DMAEEI definition.

Address: Base + 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-6. DMA Enable Error Interrupt High (DMAEEIH) register

Address: Base + 0x0014 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-7. DMA Enable Error Interrupt Low (DMAEEIL) Register

Table 15-5. DMA Enable Error Interrupt (DMAEEIH, DMAEEIL) field descriptions

Name	Description
EEI*n*, n = 0,... 15 n = 0,... 31 n = 0,... 63	Enable Error Interrupt *n* 0 The error signal for channel n does not generate an error interrupt. 1 The assertion of the error signal for channel n generate an error interrupt request.

15.3.1.5 DMA Set Enable Request (DMASERQ) register

The DMASERQ register provides a simple memory-mapped mechanism to set a given bit in the DMAERQ{H,L} registers to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQ{H,L} register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of DMAERQ{H,L} to be asserted. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 15-8 and Table 15-6 for the DMASERQ definition.

Address: Base + 0x0018 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	SERQ						
Reset	0	0	0	0	0	0	0	0

Figure 15-8. DMA Set Enable Request (DMASERQ) register

Table 15-6. DMA Set Enable Request (DMASERQ) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6-0
SERQ[0:6]	Set Enable Request 0-63 Set the corresponding bit in DMAERQ{H,L} 64-127 Set all bits in DMAERQ{H,L}

15.3.1.6 DMA Clear Enable Request (DMACERQ) register

The DMACERQ register provides a simple memory-mapped mechanism to clear a given bit in the DMAERQ{H,L} registers to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQ{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERQ{H,L} to be zeroed, disabling all DMA request inputs. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 15-9 and Table 15-7 for the DMACERQ definition.

Address: Base + 0x0019 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	CERQ[0:6]						
Reset	0	0	0	0	0	0	0	0

Figure 15-9. DMA Clear Enable Request (DMACERQ) register

Table 15-7. DMA Clear Enable Request (DMACERQ) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6-0
CERQ[0:6]	Clear Enable Request 0-63 Clear corresponding bit in DMAERQ{H,L} 64-127 Clear all bits in DMAERQ{H,L}

15.3.1.7 DMA Set Enable Error Interrupt (DMASEEI) register

The DMASEEI register provides a simple memory-mapped mechanism to set a given bit in the DMAEEI{H,L} registers to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAEEI{H,L} register to be set. A data value of 64 to 127

(regardless of the number of implemented channels) provides a global set function, forcing the entire contents of DMAEEI{H,L} to be asserted. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 15-10 and Table 15-8 for the DMASEEI definition.

Address: Base + 0x001A Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	SEEI[0:6]						
Reset	0	0	0	0	0	0	0	0

Figure 15-10. DMA Set Enable Error Interrupt (DMASEEI) register

Table 15-8. DMA Set Enable Error Interrupt (DMASEEI) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6–0
SEEI[0:6]	Set Enable Error Interrupt 0–63 Set the corresponding bit in DMAEEI{H,L} 64–127 Set all bits in DMAEEI{H,L}

15.3.1.8 DMA Clear Enable Error Interrupt (DMACEEI) register

The DMACEEI register provides a simple memory-mapped mechanism to clear a given bit in the DMAEEI{H,L} registers to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAEEI{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAEEI{H,L} to be zeroed, disabling all DMA request inputs. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 15-11 and Table 15-9 for the DMACEEI definition.

Address: Base + 0x001B Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	CEEI[0:6]						
Reset	0	0	0	0	0	0	0	0

Figure 15-11. DMA Clear Enable Error Interrupt (DMACEEI) register

Table 15-9. DMA Clear Enable Error Interrupt (DMACEEI) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6–0
CEEI[0:6]	Clear Enable Error Interrupt 0–63 Clear corresponding bit in DMAEEI{H,L} 64–127 Clear all bits in DMAEEI{H,L}

15.3.1.9 DMA Clear Interrupt Request (DMACINT) register

The DMACINT register provides a simple memory-mapped mechanism to clear a given bit in the DMAINT{H,L} registers to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the DMAINT{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAINT{H,L} to be zeroed, disabling all DMA interrupt requests. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 15-12 and Table 15-10 for the DMACINT definition.

Address: Base + 0x001C Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	CINT[0:6]						
Reset	0	0	0	0	0	0	0	0

Figure 15-12. DMA Clear Interrupt Request (DMACINT) fields

Table 15-10. DMA Clear Interrupt Request (DMACINT) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6–0
CINT[0:6]	Clear Interrupt Request 0–63 Clear the corresponding bit in DMAINT{H,L} 64–127 Clear all bits in DMAINT{H,L}

15.3.1.10 DMA Clear Error (DMACERR) register

The DMACEER register provides a simple memory-mapped mechanism to clear a given bit in the DMAERR{H,L} registers to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the DMAERR{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERR{H,L} to be zeroed, clearing all channel error indicators. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 15-13 and Table 15-11 for the DMACERR definition.

Address: Base + 0x001D Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	CERR[0:6]						
Reset	0	0	0	0	0	0	0	0

Figure 15-13. DMA Clear Error (DMACERR) register

Table 15-11. DMA Clear Error (DMACERR) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6–0
CERR[0:6]	Clear Error Indicator 0-63 Clear corresponding bit in DMAERR{H,L} 64-127 Clear all bits in DMAERR{H,L}

15.3.1.11 DMA Set START Bit (DMASSRT) register

The DMASSRT register provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding Transfer Control Descriptor to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing all START bits to be set. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Table 15-28 for the TCD START bit definition.

Address: Base + 0x001E Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	SSRT						
Reset	0	0	0	0	0	0	0	0

Figure 15-14. DMA Set START Bit (DMASSRT) register

Table 15-12. DMA Set START Bit (DMASSRT) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6–0
SSRT[0:6]	Set START Bit (Channel Service Request) 0–63 Set the corresponding channel's TCD.start 64–127 Set all TCD.start bits

15.3.1.12 DMA Clear DONE Status (DMACDNE) register

The DMACDNE register provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding Transfer Control Descriptor to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing all DONE bits to be cleared. If bit 0 (NOP) is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Table 15-28 for the TCD DONE bit definition.

Address: Base + 0x001F Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	CDNE[0:6]						
Reset	0	0	0	0	0	0	0	0

Figure 15-15. DMA Clear DONE Status (DMACDNE) register

Table 15-13. DMA Clear DONE Status (DMACDNE) field descriptions

Name	Description
NOP	No Operation 0 Normal operation. 1 No operation, ignore bits 6-0
CDNE[0:6]	Clear DONE Status Bit 0–63 Clear the corresponding channel's DONE bit 64-127 Clear all TCD DONE bits

15.3.1.13 DMA Interrupt Request (DMAINTH, DMAINTL) registers

The DMAINT{H,L} registers provide a bit map for the implemented channels {16,32,64} signaling the presence of an interrupt request for each channel. DMAINTH supports channels 63–32, while DMAINTL covers channels 31–00. The DMA engine signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are directly routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any given channel, it is software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the DMACINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the DMACINT register. On writes to the DMAINT, a one in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The DMACINT register is provided so the interrupt request for a *single* channel can easily be cleared without the need to perform a read-modify-write sequence to the DMAINT{H,L} registers. See Figure 15-16, Figure 15-17, and Table 15-14 for the DMAINT definition.

Address: Base + 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INT 63	INT 62	INT 61	INT 60	INT 59	INT 58	INT 57	INT 56	INT 55	INT 54	INT 53	INT 52	INT 51	INT 50	INT 49	INT 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INT 47	INT 46	INT 45	INT 44	INT 43	INT 42	INT 41	INT 40	INT 39	INT 38	INT 37	INT 36	INT 35	INT 34	INT 33	INT 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-16. DMA Interrupt Request High (DMAINTH) register

Address: Base + 0x0024 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INT 15	INT 14	INT 13	INT 12	INT 11	INT 10	INT 09	INT 08	INT 07	INT 06	INT 05	INT 04	INT 03	INT 02	INT 01	INT 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-17. DMA Interrupt Request Low (DMAINTL) register

Table 15-14. DMA Interrupt Request (DMAINTH, DMAINTL) field descriptions

Name	Description
INT*n*, *n* = 0,... 15 *n* = 0,... 31 *n* = 0,... 63	DMA Interrupt Request *n* 0 The interrupt request for channel *n* is cleared. 1 The interrupt request for channel *n* is active.

15.3.1.14 DMA Error (DMAERRH, DMAERRL) registers

The DMAERR{H,L} registers provide a bit map for the implemented channels {16,32,64} signaling the presence of an error for each channel. DMAERRH supports channels 63–32, while DMAERRL covers channels 31–00. The DMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the DMAEEI register, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests which is then routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error interrupt request. Typically, a write to the DMACERR register in the interrupt service routine is used for this purpose. Recall the normal DMA channel completion indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are *not* affected when an error is detected.

The contents of this register can also be polled and a non-zero value indicates the presence of a channel error, regardless of the state of the DMAEEI register. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the DMACERR register. On writes to the DMAERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The DMACERR register is provided so the error indicator for a *single* channel can easily be cleared. See Figure 15-18, Figure 15-19, and Table 15-15 for the DMAERR definition.

Address: Base + 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-18. DMA Error High (DMAERRH) register

Address: Base + 0x002C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
W	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-19. DMA Error Low (DMAERRL) register

Table 15-15. DMA Error (DMAERRH, DMAERRL) field descriptions

Name	Description
ERR*n*, *n* = 0,... 15 *n* = 0,... 31 *n* = 0,... 63	DMA Error *n* 0 An error in channel *n* has not occurred. 1 An error in channel *n* has occurred.

15.3.1.15 DMA Hardware Request Status (DMAHRSH, DMAHRSL) registers

The DMAHRS{H,L} registers provide a bit map for the implemented channels {16,32,64} to show the current hardware request status for each channel. DMAHRSH supports channels 63-32, while DMAHRSL covers channels 31-00. Hardware request status reflects the current state of the registered and qualified (via the DMAERQ field) ipd_req lines as seen by the DMA2's arbitration logic. This view into the hardware request signals may be used for debug purposes.

See Figure 15-20, Figure 15-21, and Figure 15-16 for the DMAHRS definition.

Address: Base + 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HRS 63	HRS 62	HRS 61	HRS 60	HRS 59	HRS 58	HRS 57	HRS 56	HRS 55	HRS 54	HRS 53	HRS 52	HRS 51	HRS 50	HRS 49	HRS 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HRS 47	HRS 46	HRS 45	HRS 44	HRS 43	HRS 42	HRS 41	HRS 40	HRS 39	HRS 38	HRS 37	HRS 36	HRS 35	HRS 34	HRS 33	HRS 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-20. DMA Hardware Request Status High (DMAHRSH) register

Address: Base + 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HRS 15	HRS 14	HRS 13	HRS 12	HRS 11	HRS 10	HRS 09	HRS 08	HRS 07	HRS 06	HRS 05	HRS 04	HRS 03	HRS 02	HRS 01	HRS 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-21. DMA Hardware Request Status Low (DMAHRSL) register

Table 15-16. DMA Hardware Request Status (DMAHRSH, DMAHRSL) field descriptions

Name	Description
HRS*n*, *n* = 0,... 15 *n* = 0,... 31 *n* = 0,... 63	DMA Hardware Request Status *n* 0 A hardware service request for channel *n* is not present. 1 A hardware service request for channel *n* is present. **Note:** The hardware request status reflects the state of the request as seen by the arbitration logic. Therefore, this status is affected by the DMAERQ*n* bit.

15.3.1.16 DMA General Purpose Output Register (DMAGPOR) register

The optional DMAGPOR register provides a general purpose register in the programmer's model that outputs the register contents. The DMAGPOR performs no functions within the DMA2. This general purpose register is enabled when SPP_DMA2_ENABLE_GPOR is defined. This register may be used by the SoC integrator to define and display configuration information.

See Figure 15-22 and Table 15-17 for the DMAGPOR definition.

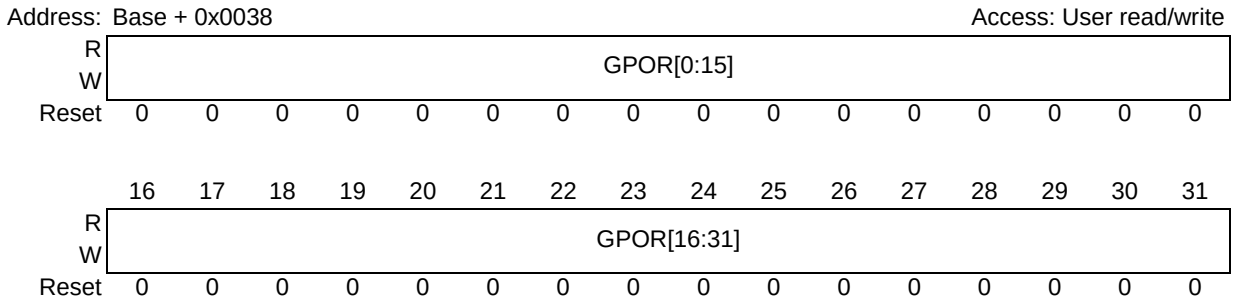


Figure 15-22. DMA General Purpose Output Register (DMAGPOR)

Table 15-17. DMA General Purpose Output Register (DMAGPOR) field descriptions

Name	Description
GPOR[0:31]	DMA General Purpose Output Register The contents of this register is exported out of the DMA2.

15.3.1.17 DMA Channel n Priority (DCHPRIn), n = 0,..., {15,31,63} registers

When the fixed-priority channel arbitration mode is enabled (DMACR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value — 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values, otherwise a configuration error will be reported. The range of the priority value is limited to the values of 0–15. When read, the GRPPRI bits of the DCHPRI*n* register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRI*n* registers. The group priority is assigned in the DMACR. See Figure 15-2 and Table 15-2 for the DMACR definition.

Channel preemption is enabled on a per channel basis by setting the ECP bit in the DCHPRI*n* register. Channel preemption allows the executing channel's data transfers to be temporarily suspended in favor of starting a higher priority channel. After the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel will be suspended and the higher priority channel will be serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is only available when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting the DPA bit in the DCHPRI*n* register. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer; regardless of the lower priority channel's ECP setting. This allows for a pool of low-priority, large data moving channels to be defined. These low-priority channels can be configured to not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true, high-priority channel. See Figure 15-23 and Table 15-18 for the DCHPRI*n* definition.

Address: Base + 0x0100 + *n* Access: User read/write

	0	1	2	3	4	5	6	7
R	ECP	DPA	GRPPRI[0:1]		CHPRI[0:3]			
W								
Reset	0	0	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]

Figure 15-23. DMA Channel *n* Priority (DCHPRI*n*) register

[1] Defaults to channel *n* after reset.

Table 15-18. DMA Channel *n* Priority (DCHPRI*n*) field descriptions

Name	Description
ECP	Enable Channel Preemption 0 Channel *n* cannot be suspended by a higher priority channel's service request. 1 Channel *n* can be temporarily suspended by the service request of a higher priority channel.
DPA	Disable Preempt Ability 0 Channel *n* can suspend a lower priority channel. 1 Channel *n* cannot suspend any channel, regardless of channel priority.
GRPPRI[0:1]	Channel *n* Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored.
CHPRI[0:3]	Channel *n* Arbitration Priority Channel priority when fixed-priority arbitration is enabled.

15.3.1.18 Transfer Control Descriptor (TCD)

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The TCD structure was previously discussed in detail in Section 15.2.2, Features. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel [n-1]. The definitions of the TCD are presented as eight 32-bit values. Table 15-19 is a 32-bit view of the basic TCD structure.

Table 15-19. TCD*n* 32-bit memory structure

DMA Offset	TCD*n* Field	
0x1000 + (32 x *n*) + 0x00	Source Address (saddr)	
0x1000 + (32 x *n*) + 0x04	Transfer Attributes (smod, ssize, dmod, dsize)	Signed Source Address Offset (soff)
0x1000 + (32 x *n*) + 0x08	Signed Minor Loop Offset (smloe, dmloe, mloff)	Inner "Minor" Byte Count (nbytes)
0x1000 + (32 x *n*) + 0x0C	Last Source Address Adjustment (slast)	
0x1000 + (32 x *n*) + 0x10	Destination Address (daddr)	
0x1000 + (32 x *n*) + 0x14	Current "Major" Iteration Count (citer)	Signed Destination Address Offset (doff)
0x1000 + (32 x *n*) + 0x18	Last Destination Address Adjustment/Scatter Gather Address (dlast_sga)	
0x1000 + (32 x *n*) + 0x1C	Beginning "Major" Iteration Count (biter)	Channel Control/Status (bwc, major.linkch, done, active, major.e_link, e_sg, d_req, int_half, int_maj, start)

Figure 15-24 and Table define word 0 of the TCD*n* structure, the saddr field.

Address: DMA_Offset + 0x1000 + (32 × *n*) + 0x00 Access: User read/write

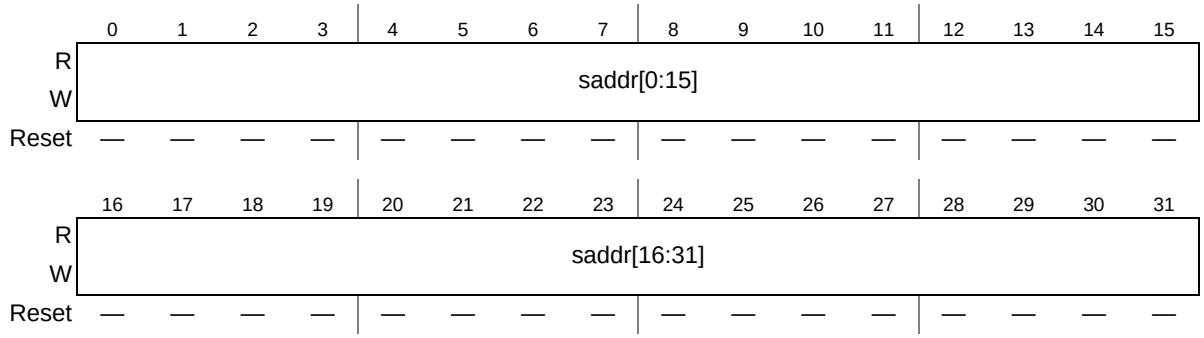


Figure 15-24. TCD*n* Word 0 (TCD*n*.saddr) field

Figure 15-25. TCD*n* Word 0 (TCD*n*.saddr) field

Table 15-20. TCD*n* Word 0 (TCD*n*.saddr) field description

Name	Description
saddr[0:31]	Source address Memory address pointing to the source data.

Figure 15-26 and Table 15-21 define word 1 of the TCD*n* structure, the soff and transfer attribute fields.

Address: DMA_Offset + 0x1000 + (32 × *n*) + 0x04 Access: User read/write

Figure 15-26. TCD*n* Word 1 (TCD*n*.{soff,smod,ssize,dmod,dsize}) fields

Table 15-21. TCD*n* Word 1 (TCD*n*.{smod,ssize,dmod,dsize,soff}) field descriptions

Name	Description
smod[0:4]	Source address modulo 0 Source address modulo feature is disabled. non-0 The value defines a specific address bit which is selected to be either the value after saddr + soff calculation is performed or the original register value. This feature provides the ability to easily implement a circular data queue. For data queues requiring power-of-2 “size” bytes, the queue should be based at a 0-modulo-size address and the smod field set to the appropriate value to freeze the upper address bits. The bit select is defined as ((1 << smod[4:0]) – 1) where a resulting 1 in a bit location selects the next state address for the corresponding address bit location and a 0 selects the original register value for the corresponding address bit location. For this application, the soff is typically set to the transfer size to implement post-increment addressing with the smod function constraining the addresses to a 0-modulo-size range.

Table 15-21. TCD*n* Word 1 (TCD*n*.{smod,ssize,dmod,dsize,soff}) field descriptions (continued)

Name	Description
ssize[0:2]	Source data transfer size 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 16-byte (32-bit AHB bus, WRAP4 burst) 100 Reserved (64-bit AHB bus, reserved) 101 32-byte (if supported by the platform) 110 Reserved 111 Reserved The attempted specification of a 64-bit source size in a 32-bit AMBA AHB bus implementation produces a configuration error. Likewise, the attempted specification of a 16-byte source size in a 64-bit AMBA AHB bus implementation generates a configuration error. The attempted specification of a 32-byte burst on platforms that do not support such a transfer type will result in a configuration error.
dmod[0:4]	Destination address modulo See the smod[5:0] definition.
dsize[0:2]	Destination data transfer size See the ssize[2:0] definition.
soff[16:31]	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

Figure 15-27 and Table define word 2 of the TCD*n* structure, the nbytes field.

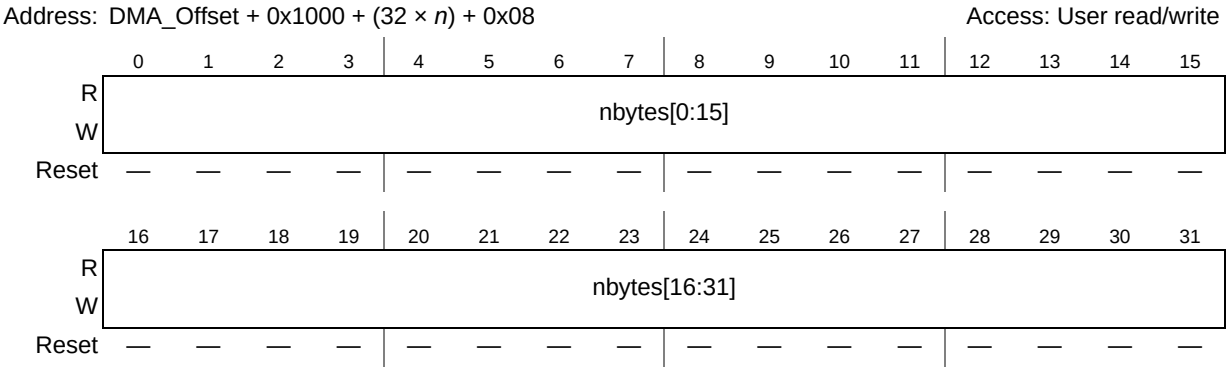


Figure 15-27. TCD*n* Word 2 (TCD*n*.nbytes) field (DMACR[EMLM] = 0)

Table 15-22. TCD*n* Word 2 (TCD*n*.nbytes) field description

Name	Description
nbytes[0:31]	Inner “minor” byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel is activated, the contents of the appropriate TCD is loaded into the DMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. After the minor count is exhausted, the current values of the saddr and daddr are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed. The nbytes value 0x0000_0000 is interpreted as 0x1_0000_0000, thus specifying a 4 GB transfer.

When minor loop mapping (DMACR[EMLM] = 1) is enabled, TCD word2 is redefined as four fields: a source minor loop offset enable, a destination minor loop offset enable, a minor loop offset field and a nbytes field.

Address: DMA_Offset + 0x1000 + (32 × *n*) + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	smloe	dmloe	mloff[0:13] or nbytes[0:13]													
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	mloff[14:19] or nbytes[14:19]								nbytes[20:29]							
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 15-28. TCD*n* Word 2 (TCD*n*.nbytes) field (DMACR[EMLM] = 1)

Table 15-23. TCD*n* Word 2 (TCDn.nbytes) field descriptions

Name	Description
smloe	Source minor loop offset enable This flag selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the saddr. 1 The minor loop offset is applied to the saddr.
dmloe	Destination minor loop offset enable This flag selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the daddr. 1 The minor loop offset is applied to the daddr.

Table 15-23. TCD*n* Word 2 (TCDn.nbytes) field descriptions (continued)

Name	Description
nbytes[0:19] or mloff[0:19]	Inner “minor” byte transfer count or Minor loop offset If both smloe and dmloe are cleared, this field is part of the byte transfer count. If either smloe or dmloe are set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop is completed.
nbytes[0:9]	Inner “minor” byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel is activated, the contents of the appropriate TCD is loaded into the DMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. Once the minor count is exhausted, the current values of the saddr and daddr are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed. This field is extended to 30 bits when both smloe and dmloe are cleared (disabled).

Figure 15-29 and Table define word 3 of the TCD*n* structure, the slast field.

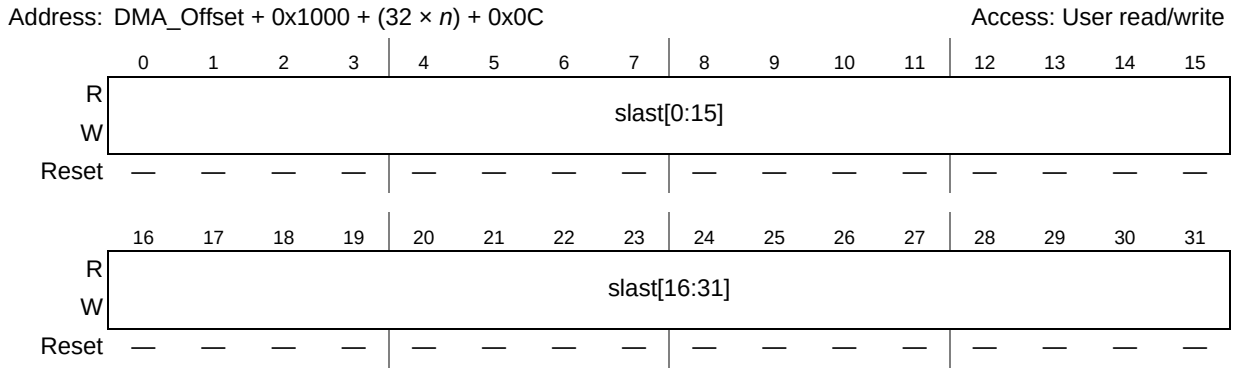


Figure 15-29. TCD*n* Word 3 (TCD*n*.slast) field

Table 15-24. TCD*n* Word 3 (TCD*n*.slast) field descriptions

Name	Description
slast[0:31]	Last source address adjustment Adjustment value added to the source address at the completion of the outer major iteration count. This value can be applied to “restore” the source address to the initial value, or adjust the address to reference the next data structure.

Figure 15-30 and Table 15-25 define word 4 of the TCD*n* structure, the daddr field.

Address: DMA_Offset + 0x1000 + (32 × *n*) + 0x10 Access: User read/write

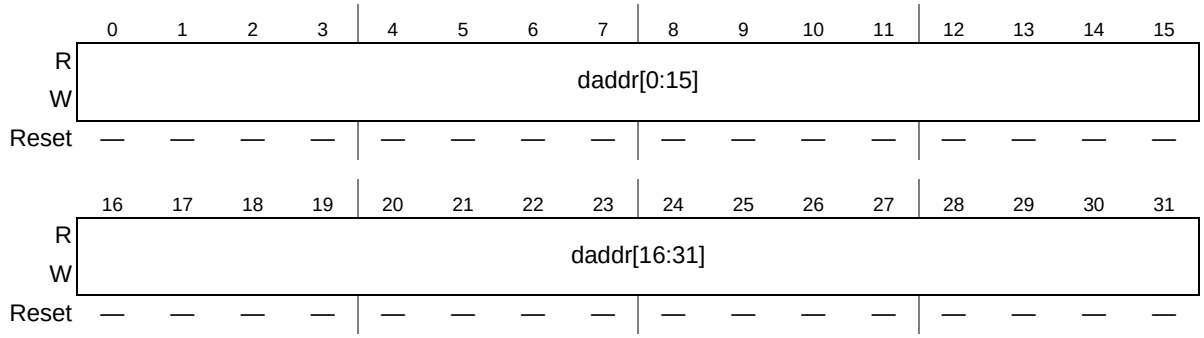


Figure 15-30. TCD*n* Word 4 (TCD*n*.daddr) field

Table 15-25. TCD*n* Word 4 (TCD*n*.daddr) field description

Name	Description
daddr[0:31]	Destination address Memory address pointing to the destination data.

Figure 15-31 and Table 15-26 define word 5 of the TCD*n* structure, the citer and doff fields.

Address: DMA_Offset + 0x1000 + (32 × *n*) + 0x14 Access: User read/write

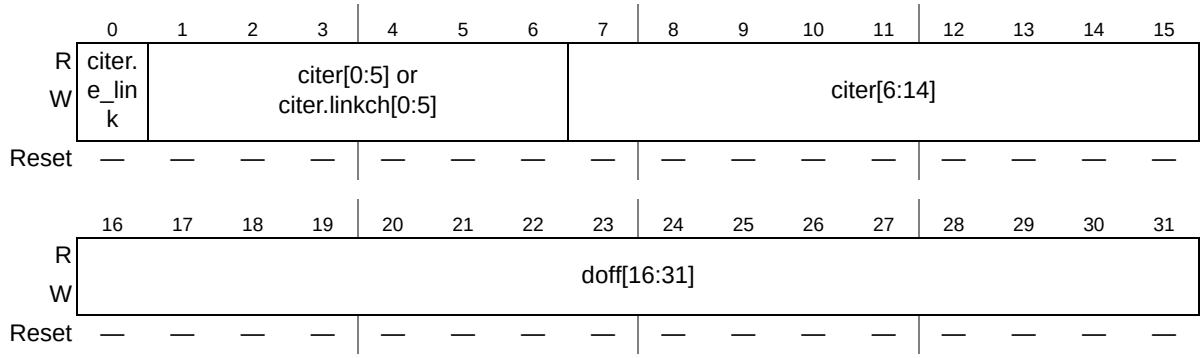


Figure 15-31. TCD*n* Word 5 (TCD*n*.{citer,doff}) fields

Table 15-26. TCD*n* Word 5 (TCD*n*.{doff,citer}) field descriptions

Name	Description
citer.e_link	Enable channel-to-channel linking on minor loop complete As the channel completes the inner minor loop, this flag enables the linking to another channel, defined by citer.linkch[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.start bit of the specified channel. If channel linking is disabled, the citer value is extended to 15 bits in place of a link channel number. If the “major” loop is exhausted, this link mechanism is suppressed in favor of the major.e_link channel linking. **Note:** This bit must be equal to the biter.e_link bit . Otherwise, a configuration error will be reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.

Table 15-26. TCD*n* Word 5 (TCD*n*.{doff,citer}) field descriptions (continued)

Name	Description

citer[0:5]
or
citer.linkch[0:5]
	Current “major” iteration count
	or
	Link channel number
	if (TCD.citer.e_link = 0) then
	No channel-to-channel linking (or chaining) is performed after the inner “minor” loop is exhausted. TCD word 5, bits [30:25] are used to form a 15 bit citer field.
	
	else
	After the “minor” loop is exhausted, the DMA engine initiates a channel service request at the channel defined by citer.linkch[5:0] by setting that channel’s TCD.start bit.
	
	The value contained in citer.linkch[5:0] must not exceed the number of implemented channels.

citer[6:14]	Current “major” iteration count This 9 or 15-bit count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. Once the major iteration count is exhausted, the channel performs a number of operations (e.g., final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the citer field from the beginning iteration count (biter) field. When the citer field is initially loaded by software, it must be set to the same value as that contained in the biter field. If the channel is configured to execute a single service request, the initial values of biter and citer should be 0x0001.
doff[16:31]	Destination address signed offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

Figure 15-32 and Table 15-27 define word 6 of the TCD*n* structure, the dlast_sga field.

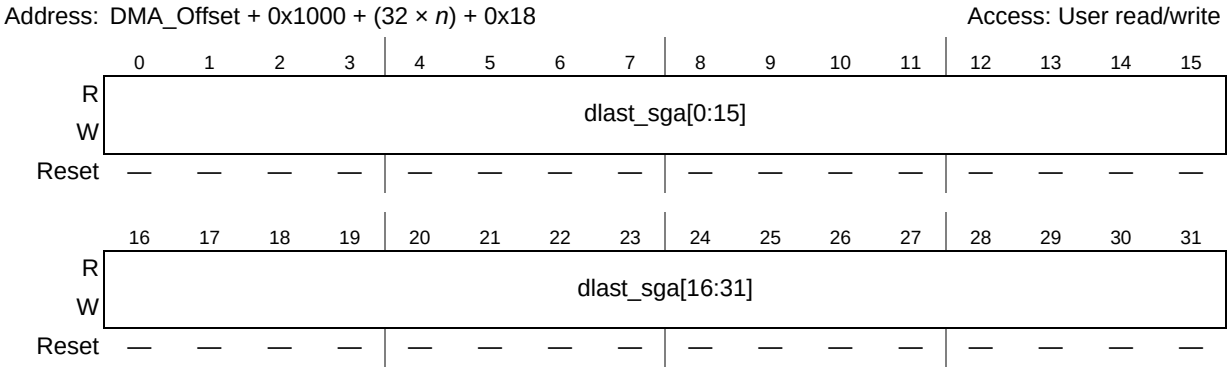


Figure 15-32. TCD*n* Word 6 (TCD*n*.dlast_sga) field

Table 15-27. TCD*n* Word 6 (TCD*n*.dlast_sga) field description

Name	Description

dlast_sga[31:00:31]	Last destination address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather)
	if (TCD.e_sg = 0) then
	Adjustment value added to the destination address at the completion of the outer major iteration count.
	
	This value can be applied to “restore” the destination address to the initial value, or adjust the address to reference the next data structure.
	
	else
	This address points to the beginning of a 0-modulo-32 region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32, else a configuration error is reported.

Figure 15-33 and Table 15-28 define word 7 of the TCD*n* structure, the biter and control/status fields.

Address: DMA_Offset + 0x1000 + (32 × *n*) + 0x1C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	bwc		major.linkch[0:5]						done	active	major. e_link	e_sg	d_req	int_ha lf	int_m aj	start
Reset	—	—	—	—	—	—	—	—	0	0	—	—	—	—	—	0

Figure 15-33. TCD*n* Word 7 (TCD*n*.{biter,control/status}) fields

Table 15-28. TCD*n* Word 7 (TCD*n*.{biter, control/status}) field descriptions

Name	Description
biter.e_link	Enable channel-to-channel linking on minor loop complete This is the initial value copied into the citer.e_link field when the major loop is completed. The citer.e_link field controls channel linking during channel execution. **Note:** This bit must be equal to the citer.e_link bit. Otherwise, a configuration error will be reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.

Table 15-28. TCD*n* Word 7 (TCD*n*.{biter, control/status}) field descriptions (continued)

Name	Description

biter[0:5]
or
biter.linkch[0:5]
	Beginning “major” iteration count
	or
	Beginning Link channel number
	This is the initial value copied into the citer field or citer.linkch field when the major loop is completed. The citer fields controls the iteration count and linking during channel execution.
	
	if (TCD.biter.e_link = 0) then
	No channel-to-channel linking (or chaining) is performed after the inner “minor” loop is exhausted. TCD word 5, bits [30:25] are used to form a 15 bit biter field.
	
	else
	After the “minor” loop is exhausted, the DMA engine initiates a channel service request at the channel defined by biter.linkch[5:0] by setting that channel’s TCD.start bit.
	
	The value contained in biter.linkch[5:0] must not exceed the number of implemented channels.

biter[6:14]	Beginning “major” iteration count This is the initial value copied into the citer field or citer.linkch field when the major loop is completed. The citer fields controls the iteration count and linking during channel execution. This 9- or 15-bit count represents the beginning major loop count for the channel. As the major iteration count is exhausted, the contents of the entire 16-bit biter entry is reloaded into the 16-bit citer entry. When the biter field is initially loaded by software, it must be set to the same value as that contained in the citer field. If the channel is configured to execute a single service request, the initial values of biter and citer should be 0x0001.
bwc[0:1]	Bandwidth control This two-bit field provides a mechanism to effectively throttle the amount of bus bandwidth consumed by the DMA. In general, as the DMA processes the inner minor loop, it continuously generates read/write, read/write, ... sequences until the minor count is exhausted. This field forces the DMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the platform’s cross-bar arbitration switch. To minimize startup latency, bandwidth control stalls are suppressed for the first two AHB bus cycles and after the last write of each minor loop. The dynamic priority elevation setting elevates the priority of the DMA as seen by the cross-bar arbitration switch for the executing channel. Dynamic priority elevation is suppressed during the first two AHB bus cycles. 00 No DMA engine stalls 01 Dynamic priority elevation 10 DMA engine stalls for 4 cycles after each r/w 11 DMA engine stalls for 8 cycles after each r/w

Table 15-28. TCD*n* Word 7 (TCD*n*.{biter, control/status}) field descriptions (continued)

Name	Description

major.linkch[0:5]	Link channel number
	if (TCD.major.e_link = 0) then
	No channel-to-channel linking (or chaining) is performed after the outer “major” loop counter is exhausted.
	
	else
	After the “major” loop counter is exhausted, the DMA engine initiates a channel service request at the channel defined by major.linkch[5:0] by setting that channel’s TCD.start bit.
	
	The value contained in major.linkch[5:0] must not exceed the number of implemented channels.

done	Channel done This flag indicates the DMA has completed the outer major loop. It is set by the DMA engine as the citer count reaches zero; it is cleared by software, or the hardware when the channel is activated. This bit must be cleared in order to write the major.e_link or e_sg bits.
active	Channel active This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the DMA engine as the inner minor loop completes or if any error condition is detected.
major.e_link	Enable channel-to-channel linking on major loop complete As the channel completes the outer major loop, this flag enables the linking to another channel, defined by major.linkch[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.start bit of the specified channel. *To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD.done bit is set.* 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
e_sg	Enable scatter/gather processing As the channel completes the outer major loop, this flag enables scatter/gather processing in the current channel. If enabled, the DMA engine uses dlast_sga as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure which is loaded as the transfer control descriptor into the local memory. *To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD.done bit is set.* 0 The current channel’s TCD is “normal” format. 1 The current channel’s TCD specifies a scatter gather format. The dlast_sga field provides a memory pointer to the next TCD to be loaded into this channel after the outer major loop completes its execution.
d_req	Disable request If this flag is set, the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero. 0 The channel’s DMAERQ bit is not affected. 1 The channel’s DMAERQ bit is cleared when the outer major loop is complete.

Table 15-28. TCD*n* Word 7 (TCD*n*.{biter, control/status}) field descriptions (continued)

Name	Description
int_half	Enable an interrupt when major counter is half complete If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the DMA engine is (citer = (biter >> 1)). This halfway point interrupt request is provided to support double-buffered schemes or other types of data movement where the processor needs an early indication of the transfer's progress. The halfway complete interrupt is disabled when biter values are less than two. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
int_maj	Enable an interrupt when major iteration count completes If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
start	Channel start If this flag is set, the channel is requesting service. The DMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

15.4 Functional description

This section provides an overview of the microarchitecture and functional operation of the DMA module.

15.4.1 DMA microarchitecture

The DMA module is partitioned into two major modules: the DMA engine and the transfer control descriptor local memory. Additionally, the DMA engine is further partitioned into four submodules, which are detailed below.

- DMA engine
  - *addr_path*: This module implements registered versions of two channel transfer control descriptors: channel "x" and channel "y," and is responsible for all the master bus address calculations. All the implemented channels provide the exact same functionality. This hardware structure allows the data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel service request is asserted while the first channel is active. Once a channel is activated, it runs until the minor loop is completed unless preempted by a higher priority channel. This capability provides a mechanism (optionally enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

    When any other channel is activated, the contents of its transfer control descriptor is read from the local memory and loaded into the registers of the other addr_path.channel_{x,y}. Once the inner minor loop completes execution, the addr_path hardware writes the new values for the

TCD*n*.{saddr, daddr, citer} back into the local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*.citer field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.

— *data_path*: This module implements the actual bus master read/write datapath. It includes 32 bytes of register storage (matching the maximum transfer size) and the necessary mux logic to support any required data alignment. The AMBA-AHB read data bus is the primary input, and the AHB write data bus is the primary output.

The addr_ and data_path modules directly support the 2-stage pipelined AMBA-AHB bus. The addr_path module represents the first stage of the bus pipeline (the address phase), while the data_path module implements the second stage of the pipeline (the data phase).

— *pmodel_charb*: This module implements the first section of DMA's programming model as well as the channel arbitration logic. The programming model registers are connected to the IPS bus (not shown). The ipd_req[n] inputs and dma_ipi_int[n] outputs are also connected to this module (via the control logic).

— *control*: This module provides all the control functions for the DMA engine. For data transfers where the source and destination sizes are equal, the DMA engine performs a series of source read, destination write operations until the number of bytes specified in the inner 'minor loop' byte count has been moved. For descriptors where the sizes are not equal, multiple access of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- transfer_control_descriptor local memory
  — *memory controller*: This logic implements the required dual-ported controller, handling accesses from both the DMA engine as well as references from the IPS bus. As noted earlier, in the event of simultaneous accesses, the DMA engine is given priority and the IPS transaction is stalled. The hooks to a BIST controller for the local TCD memory are included in this module.
  — *memory array*: The TCD is implemented using a single-ported, synchronous compiled RAM memory array

15.4.2 DMA basic data flow

The basic flow of a data transfer can be partitioned into three segments. As shown in Figure 15-34, the first segment involves the channel service request. In the diagram, this example uses the assertion of the ipd_req[n] signal to request service for channel n. Channel service request via software and the TCD*n*.start bit follows the same basic flow as an ipd_req. The ipd_req[n] input signal is registered internally and then routed to through the DMA engine, first through the control module, then into the programming model/channel arbitration (pmodel_charb) module. In the next cycle, the channel arbitration is performed, either using the fixed-priority or round-robin algorithm. After the arbitration is complete, the activated channel number is sent through the address path (addr_path) and converted into the required address to access the TCD local memory. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the dma_engine.addr_path.channel_{x,y} registers. The TCD memory

is organized 64-bits in width to minimize the time needed to fetch the activated channel's descriptor and load it into the dma_engine.addr_path.channel_{x,y} registers.

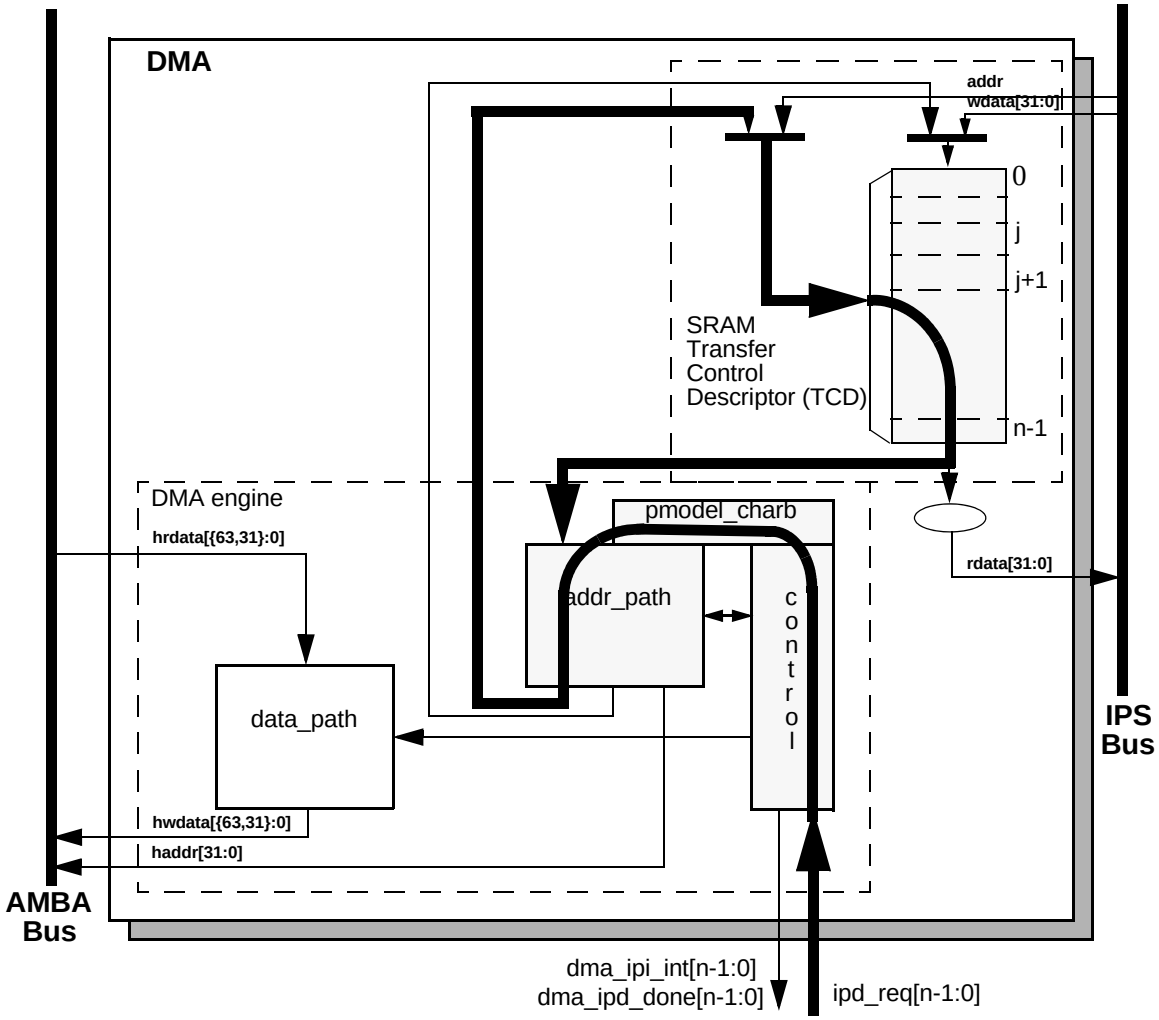


Figure 15-34. DMA operation, part 1

In the second part of the basic data flow as shown in Figure 15-35, the modules associated with the data transfer (addr_path, data_path and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data_path module until it is gated onto the AMBA-AHB bus during the destination write. This source read/destination write processing continues until the inner minor byte count has been transferred. The dma_ipd_done[n] signal is asserted at the end of the minor byte count transfer.

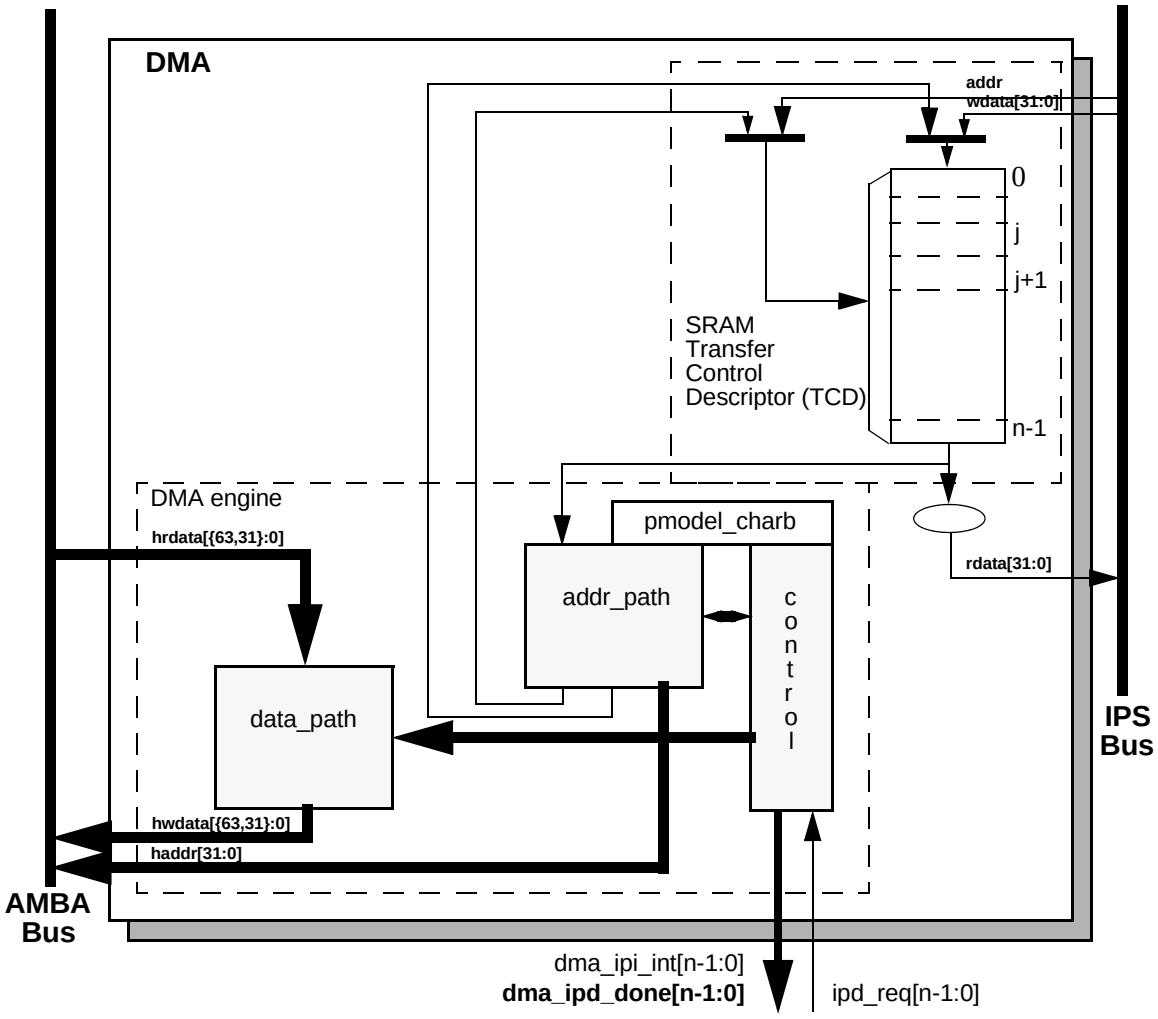


Figure 15-35. DMA operation, part 2

Once the inner minor byte count has been moved, the final phase of the basic data flow is performed. In this segment, the addr_path logic performs the required updates to certain fields in the channel's TCD, e.g., saddr, daddr, citer. If the outer major iteration count is exhausted, then there are additional operations which are performed. These include the final address adjustments and reloading of the biter field into the citer. Additionally, assertion of an optional interrupt request occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in Figure 15-36.

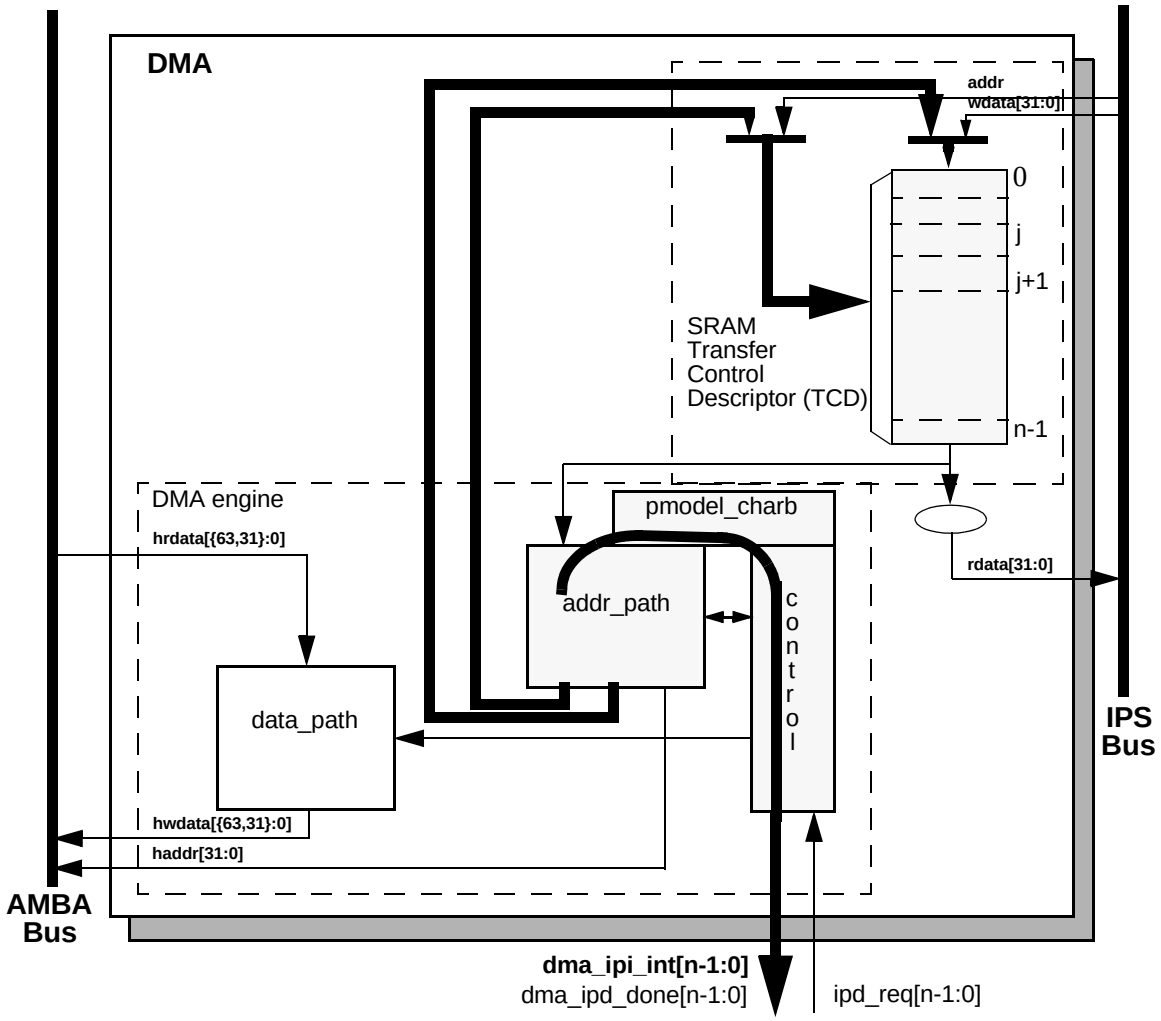


Figure 15-36. DMA operation, part 3

15.4.3 DMA performance

This section addresses the performance of the DMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the DMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests which can be serviced in a fixed time is a more interesting metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the DMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in Table 15-29. The following assumptions apply to Table 15-29 and Table 15-30:

- Platform SRAM can be accessed with zero wait-states when viewed from the AMBA-AHB data phase

- All IPS reads require two wait-states, and IPS writes three wait-states, again viewed from the system bus data phase
- All IPS accesses are 32 bits in size

Table 15-29 presents a peak transfer rate comparison, measured in megabytes per second. In this table, the Platform_SRAM-to-Platform_SRAM transfers occur at the native platform datapath width, that is, either 32- or 64-bits per access. For all transfers involving the IPS bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

Table 15-29. DMA peak transfer rates [MB/s]

Platform Speed, Width	Platform SRAM-to-Platform SRAM	32-bit IPS-to-Platform SRAM	Platform SRAM-to-32-bit IPS
66.7 MHz, 32-bit	133.3	66.7	53.3
66.7 MHz, 64-bit	266.7	66.6	53.3
83.3 MHz, 32-bit	166.7	83.3	66.7
83.3 MHz, 64-bit	333.3	83.3	66.7
100.0 MHz, 32-bit	200.0	100.0	80.0
100.0 MHz, 64-bit	400.0	100.0	80.0
133.3 MHz, 32-bit	266.7	133.3	106.7
133.3 MHz, 64-bit	533.3	133.3	106.7
150.0 MHz, 32-bit	300.0	150.0	120.0
150.0 MHz, 64-bit	600.0	150.0	120.0

The second performance metric is a measure of the number of DMA requests which can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single IPS-mapped operand to/from the platform SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The DMA design supports the following hardware service request sequence:

- Cycle 1: ipd_req[n] is asserted
- Cycle 2: The ipd_req[n] is registered locally in the DMA module and qualified (TCD.start bit initiated requests start at this point with the registering of the IPS write to TCD word7)
- Cycle 3: Channel arbitration begins
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated.
- Cycle 5–6: The first two parts of the activated channel's TCD is read from the local memory. The memory width to the DMA engine is 64 bits, so the entire descriptor can be accessed in four cycles.
- Cycle 7: The first AMBA-AHB read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the platform's crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
- Cycle 8–?: The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.
  The exact timing from this point is a function of the response times for the channel's read and write accesses. In this case of an IPS read and a platform SRAM write, the combined data phase time is 4 cycles. For an SRAM read and IPS write, it is 5 cycles.

- Cycle ?+1: This cycle represents the data phase of the last destination write
- Cycle ?+2: The DMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. TCD word7 is read and checked for channel linking or scatter/gather requests.
- Cycle ?+3: The appropriate fields in the first part of the TCD*n* are written back into the local memory
- Cycle ?+4: The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
- Cycle ?+5: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the AHB system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with IPS-to-SRAM (4 cycles) and SRAM-to-IPS (5 cycles), DMA requests can be processed every 11.5 cycles (4 + (4+5)÷2 + 3). This is the time from Cycle 4 to Cycle "?+5". The resulting peak request rate, as a function of the platform frequency, is shown in Table 15-30. This metric represents millions of requests per second.

Table 15-30. DMA peak request rate [MReq/sec]

Platform Speed	Request Rate (zero wait state)	Request Rate (with wait states)
66.6 MHz	7.4	5.8
83.3 MHz	9.2	7.2
100.0 MHz	11.1	8.7
133.3 MHz	14.8	11.6
150.0 MHz	16.6	13.0

A general formula to compute the peak request rate (with overlapping requests) is:

$$PEAKreq = freq \div [\ entry + (1 + read\_ws) + (1 + write\_ws) + exit\ ] \qquad \textit{Eqn. 15-1}$$

where:

PEAKreq—peak request rate

freq—platform frequency

entry—channel startup (4 cycles)

read_ws—wait states seen during the system bus read data phase

write_ws—wait states seen during the system bus write data phase

exit—channel shutdown (3 cycles)

For example: consider a platform with the following characteristics:

- Platform SRAM can be accessed with one wait-state when viewed from the AMBA-AHB data phase
- All IPS reads require two wait-states, and IPS writes three wait-states, again viewed from the system bus data phase
- Platform operates at 150 MHz

For an SRAM to IPS transfer:

$$\text{PEAKreq} = 150\ \text{MHz} \div [\ 4 + (1 + 1) + (1 + 3) + 3\ ]\ \text{cycles} = 11.5\ \text{Mreq/sec} \qquad \textit{Eqn. 15-2}$$

For an IPS to SRAM transfer:

$$\text{PEAKreq} = 150\ \text{MHz} \div [\ 4 + (1 + 2) + (1 + 1) + 3\ ]\ \text{cycles} = 12.5\ \text{Mreq/sec} \qquad \textit{Eqn. 15-3}$$

Assuming an even distribution of the two transfer types, the average Peak Request Rate would be:

$$\text{PEAKreq} = (11.5\ \text{Mreq/sec} + 12.5\ \text{Mreq/sec}) \div 2 = 12.0\ \text{Mreq/sec} \qquad \textit{Eqn. 15-4}$$

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start (where no channel is executing, DMA is idle) are:

- 11 cycles for a software (TCD.start bit) request
- 12 cycles for a hardware (ipd_req signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the ipd_req signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlap the previous executing channel.

NOTE

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

15.5 Initialization/application information

15.5.1 DMA initialization

A typical initialization of the DMA is:

1. Write the DMACR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the DMAEEI registers if so desired.
4. Write the 32 byte TCD for each channel that may request service.
5. Enable any hardware service requests via the DMAERQ register.
6. Request channel service by either software (setting the TCD.start bit) or by hardware (slave device asserting its ipd_req signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The DMA engine will read the entire TCD for the selected channel into its internal address path module. As the TCD is being read, the first transfer is initiated on the AHB bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD.saddr) to the destination (as defined by the destination address, TCD.daddr) continue until the specified number of bytes (TCD.nbytes) have been transferred. When the transfer is complete, the DMA engine's local TCD.saddr, TCD.daddr, and TCD.citer are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing is executed; that is, interrupts, major loop channel linking, and scatter/gather operations, if enabled.

15.5.2 DMA programming errors

The DMA performs various tests on the Transfer Control Descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of two errors; Group Priority Error and Channel Priority Error, GPE and CPE in the DMAES register respectively.

For all error types other than Group or Channel Priority Errors, the channel number causing the error is recorded in the DMAES register. If the error source is not removed before the next activation of the problem channel, the error will be detected and recorded again.

The sequence listed below is correct. For item 2, the dma_ipd_ack{done} lines will assert only if the selected channel is requesting service via the ipd_req signal. I think the typical application will enable error interrupts for all channels. So the user will get an error interrupt, but the channel number for the DMAERR register and the error interrupt request line may be wrong because they reflect the selected channel.

Channel priority errors are identified within a group after that group has been selected as the active group. For example:

1. The DMA is configured for fixed group and fixed channel arbitration modes.
2. Group3 is the highest priority and all channels are unique in that group.
3. Group2 is the next highest priority and has two channels with the same priority level.
4. If Group3 has any service requests, those requests will be executed.
5. Once all of Group3 requests have completed, Group2 will be the next active group.
6. If Group2 has a service request, then an undefined channel in Group2 will be selected and a channel priority error will occur.
7. This will repeat until the all of Group2 requests have been removed or a higher priority Group3 request comes in.

A group priority error is global and any request in any group will cause a group priority error.

In general, if priority levels are not unique, the highest (channel/group) priority that has an active request will be selected, but the lowest numbered (channel/group) with that priority will be selected by arbitration and executed by the DMA engine. The hardware service request handshake signals, error interrupts and error reporting will be associated with the selected channel.

15.5.3 DMA arbitration mode considerations

15.5.3.1 Fixed group arbitration, fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group will be selected to execute. If the DMA is programmed so the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the DMA controller — no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request.

The advantage of this scenario is that latency can be small for channels that need to be serviced quickly.

Preemption is available in this scenario only.

15.5.3.2 Round-robin group arbitration, fixed channel arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with an service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round robin algorithm will select the highest pending request from the next group in the round robin sequence. Servicing continues round robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced.

The advantage of this scenario is that no one group uses all the DMA bandwidth.

The highest priority channel selection latency is potentially greater than fixed/fixed arbitration.

Excessive request rates on high-priority channels can prevent the servicing of lower priority channels in the same group.

15.5.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in Section 15.5.3.2, Round-robin group arbitration, fixed channel arbitration, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round robin manner, any channel that generates DMA requests faster than a combination of the group round robin service rate and the channel service rate for its group will not

prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high.

All channels are treated equally. Priority levels are not used in round robin/round robin mode.

15.5.3.4 Fixed group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Section 15.5.3.1, Fixed group arbitration, fixed channel arbitration, but all the channels in the highest priority group will be serviced.

Service latency will be short on the highest priority group, but can become much longer as the group priority decreases.

15.5.4 DMA transfer

15.5.4.1 Single request

To perform a simply transfer of ‘n’ bytes of data with one activation, set the major loop to one (TCD.citer = TCD.biter = 1). The data transfer will begin after the channel service request is acknowledged and the channel is selected to execute. Once the transfer is complete, the TCD.done bit will be set and an interrupt will be generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The DMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

TCD.citer = TCD.biter = 1

TCD.nbytes = 16

TCD.saddr = 0x1000

TCD.soff = 1

TCD.ssize = 0

TCD.slast = -16

TCD.daddr = 0x2000

TCD.doff = 4

TCD.dsize = 2

TCD.dlast_sga= -16

TCD.int_maj = 1

TCD.start = 1 (TCD.word7 should be written last after all other fields have been initialized)

All other TCD fields = 0

This generates the following sequence of events:

1. IPS write to the TCD.start bit requests channel service
2. The channel is selected by arbitration for servicing
3. DMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
4. DMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
   a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b. write_word(0x2000) → *first iteration of the minor loop*
   c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
   d. write_word(0x2004) → *second iteration of the minor loop*
   e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
   f. write_word(0x2008) → *third iteration of the minor loop*
   g. read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
   h. write_word(0x200c) → *last iteration of the minor loop* → *major loop complete*
6. DMA engine writes: TCD.saddr = 0x1000, TCD.daddr = 0x2000, TCD.citer = 1 (TCD.biter)
7. DMA engine writes: TCD.active = 0, TCD.done = 1, DMAINT[n] = 1
8. The channel retires

The DMA goes idle or services next channel.

15.5.4.2 Multiple requests

The next example is the same as the previous example, with the exception of transferring 32 bytes via two hardware requests. The only fields that change are the major loop iteration count and the final address offsets. The DMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests is enabled in the DMAERQ register, channel service requests are initiated by the slave device.

TCD.citer = TCD.biter .

TCD.slast = -32

TCD.dlast_sga = -32

This would generate the following sequence of events:

1. First hardware (ipd_req) request for channel service
2. The channel is selected by arbitration for servicing
3. DMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
4. DMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
   a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b. write_word(0x2000) → *first iteration of the minor loop*
   c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
   d. write_word(0x2004) → *second iteration of the minor loop*
   e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
   f. write_word(0x2008) → *third iteration of the minor loop*
   g. read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
   h. write_word(0x200c) → *last iteration of the minor loop*
6. DMA engine writes: TCD.saddr = 0x1010, TCD.daddr = 0x2010, TCD.citer = 1
7. DMA engine writes: TCD.active = 0
8. The channel retires → *one iteration of the major loop*

The DMA goes idle or services next channel.

9. Second hardware (ipd_req) requests channel service
10. The channel is selected by arbitration for servicing
11. DMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
12. DMA engine reads: channel TCD data from local memory to internal register file
13. The source to destination transfers are executed as follows:
    a. read_byte(0x1010), read_byte(0x1011), read_byte(0x1012), read_byte(0x1013)
    b. write_word(0x2010) → *first iteration of the minor loop*
    c. read_byte(0x1014), read_byte(0x1015), read_byte(0x1016), read_byte(0x1017)
    d. write_word(0x2014) → *second iteration of the minor loop*
    e. read_byte(0x1018), read_byte(0x1019), read_byte(0x101a), read_byte(0x101b)
    f. write_word(0x2018) → *third iteration of the minor loop*
    g. read_byte(0x101c), read_byte(0x101d), read_byte(0x101e), read_byte(0x101f)
    h. write_word(0x201c) → *last iteration of the minor loop* → *major loop complete*

14. DMA engine writes: TCD.saddr = 0x1000, TCD.daddr = 0x2000, TCD.citer = 2 (TCD.biter)
15. DMA engine writes: TCD.active = 0, TCD.done = 1, DMAINT[n] = 1
16. The channel retires → *major loop complete*

The DMA goes idle or services the next channel.

15.5.5 TCD status

15.5.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first method is to read the TCD.citer field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD.start bit AND the TCD.active bit. The minor loop complete condition is indicated by both bits reading zero after the TCD.start was written to a one. Polling the TCD.active bit may be inconclusive because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

1. TCD.start = 1, TCD.active = 0, TCD.done = 0 (channel service request via software)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle)
   or
   TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

The best method to test for minor loop completion when using hardware initiated service requests is to read the TCD.citer field and test for a change. The hardware request and acknowledge handshakes signals are not visible in the programmer’s model.

The TCD status bits execute the following sequence for a hardware activated channel:

1. ipd_req asserts (channel service request via hardware)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle)
   or
   TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

For both activation types, the major loop complete status is explicitly indicated via the TCD.done bit.

The TCD.start bit is cleared automatically when the channel begins execution regardless of how the channel was activated.

15.5.5.2 Active channel TCD reads

The DMA will read back the 'true' TCD.saddr, TCD.daddr, and TCD.nbytes values if read while a channel is executing. The 'true' values of the saddr, daddr, and nbytes are the values the DMA engine is currently using in its internal register file and not the values in the TCD local memory for that channel. The

addresses (saddr and daddr) and nbytes (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

15.5.5.3 Preemption status

Preemption is only available when *fixed* arbitration is selected for *both* group and channel arbitration modes. A preempt-able situation is one in which a preempt-enabled channel is running and a higher priority request becomes active. When the DMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the relative priority of the actively running and the outstanding requests become undefined. Channel and/or group priorities are treated as equal (constantly rotating) when round-robin arbitration mode is selected.

The TCD.active bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one iteration of the major loop. Two TCD.active bits set at the same time in the overall TCD map indicates a higher priority channel is actively preempting a lower priority channel.

The worst case latency when switching to a preempt channel is the summation of:

- arbitration latency (2 cycles)
- bandwidth control stalls (if enabled)
- the time to execute two read/write sequences (including AHB bus holds; a system dependency driven by the slave devices or the crossbar)

15.5.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD.start bit of another channel (or itself) thus initiating a service request for that channel. This operation is automatically performed by the DMA engine at the conclusion of the major or minor loop when properly enabled.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD.citer.e_link field are used to determine whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, with the initial fields of:

TCD.citer.e_link = 1

TCD.citer.linkch = 0xC

TCD.citer value = 0x4

TCD.major.e_link = 1

TCD.major.linkch = 0x7

will execute as:

1. minor loop done → set channel 12 TCD.start bit
2. minor loop done → set channel 12 TCD.start bit

3. minor loop done → set channel 12 TCD.start bit
4. minor loop done, major loop done → set channel 7 TCD.start bit

When minor loop linking is enabled (TCD.citer.e_link = 1), the TCD.citer field uses a nine bit vector to form the current iteration count.

When minor loop linking is disabled (TCD.citer.e_link = 0), the TCD.citer field uses a 15 bit vector to form the current iteration count. The bits associated with the TCD.citer.linkch field are concatenated onto the citer value to increase the range of the citer.

NOTE

The TCD.citer.e_link bit and the TCD.biter.e_link bit must equal or a configuration error will be reported. The citer and biter vector widths must be equal to calculate the major loop, half-way done interrupt point.

15.5.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

15.5.7.1 Dynamic priority changing

The following two options are recommended for dynamically changing ***channel*** priority levels:

1. Switch to round-robin channel arbitration mode, change the channel priorities, then switch back to fixed arbitration mode.
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing ***group*** priority levels:

1. Switch to round-robin group arbitration mode, change the group priorities, then switch back to fixed arbitration mode.
2. Disable ALL channels, change the group priorities, then enable the appropriate channels.

15.5.7.2 Dynamic channel linking and dynamic scatter/gather

Dynamic channel linking and dynamic scatter/gather is the process of changing the TCD.major.e_link or TCD.e_sg bits during channel execution. These bits are read from the TCD local memory at the ***end*** of channel execution thus allowing the user to enable either feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the DMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer’s model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link or dynamic scatter/gather request:

1. Set the TCD.major.e_link bit.

2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
   a. If the bit is set, the dynamic link attempt was successful.
   b. If the bit is cleared, the attempted dynamic link did not succeed, the channel was already retiring.

This same coherency model is true for dynamic scatter/gather operations. For both dynamic requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the DMA engine after a channel begins execution.

15.5.8 Hardware request release timing

This section provides a timing diagram for deasserting the ipd_req hardware request signal. Figure 15-37 shows two read write sequences with grey indicating the release of the ipd_req hardware request signal.

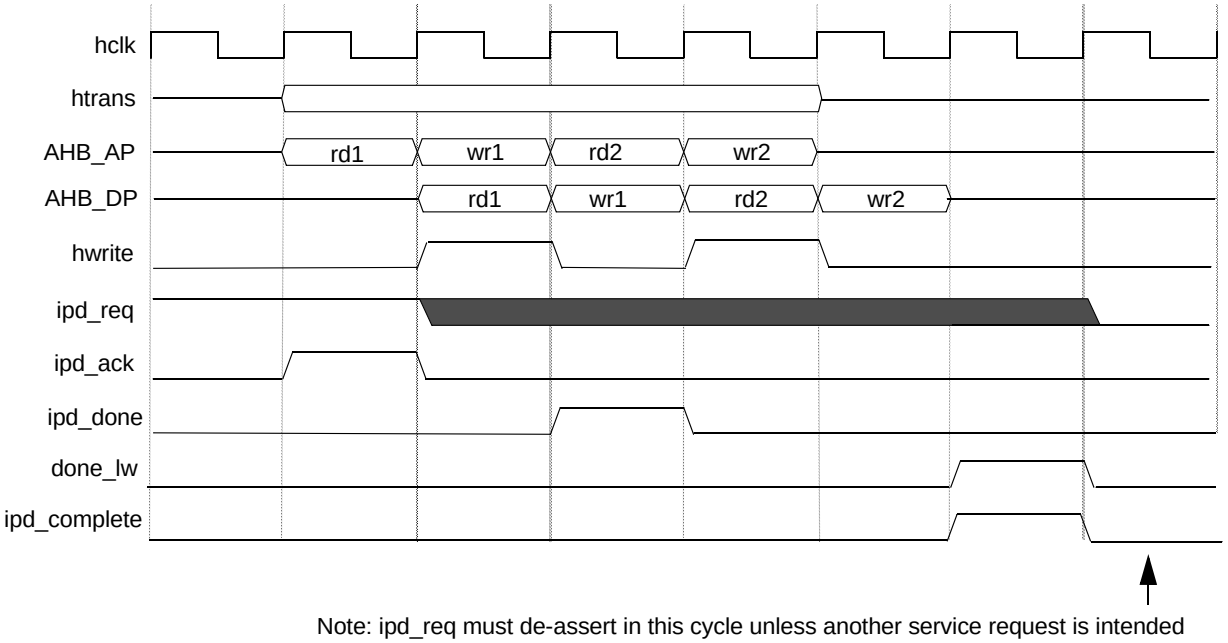


Figure 15-37. ipd_req hardware handshake

Chapter 16
Error Correction Status Module (ECSM)

16.1 Introduction

The Error Correction Status Module (ECSM) provides a myriad of miscellaneous control functions for the device including program-visible information about configuration and revision levels, a reset status register, wakeup control for exiting low-power modes, and optional features such as information on memory errors reported by error-correcting codes.

The register protection module (see Appendix A, Registers Under Protection) provides access protection for slave modules INTC, ECSM, MPU, STM, and SWT.

16.2 Overview

The Error Correction Status Module is mapped into the IPS space and supports a number of miscellaneous control functions for the device.

16.3 Features

The ECSM includes these features:

- Program-visible information on the device configuration and revision
- Optional registers for capturing information on memory errors if error-correcting codes (ECC) are implemented
- Optional registers to specify the generation of single- and double-bit memory data inversions for test purposes if error-correcting codes are implemented
- XBAR_lite priority functions, including forcing round robin and high-priority enabling
- Spp_ips_reg_protection provides privileged-only access to selected on-platform slave devices: INTC, ECSM, MPU, STM, and SWT

16.4 Memory map and register description

This section details the programming model for the Error Correction Status Module. This is a 128-byte space mapped to the region serviced by an IPS bus controller.

16.4.1 Memory map

The Error Correction Status Module does not include any logic which provides access control. Rather, this function is supported using the standard access control logic provided by the IPS controller.

Table 16-1 is a 32-bit view of the ECSM memory map.

Table 16-1. ECSM memory map

ECSM Offset	Register			
0x00	Processor Core Type (PCT)	Revision (REV)	Reserved	
0x08	Reserved			
0x0C	Reserved			Misc. Reset Status (MRSR)
0x10	Reserved			Misc. Wakeup Control (MWCR)
0x14	Reserved			
0x18	Reserved			
0x1C	Reserved			Misc. Interrupt (MIR)
0x20	Reserved			
0x24	Miscellaneous User-Defined Control Register (MUDCR)			
0x28	Reserved			
0x2C–0x3C	Reserved			
0x40	Reserved			ECC Configuration (ECR)
0x44	Reserved			ECC Status (ESR)
0x48	Reserved		ECC Error Generation (EEGR)	
0x4c	Reserved			
0x50	Flash ECC Address (FEAR)			
0x54	Reserved		Flash ECC Master (FEMR)	Flash ECC Attributes (FEAT)
0x58	Reserved			
0x5c	Flash ECC Data (FEDR)			
0x60	RAM ECC Address (REAR)			
0x64	Reserved	RAM ECC Syndrome (RESR)	RAM ECC Master (REMR)	RAM ECC Attributes (REAT)
0x68	Reserved			
0x6C	RAM ECC Data (REDR)			
0x70–0x7C	Reserved			

16.4.2 Register description

Attempted accesses to reserved addresses result in an error termination, while attempted writes to read-only registers are ignored and do not terminate with an error. Unless noted otherwise, writes to the

programming model must match the size of the register, for example, an *n*-bit register only supports *n*-bit writes, etc. Attempted writes of a different size than the register width produce an error termination of the bus cycle and no change to the targeted register.

16.4.2.1 Processor Core Type (PCT) register

The PCT is a 16-bit read-only register specifying the architecture of the processor core in the device. The state of this register is defined by a module input signal; it can only be read from the IPS programming model. Any attempted write is ignored.

See Figure 16-1 and Table 16-2 for the Processor Core Type definition.

Address: Base + 0x0000 Access: User read-only

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PCT[0:15]															
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0

Figure 16-1. Processor Core Type (PCT) register

Table 16-2. Processor Core Type (PCT) field descriptions

Name	Description
0-15 PCT[0:15]	Processor Core Type 0xE012 identifies the e200z0h Power Architecture processor core.

16.4.2.2 Revision (REV) register

The REV is a 16-bit read-only register specifying a revision number that can only be read from the IPS programming model. Any attempted write is ignored.

See Figure 16-2 and Table 16-3 for the Revision definition.

Address: Base + 0x0002 Access: User read-only

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REV[0:15]															
W																
Reset	REV[0:15]															

Figure 16-2. Revision (REV) register

Table 16-3. Revision (REV) field descriptions

Name	Description
0-15 REV[0:15]	Revision The REV[0:15] field is specified by an input signal to define a software-visible revision number.

16.4.2.3 Miscellaneous Reset Status Register (MRSR)

The MRSR contains a bit for each of the reset sources to the device. An asserted bit indicates the last type of reset that occurred. Only one bit is set at any time in the MRSR, reflecting the cause of the most recent

reset as signaled by device reset input signals. The MRSR can only be read from the IPS programming model. Any attempted write is ignored.

See Figure 16-3 and Table 16-4 for the Miscellaneous Reset Status Register definition.

Address: Base + 0x000F Access: User read-only

	0	1	2	3	4	5	6	7
R	POR	OFPLR	0	0	0	0	0	0
W								
Reset	0	1	0	0	0	0	0	0

Figure 16-3. Miscellaneous Reset Status Register (MRSR)

Table 16-4. Miscellaneous Reset Status (MRSR) field descriptions

Name	Description
0 POR	Power-On Reset 1 Last recorded event was caused by a power-on reset (based on a device input signal)
1 DIR	**Device Input Reset** 1 Last recorded event was a reset caused by a device input reset

16.4.2.4 Miscellaneous Wakeup Control Register (MWCR)

Implementation of low-power modes and exit from these modes via an interrupt requires communication between the ECSM, the interrupt controller, and external logic typically associated with phase-locked loop clock generation circuitry. The Miscellaneous Wakeup Control Register (MWCR) provides an 8-bit register controlling entry into these types of low-power modes as well as definition of the interrupt level needed to exit the mode.

The following sequence of operations is generally needed to enable this functionality. Note that the exact details are likely to be system-specific.

1. The processor core loads the appropriate data value into the MWCR, setting the ENBWCR bit and the desired interrupt priority level.
2. At the appropriate time, the processor ceases execution. The exact mechanism varies by processor core. In some cases, a processor-is-stopped status is signaled to the ECSM and external logic. This assertion, if properly enabled by MWCR[ENBWCR], causes the ECSM output signal enter_low_power_mode to be set. This, in turn, causes the selected external, low-power mode, to be entered, and the appropriate clock signals disabled. In most implementations, there are multiple low-power modes, where the exact clocks to be disabled vary across the different modes.
3. After entering the low-power mode, the interrupt controller enables a special combinational logic path which evaluates all unmasked interrupt requests. The device remains in this mode until an event which generates an unmasked interrupt request with a priority level greater than the value programmed in the MWCR[PRILVL] occurs.
4. Once the appropriately high interrupt request level arrives, the interrupt controller signals its presence, and the ECSM responds by asserting an exit_low_power_mode signal.
5. The external logic senses the assertion of the exit signal, and re-enables the appropriate clock signals.

6. With the processor core clocks enabled, the core handles the pending interrupt request.

See Figure 16-4 and Table 16-5 for the Miscellaneous Wakeup Control Register definition.

Address: Base + 0x0013 Access: User read/write

	0	1	2	3	4	5	6	7
R	ENBWCR	0	0	0	PRILVL[0:3]			
W								
Reset	0	1	0	0	0	0	0	0

Figure 16-4. Miscellaneous Wakeup Control (MWCR) Register

Table 16-5. Wakeup Control (MWCR) field descriptions

Name	Description
0 ENBWCR	Enable WCR 0 MWCR is disabled. 1 MWCR is enabled.
4-7 PRILVL[0:3]	Interrupt Priority Level The interrupt priority level is a core-specific definition. It specifies the interrupt priority level needed to exit the low-power mode. Specifically, an unmasked interrupt request of a priority level greater than the PRILVL value is required to exit the mode. Certain interrupt controller implementations include logic associated with this priority level that restricts the data value contained in this field to a [0, maximum – 1] range. See the specific interrupt controller module for details.

16.4.2.5 Miscellaneous Interrupt Register (MIR)

All interrupt requests associated with ECSM are collected in the MIR register. This includes the processor core system bus fault interrupt.

During the appropriate interrupt service routine handling these requests, the interrupt source contained in the MIR must be explicitly cleared. See Figure 16-5 and Table 16-6.

Address: Base + 0x001F Access: User read/write

	0	1	2	3	4	5	6	7
R	FB0AI	FB0SI	FB1AI	FB1SI	0	0	0	0
W	w1c	w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0

Figure 16-5. Miscellaneous Interrupt Register (MIR)

Table 16-6. Miscellaneous Interrupt (MIR) field descriptions

Name	Description
0 FB0AI	Flash Bank 0 Abort Interrupt 0 A flash memory bank 0 abort has not occurred. 1 A flash memory bank 0 abort has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.
1 FB0SI	Flash Bank 0 Stall Interrupt 0 A flash memory bank 0 stall has not occurred. 1 A flash memory bank 0 stall has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.
2 FB1AI	Flash Bank 1 Abort Interrupt 0 A flash memory bank 1 abort has not occurred. 1 A flash memory bank 1 abort has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.
3 FB1SI	Flash Bank 1 Stall Interrupt 0 A flash memory bank 1 stall has not occurred. 1 A flash memory bank 1 stall has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.

16.4.2.6 Miscellaneous User-Defined Control Register (MUDCR)

The MUDCR provides a program-visible register for user-defined control functions. It typically is used as configuration control for miscellaneous device-level modules. The contents of this register are output from the ECSM to other modules where the user-defined control functions are implemented. See Figure 16-6 and Table 16-7 for the Miscellaneous User-Defined Control Register definition.

Address: Base + 0x0024 Access: User read/write

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

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

Figure 16-6. Miscellaneous User-Defined Control Register (MUDCR)

Table 16-7. Miscellaneous User-Defined Control Register (MUDCR) field descriptions

Name	Description
MUDCR	User-Defined Control Register 0 The control associated with this MUDCR bit is disabled. 1 The control associated with this MUDCR bit is enabled.

16.4.2.6.1 XBAR_lite force_round_robin bit (MUDCR[31])

When the XBAR_lite is included on the platform, this bit is used to drive the force_round_robin bit of the XBAR_lite. This will force the slaves into round robin mode of arbitration rather than fixed mode. However, the situation is different if a master is using priority elevation, which forces the design back into fixed mode regardless of this bit. By defining the define ENABLE_ROUND_ROBIN_RESET, this bit will reset to 1.

16.4.2.7 ECC registers

For designs including error-correcting code (ECC) implementations to improve the quality and reliability of memories, there are a number of program-visible registers for the sole purpose of reporting and logging memory failures. These optional registers include:

- ECC Configuration Register (ECR)
- ECC Status Register (ESR)
- ECC Error Generation Register (EEGR)
- Flash ECC Address Register (FEAR)
- Flash ECC Master Number Register (FEMR)
- Flash ECC Attributes Register (FEAT)
- Flash ECC Data Register (FEDR)
- RAM ECC Address Register (REAR)
- RAM ECC Syndrome Register (RESR)
- RAM ECC Master Number Register (REMR)
- RAM ECC Attributes Register (REAT)
- RAM ECC Data Register (REDR)

The details on the ECC registers are provided in the subsequent sections. If the design does not include ECC on the memories, these addresses are reserved locations within the ECSM programming model.

16.4.2.8 ECC Configuration Register (ECR)

The ECC Configuration Register is an 8-bit control register for specifying which types of memory errors are reported. In all systems with ECC, the occurrence of a non-correctable error causes the current access to be terminated with an error condition. In many cases, this error termination is reported directly by the initiating bus master.

However, there are certain situations where the occurrence of this type of non-correctable error is not reported by the master. Examples include speculative instruction fetches which are discarded due to a change-of-flow operation, and buffered operand writes. The ECC reporting logic in the ECSM provides an optional error interrupt mechanism to signal all non-correctable memory errors. In addition to the interrupt generation, the ECSM captures specific information (memory address, attributes and data, bus master number, etc.) which may be useful for subsequent failure analysis.

The reporting of single-bit memory corrections can only be enabled via an SoC-configurable module input signal. While not directly accessible by a user, this capability is viewed as important for error logging and failure analysis.

See Figure 16-7 and Table 16-8 for the ECC Configuration Register definition.

Address: Base + 0x0043 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	ER1BR	EF1BR	0	0	ERNCR	EFNCR
W								
Reset	0	0	0	0	0	0	0	0

Figure 16-7. ECC Configuration Register (ECR)

Table 16-8. ECC Configuration (ECR) field descriptions

Name	Description
2 ER1BR	Enable RAM 1-bit Reporting 0 Reporting of single-bit RAM corrections is disabled. 1 Reporting of single-bit RAM corrections is enabled. This bit can only be set if the SoC-configurable input enable signal is asserted. The occurrence of a single-bit RAM correction generates an ECSM ECC interrupt request as signaled by the assertion of ESR[R1BC]. The address, attributes, and data are also captured in the REAR, RESR, REMR, REAT, and REDR registers.
3 EF1BR	Enable Flash 1-bit Reporting 0 Reporting of single-bit flash memory corrections is disabled. 1 Reporting of single-bit flash memory corrections is enabled. This bit can only be set if the SoC-configurable input enable signal is asserted. The occurrence of a single-bit flash memory correction generates an ECSM ECC interrupt request as signaled by the assertion of ESR[F1BC]. The address, attributes, and data are also captured in the FEAR, FEMR, FEAT, and FEDR registers.
6 ERNCR	Enable RAM Non-Correctable Reporting 0 Reporting of non-correctable RAM errors is disabled. 1 Reporting of non-correctable RAM errors is enabled. The occurrence of a non-correctable multi-bit RAM error generates an ECSM ECC interrupt request as signaled by the assertion of ESR[RNCE]. The faulting address, attributes, and data are also captured in the REAR, RESR, REMR, REAT, and REDR registers.
7 EFNCR	Enable Flash Non-Correctable Reporting 0 Reporting of non-correctable flash memory errors is disabled. 1 Reporting of non-correctable flash memory errors is enabled. The occurrence of a non-correctable multi-bit flash memory error generates an ECSM ECC interrupt request as signaled by the assertion of ESR[FNCE]. The faulting address, attributes, and data are also captured in the FEAR, FEMR, FEAT, and FEDR registers.

16.4.2.9 ECC Status Register (ESR)

The ECC Status Register is an 8-bit control register for signaling which types of properly enabled ECC events have been detected. The ESR signals the last properly enabled memory event to be detected. ECC interrupt generation is separated into single-bit error detection/correction, uncorrectable error detection,

last properly enabled memory event, and the combination of the two as defined by the following Boolean equations. In these equations, “&” refers to a bitwise AND operator, and “|” refers to a bitwise OR operator; bitwise AND has precedence over bitwise OR.

```
ECSM_ECC1BIT_IRQ
    =  ECR[ER1BR] &  ESR[R1BC]// RAM, 1-bit correction
    |  ECR[EF1BR] &  ESR[F1BC]// Flash, 1-bit correction
ECSM_ECCRNCR_IRQ
    =  ECR[ERNCR] &  ESR[RNCE]// RAM, noncorrectable error
ECSM_ECCFNCR_IRQ
    =  ECR[EFNCR] &  ESR[FNCE]// Flash, noncorrectable error
ECSM_ECC2BIT_IRQ
    =  ECSM_ECCRNCR_IRQ// RAM, noncorrectable error
    |  ECSM_ECCFNCR_IRQ// Flash, noncorrectable error
ECSM_ECC_IRQ
    =  ECSM_ECC1BIT_IRQ // 1-bit correction
    |  ECSM_ECC2BIT_IRQ// noncorrectable error
```

where the combination of a properly enabled category in the ECR and the detection of the corresponding condition in the ESR produces the interrupt request.

The ECSM allows a maximum of one bit of the ESR to be asserted at any given time. This preserves the association between the ESR and the corresponding address and attribute registers, which are loaded on each occurrence of a properly enabled ECC event. If there is a pending ECC interrupt and another properly enabled ECC event occurs, the ECSM hardware automatically handles the ESR reporting, clearing the previous data and loading the new state, thus guaranteeing that only a single flag is asserted.

To maintain the coherent software view of the reported event, the following sequence in the ECSM error interrupt service routine is recommended:

1. Read the ESR and save it.
2. Read and save all the address and attribute reporting registers.
3. Re-read the ESR and verify the current contents match the original contents. If the two values are different, go back to step 1 and repeat.
4. When the values are identical, write a 1 to the asserted ESR flag to negate the interrupt request.

See Figure 16-8 and Table 16-9 for the ECC Status Register definition.

Address: Base + 0x0047 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	R1BC	F1BC	0	0	RNCE	FNCE
W			w1c	w1c			w1c	w1c
Reset	0	0	0	0	0	0	0	0

Figure 16-8. ECC Status Register (ESR)

Table 16-9. ECC Status (ESR) field descriptions

Name	Description
2 R1BC	RAM 1-bit Correction 0 No reportable single-bit RAM correction has been detected. 1 A reportable single-bit RAM correction has been detected. This bit can only be set if ECR[EPR1BR] is asserted. The occurrence of a properly enabled single-bit RAM correction generates an ECSM ECC interrupt request. The address, attributes, and data are also captured in the REAR, RESR, REMR, REAT, and REDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.
3 F1BC	Flash 1-bit Correction 0 No reportable single-bit flash memory correction has been detected. 1 A reportable single-bit flash memory correction has been detected. This bit can only be set if ECR[EPF1BR] is asserted. The occurrence of a properly enabled single-bit flash memory correction generates an ECSM ECC interrupt request. The address, attributes, and data are also captured in the FEAR, FEMR, FEAT, and FEDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.
6 RNCE	RAM Non-Correctable Error 0 No reportable non-correctable RAM error has been detected. 1 A reportable non-correctable RAM error has been detected. The occurrence of a properly enabled non-correctable RAM error generates an ECSM ECC interrupt request. The faulting address, attributes, and data are also captured in the REAR, RESR, REMR, REAT, and REDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.
7 FNCE	Flash Non-Correctable Error 0 No reportable non-correctable flash memory error has been detected. 1 A reportable non-correctable flash memory error has been detected. The occurrence of a properly enabled non-correctable flash memory error generates an ECSM ECC interrupt request. The faulting address, attributes, and data are also captured in the FEAR, FEMR, FEAT, and FEDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.

In the event that multiple status flags are signaled simultaneously, ECSM records the event with the R1BC as highest priority, then F1BC, then RNCE, and finally FNCE.

16.4.2.10 ECC Error Generation Register (EEGR)

The ECC Error Generation Register is a 16-bit control register used to force the generation of single- and double-bit data inversions in the memories with ECC, most notably the RAM. This capability is provided for two purposes:

- It provides a software-controlled mechanism for “injecting” errors into the memories during data writes to verify the integrity of the ECC logic.
- It provides a mechanism to allow testing of the software service routines associated with memory error logging.

It should be noted that while the EEGR is associated with the RAM, similar capabilities exist for the flash memory. The ability to program the non-volatile memory with single- or double-bit errors is supported for the same two reasons previously identified.

For both types of memories (RAM and flash memory), the intent is to generate errors during data write cycles, such that subsequent reads of the corrupted address locations generate ECC events, either single-bit corrections or double-bit noncorrectable errors that are terminated with an error response.

The enabling of these error generation modes requires the same SoC-configurable input enable signal (as that used to enable single-bit correction reporting) be asserted.

See Figure 16-9 and Table 16-10 for the ECC Configuration Register definition.

Address: Base + 0x004A Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FRC 1BI	FR11 BI	0	0	FRC NCI	FR1 NCI	0	ERRBIT[0:6]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-9. ECC Error Generation Register (EEGR)

Table 16-10. ECC Error Generation (EEGR) field descriptions

Name	Description
2 FRC1BI	Force RAM Continuous 1-Bit Data Inversions 0 No RAM continuous 1-bit data inversions are generated. 1 1-bit data inversions in the RAM are continuously generated. The assertion of this bit forces the RAM controller to create 1-bit data inversions, as defined by the bit position specified in ERRBIT[0:6], continuously on every write operation. The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM. After this bit has been enabled to generate another continuous 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic. This bit can only be set if the same SoC configurable input enable signal (as that used to enable single-bit correction reporting) is asserted. **Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior.

Table 16-10. ECC Error Generation (EEGR) field descriptions (continued)

Name	Description
3 FR11BI	Force RAM One 1-bit Data Inversion 0 No RAM single 1-bit data inversion is generated. 1 One 1-bit data inversion in the RAM is generated. The assertion of this bit forces the RAM controller to create one 1-bit data inversion, as defined by the bit position specified in ERRBIT[0:6], on the first write operation after this bit is set. The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM. After this bit has been enabled to generate a single 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic. This bit can only be set if the same SoC configurable input enable signal (as that used to enable single-bit correction reporting) is asserted. **Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior.
6 FRCNCI	Force RAM Continuous Noncorrectable Data Inversions 0 No RAM continuous 2-bit data inversions are generated. 1 2-bit data inversions in the RAM are continuously generated. The assertion of this bit forces the RAM controller to create 2-bit data inversions, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, continuously on every write operation. After this bit has been enabled to generate another continuous noncorrectable data inversion, it must be cleared before being set again to properly re-enable the error generation logic. The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM. **Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior.
7 FR1NCI	Force RAM One Noncorrectable Data Inversions 0 No RAM single 2-bit data inversions are generated. 1 One 2-bit data inversion in the RAM is generated. The assertion of this bit forces the RAM controller to create one 2-bit data inversion, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, on the first write operation after this bit is set. The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM. After this bit has been enabled to generate a single 2-bit error, it must be cleared before being set again to properly re-enable the error generation logic. **Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior.

Table 16-10. ECC Error Generation (EEGR) field descriptions (continued)

Name	Description
9-15 ERRBIT [0:6]	Error Bit Position The vector defines the bit position which is complemented to create the data inversion on the write operation. For the creation of 2-bit data inversions, the bit specified by this field plus the odd parity bit of the ECC code are inverted. The RAM controller follows a vector bit ordering scheme where LSB=0. Errors in the ECC syndrome bits can be generated by setting this field to a value greater than the RAM width. For example, consider a 32-bit RAM implementation. The 32-bit ECC approach requires 7 code bits for a 32-bit word. For PRAM data width of 32 bits, the actual SRAM (32b data + 7b for ECC) = 39 bits. The following association between the ERRBIT field and the corrupted memory bit is defined: if ERRBIT = 0, then RAM[0] of the odd bank is inverted if ERRBIT = 1, then RAM[1] of the odd bank is inverted ... if ERRBIT = 31, then RAM[31] of the odd bank is inverted if ERRBIT = 64, then ECC Parity[0] of the odd bank is inverted if ERRBIT = 65, then ECC Parity[1] of the odd bank is inverted ... if ERRBIT = 70, then ECC Parity[6] of the odd bank is inverted For ERRBIT values of 32 to 63 and greater than 70, no bit position is inverted.

If an attempt to force a non-correctable inversion (by asserting EEGR[FRCNCI] or EEGR[FRC1NCI]) and EEGR[ERRBIT] equals 64, then no data inversion will be generated.

16.4.2.11 Flash ECC Address Register (FEAR)

The FEAR is a 32-bit register for capturing the address of the last properly enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash memory causes the address, attributes, and data associated with the access to be loaded into the FEAR, FEMR, FEAT, and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-10 and Table 16-11 for the Flash ECC Address Register definition.

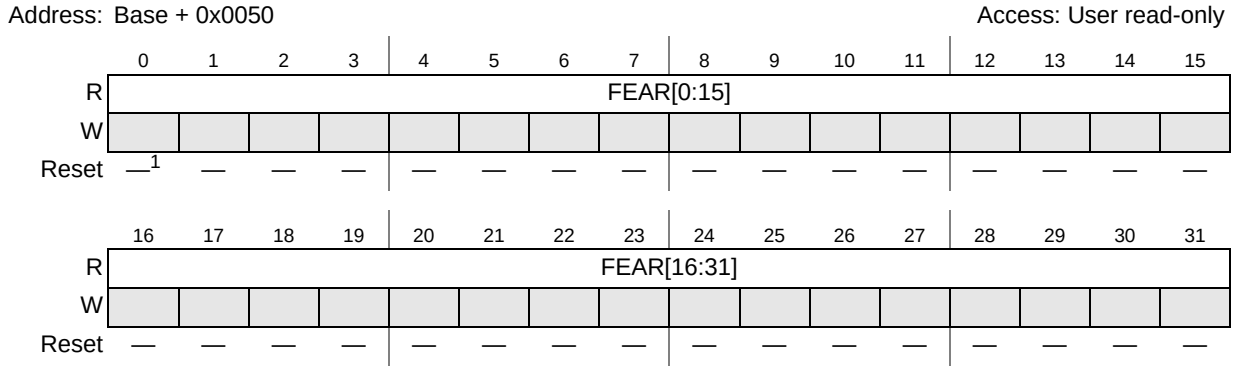


Figure 16-10. Flash ECC Address Register (FEAR)

1 Value is undefined at reset.

Table 16-11. Flash ECC Address (FEAR) field descriptions

Name	Description
0-31 FEAR[0:31]	Flash ECC Address Register This 32-bit register contains the faulting access address of the last properly enabled flash memory ECC event.

16.4.2.12 Flash ECC Master Number Register (FEMR)

The FEMR is a 4-bit register for capturing the XBAR bus master number of the last properly enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash memory causes the address, attributes, and data associated with the access to be loaded into the FEAR, FEMR, FEAT, and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-11 and Table 16-12 for the Flash ECC Master Number Register definition.

Address: Base + 0x0056 Access: User read-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	FEMR[0:3]			
W								
Reset	—[1]	—	—	—	—	—	—	—

Figure 16-11. Flash ECC Master Number Register (FEMR)

[1] Value is undefined at reset.

Table 16-12. Flash ECC Master Number (FEMR) field descriptions

Name	Description
4-7 FEMR[0:3]	Flash ECC Master Number Register This 4-bit register contains the XBAR bus master number of the faulting access of the last properly enabled flash memory ECC event.

16.4.2.13 Flash ECC Attributes (FEAT) register

The FEAT is an 8-bit register for capturing the XBAR bus master attributes of the last properly enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash memory causes the address, attributes, and data associated with the access to be loaded into the FEAR, FEMR, FEAT, and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-12 and Table 16-13 for the Flash ECC Attributes Register definition.

Address: Base + 0x0057 Access: User read-only

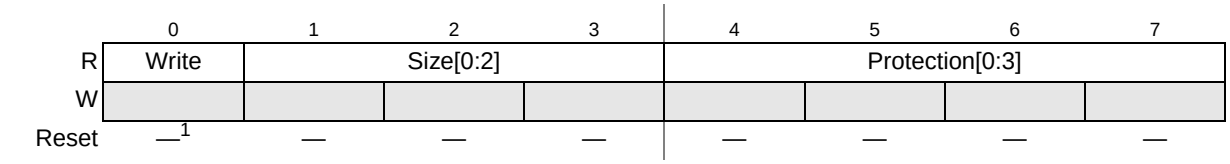


Figure 16-12. Flash ECC Attributes (FEAT) register

1 Value is undefined at reset.

Table 16-13. Flash ECC Attributes (FEAT) field descriptions

Name	Description
0 Write	AMBA-AHB HWRITE 0 AMBA-AHB read access 1 AMBA-AHB write access
1-3 Size[0:2]	AMBA-AHB HSIZE[0:2] 000 8-bit AMBA-AHB access 001 16-bit AMBA-AHB access 010 32-bit AMBA-AHB access 1xx Reserved
4-7 Protection[0:3]	AMBA-AHB HPROT[0:3] Protection[3]: Cacheable 0 = Non-cacheable,1 = Cacheable Protection[2]: Bufferable 0 = Non-bufferable,1 = Bufferable Protection[1]: Mode 0 = User mode, 1 = Supervisor mode Protection[0]: Type 0 = I-Fetch, 1 = Data

16.4.2.14 Flash ECC Data Register (FEDR)

The FEDR is a 32-bit register for capturing the data associated with the last properly enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash memory causes the address, attributes, and data associated with the access to be loaded into the FEAR, FEMR, FEAT, and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-13 and Table 16-14 for the Flash ECC Data Register definition.

Address: Base + 0x005C Access: User read-only

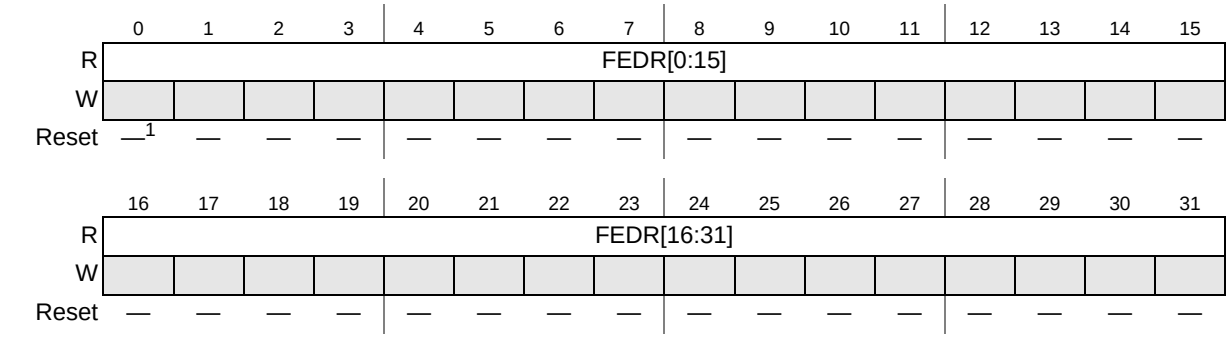


Figure 16-13. Flash ECC Data Register (FEDR)

1 Value is undefined at reset.

Table 16-14. Flash ECC Data (FEDR) field descriptions

Name	Description
0-31 FEDR[0:31]	Flash ECC Data Register This 32-bit register contains the data associated with the faulting access of the last properly enabled flash memory ECC event. The register contains the data value taken directly from the data bus.

16.4.2.15 RAM ECC Address Register (REAR)

The REAR is a 32-bit register for capturing the address of the last properly enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes, and data associated with the access to be loaded into the REAR, RESR, REMR, REAT, and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-14 and Table 16-15 for the RAM ECC Address Register definition.

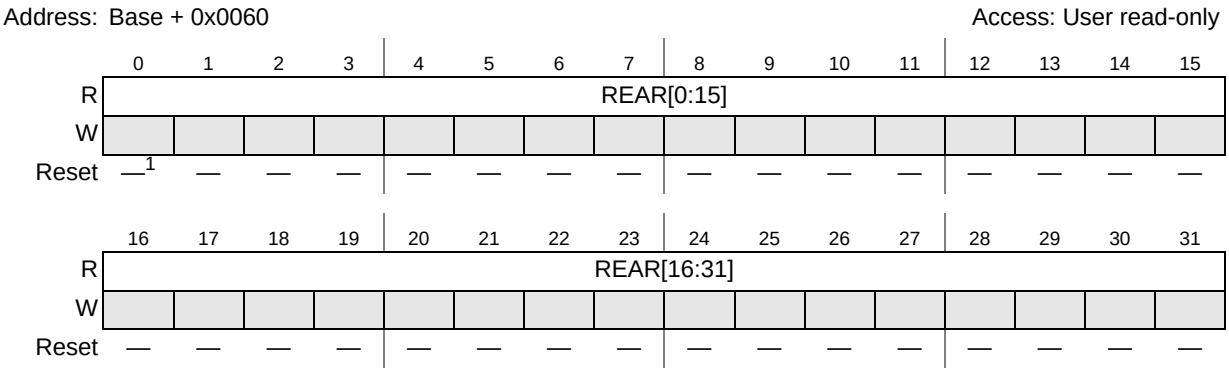


Figure 16-14. RAM ECC Address Register (REAR)

1 Value is undefined at reset.

Table 16-15. RAM ECC Address (REAR) field descriptions

Name	Description
0-31 REAR[0:31]	RAM ECC Address Register This 32-bit register contains the faulting access address of the last properly enabled RAM ECC event.

16.4.2.16 RAM ECC Syndrome Register (RESR)

The RESR is an 8-bit register for capturing the error syndrome of the last properly enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes, and data associated with the access to be loaded into the REAR, RESR, REMR, REAT, and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-15 and Table 16-16 for the RAM ECC Syndrome Register definition.

Address: Base + 0x0065 Access: User read-only

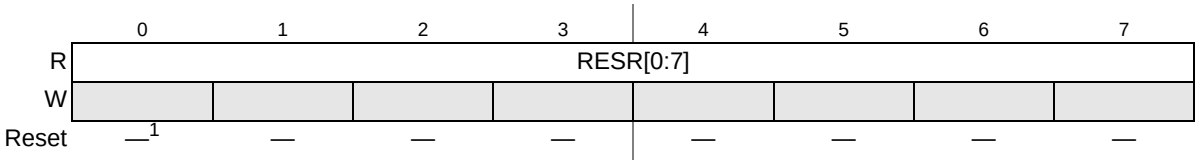


Figure 16-15. RAM ECC Syndrome Register (RESR)

1 Value is undefined at reset.

Table 16-16. RAM ECC Syndrome (RESR) field descriptions

Name	Description
0-7 RESR[0:7]	RAM ECC Syndrome Register This 8-bit syndrome field includes 6 bits of Hamming decoded parity plus an odd-parity bit for the entire 39-bit (32-bit data + 7 ECC) code word. The upper 7 bits of the syndrome specify the exact bit position in error for single-bit correctable code words, and the combination of a non-zero 7-bit syndrome plus overall incorrect parity bit signal a multi-bit, non-correctable error. For correctable single-bit errors, the mapping shown in Table 16-16 associates the upper 7 bits of the syndrome with the data bit in error.

Note: Table 16-16 associates the upper 7 bits of the ECC syndrome with the exact data bit in error for single-bit correctable code words. This table follows the bit vectoring notation where the LSB=0. Note that the syndrome value of 0x01 implies no error condition but this value is not readable when the PRESR is read for the no-error case.

Table 16-17. RAM syndrome mapping for single-bit correctable errors

RESR[0:7]	Data bit in error	RESR[0:7]	Data bit in error
0x00	ECC ODD[0]	0x28	DATA ODD BANK[17]
0x01	No Error	0x2A	DATA ODD BANK[16
0x02	ECC ODD[1]	0x2C	DATA ODD BANK[15]
0x04	ECC ODD[2]	0x58	DATA ODD BANK[14]
0x06	DATA ODD BANK[31]	0x30	DATA ODD BANK[13]
0x08	ECC ODD[3]	0x32	DATA ODD BANK[12]
0x0A	DATA ODD BANK[30]	0x34	DATA ODD BANK[11]
0x0C	DATA ODD BANK[29]	0x64	DATA ODD BANK[10]
0x0E	DATA ODD BANK[28]	0x38	DATA ODD BANK[9]
0x10	ECC ODD[4]	0x62	DATA ODD BANK[8]
0x12	DATA ODD BANK[27]	0x70	DATA ODD BANK[7]
0x14	DATA ODD BANK[26]	0x60	DATA ODD BANK[6]
0x16	DATA ODD BANK[25]	0x40	ECC ODD[6]
0x18	DATA ODD BANK[24]	0x42	DATA ODD BANK[5]
0x1A	DATA ODD BANK[23]	0x44	DATA ODD BANK[4]
0x1C	DATA ODD BANK[22]	0x46	DATA ODD BANK[3]
0x50	DATA ODD BANK[21]	0x48	DATA ODD BANK[2]
0x20	ECC ODD[5]	0x4A	DATA ODD BANK[1]
0x22	DATA ODD BANK[20]	0x4C	DATA ODD BANK[0]
0x24	DATA ODD BANK[19]	0x03,0x05.....0x4D	Multiple bit error
0x26	DATA ODD BANK[18]	> 0x4D	Multiple bit error

16.4.2.17 RAM ECC Master Number Register (REMR)

The REMR is a 4-bit register for capturing the XBAR bus master number of the last properly enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes, and data associated with the access to be loaded into the REAR, RESR, REMR, REAT, and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-16 and Table 16-18 for the RAM ECC Master Number Register definition.

Address: Base + 0x0066 Access: User read-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	REMR[0:3]			
W								
Reset	—[1]	—	—	—	—	—	—	—

Figure 16-16. RAM ECC Master Number Register (REMR)

1 Value is undefined at reset.

Table 16-18. RAM ECC Master Number (REMR) field descriptions

Name	Description
4-7 REMR[0:3]	RAM ECC Master Number Register This 4-bit register contains the XBAR bus master number of the faulting access of the last properly enabled RAM ECC event.

16.4.2.18 RAM ECC Attributes (REAT) register

The REAT is an 8-bit register for capturing the XBAR bus master attributes of the last properly enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes, and data associated with the access to be loaded into the REAR, RESR, REMR, REAT, and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-17 and Table 16-19 for the RAM ECC Attributes Register definition.

Address: Base + 0x0067 Access: User read-only

	0	1	2	3	4	5	6	7
R	Write	Size[0:2]			Protection[0:3]			
W								
Reset	—[1]	—	—	—	—	—	—	—

Figure 16-17. RAM ECC Attributes (REAT) Register

1 Value is undefined at reset.

Table 16-19. RAM ECC Attributes (REAT) field descriptions

Name	Description
0 Write	AMBA-AHB HWRITE 0 AMBA-AHB read access 1 AMBA-AHB write access
1-3 Size[0:2]	AMBA-AHB HSIZE[0:2] 000 8-bit AMBA-AHB access 001 16-bit AMBA-AHB access 010 32-bit AMBA-AHB access 1xx Reserved
4-7 Protection[0:3]	AMBA-AHB HPROT[0:3] Protection[3]: Cacheable 0 = Non-cacheable, 1 = Cacheable Protection[2]: Bufferable 0 = Non-bufferable,1 = Bufferable Protection[1]: Mode 0 = User mode, 1 = Supervisor mode Protection[0]: Type 0 = I-Fetch, 1 = Data

16.4.2.19 RAM ECC Data Register (REDR)

The REDR is a 32-bit register for capturing the data associated with the last properly enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes, and data associated with the access to be loaded into the REAR, RESR, REMR, REAT, and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 16-18 and Table 16-20 for the RAM ECC Data Register definition.

Address: Base + 0x006C Access: User read-only

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

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

Figure 16-18. RAM ECC Data Register (REDR)

[1] Value is undefined at reset.

Table 16-20. RAM ECC Data (REDR) field descriptions

Name	Description
0-31 REDR[0:31]	RAM ECC Data Register This 32-bit register contains the data associated with the faulting access of the last properly enabled RAM ECC event. The register contains the data value taken directly from the data bus.

16.4.3 High-priority enables

e200 processors can be configured to support critical and/or external interrupts. Furthermore, each processor can be configured to employ priority elevation on critical and/or external interrupt events. Critical interrupts come from outside the platform, and are routed directly to the processor's critical interrupt input. External interrupts are routed through the interrupt controller. In addition to the interrupt notification signals, various processor-specific configuration flags from the processor's Machine Check Register (MCR[ee,ce]) and the Hardware Implementation Register (HID1) are sent to the ECSM to determine when interrupt servicing is enabled and when high-priority elevation should be enabled. If the corresponding processor is configured to allow high-priority elevation on critical interrupt events, the ECSM generates the high-priority signal upon critical interrupt detection and holds it active throughout the duration of interrupt servicing. If the corresponding processor is configured to allow high-priority elevation on external interrupt events, the ECSM generates the high-priority signal upon external interrupt detection and holds it active throughout the duration of interrupt servicing. During interrupt servicing the processor status output, p_stat, is monitored for indication of a return from interrupt (rfi).

Great care needs to be taken when using the priority elevation as it can enable a master to starve the rest of the masters in the system. Please see Chapter 10, Crossbar Switch (XBAR), for information on priority elevation.

16.4.4 Spp_ips_reg_protection

The spp_ips_reg_protection logic provides hardware enforcement of supervisor mode access protection for five on-platform IPS modules: INTC, ECSM, MPU, STM, and SWT. This logic resides between the on-platform bus sourced by the PBRIDGE bus controller and the individual slave modules. It monitors the bus access type (supervisor or user) and if a user access is attempted, the transfer is terminated with an error and inhibited from reaching the slave module. Identical logic is replicated for each of the five targeted slave modules. A block diagram of the spp_ips_reg_protection module is shown in Figure 16-19.

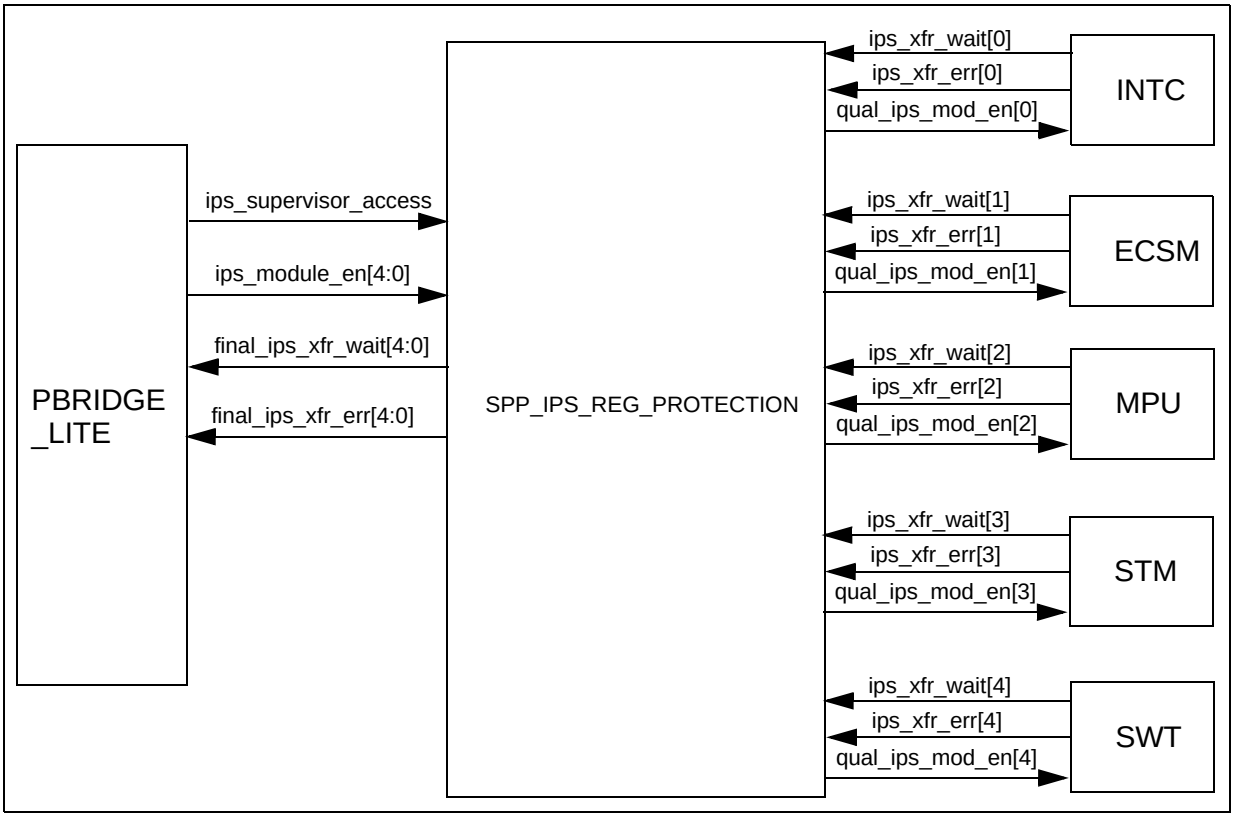


Figure 16-19. Spp_Ips_Reg_Protection Block Diagram

Chapter 17
Flash Memory

17.1 Introduction

The flash memory comprises a platform flash controller (PFlash) interface and three flash memory arrays: two arrays of up to 512 KB for code flash memory (CFlash) and one array of 64 KB for data flash memory (DFlash). The flash memory architecture of this device is illustrated in Figure 17-1. See Table 2-1 in Chapter 2, Memory Map, for exact memory sizes of each family member.

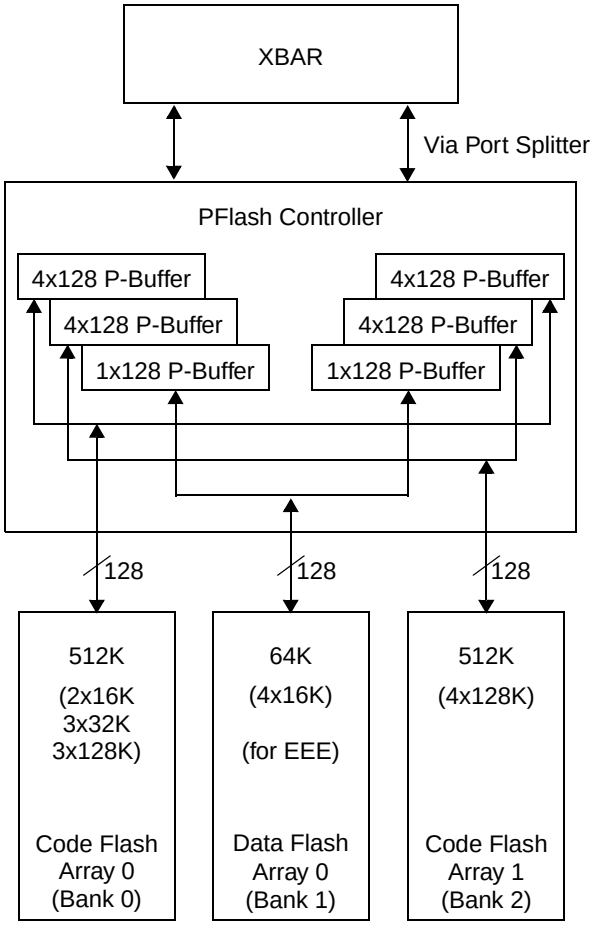


Figure 17-1. MPC5606S flash memory architecture

17.2 Program flash memory (code flash 0 and code flash 1)

17.2.1 Introduction

The primary function of the program flash module is to serve as electrically programmable and erasable non-volatile memory.

Non-volatile memory may be used for instruction and/or data storage.

The module is a non-volatile solid-state silicon memory device consisting of blocks (called also sectors) of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The flash module is arranged as two functional units: the flash core and the memory interface.

The flash core is composed of arrayed non-volatile storage elements, sense amplifiers, row decoders, column decoders, and charge pumps. The arrayed storage elements in the Flash Core are subdivided into physically separate units referred to as blocks (or sectors).

The memory interface contains the registers and logic which control the operation of the flash core. The memory interface is also the interface between the flash module and a Bus Interface Unit (BIU) and contains the ECC logic and redundancy logic.

A BIU connects the flash module to a system bus, and contains all system-level customization required for the SoC application.

17.2.2 Main features

- High read parallelism (128 bits)
- Error Correction Code (SEC-DED) to enhance data retention
- Double-word program (64 bits)
- Sector Erase
- Double bank (code flash 0 and code flash 1): allows one bank to be read while the other bank is being modified
- Erase suspend available (program suspend not available)
- Software programmable program/erase protection to avoid unwanted writes
- Censored mode against piracy
- Usable as main code memory of the SoC
- Shadow block available
- One-time programmable (OTP) area in Test flash block

17.2.3 Block diagram

Each bank contains one flash macrocell comprising one matrix module, normally used for code storage. No Read-While-Modify operations are possible within the same bank.

The modify operations are managed by an embedded Flash Program/Erase Controller (FPEC). Commands to the FPEC are given through a User Registers Interface.

The read data bus is 128 bits wide, while the flash memory registers are on a separate 32-bit bus addressed in the user memory map.

The high voltages needed for program/erase operations are internally generated.

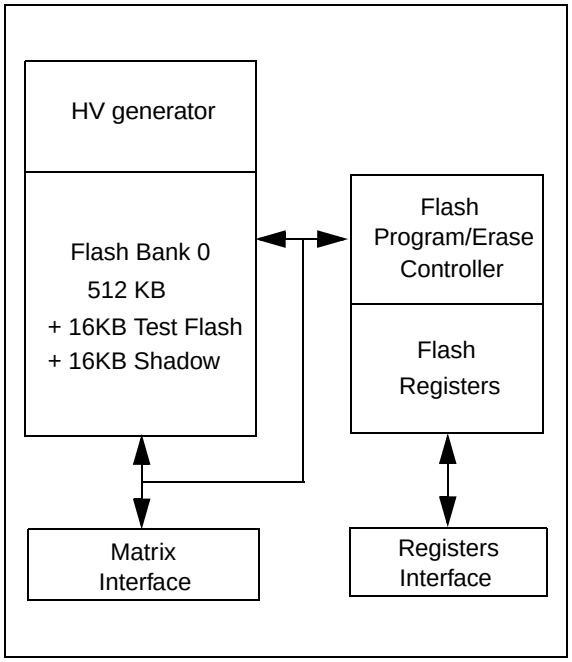


Figure 17-2. Bank 0 Flash Macrocell Structure

17.2.4 Functional description

17.2.4.1 Macrocell structure

The flash module is addressable by word (32 bits) or double-word (64 bits) for program, and page (128 bits) for read. Reads done to the flash module always return 128 bits, although read page buffering may be done in the platform BIU.

Each read of the flash module retrieves a page, or four consecutive words (128 bits) of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits (3:2).

The flash module supports fault tolerance through Error Correction Code (ECC) and/or error detection. The ECC implemented within the flash module will correct single bit failures and detect double bit failures.

The flash module uses an embedded hardware algorithm implemented in the memory interface to program and erase the flash core.

Control logic that works with the software block enables, and software lock mechanisms, is included in the embedded hardware algorithm to guard against accidental program/erase commands.

The hardware algorithm perform the steps necessary to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the flash module reads as logic level 0 (or low).

An erased bit in the flash module reads as logic level 1 (or high).

Program and erase of the flash module requires multiple system clock cycles to complete.

The erase sequence may be suspended.

The program and erase sequences may be aborted.

17.2.4.2 Flash module sectorization

Bank 0 (Code flash 0) of the flash module supports up to 512 KB of user memory, plus 16 KB of test memory (a portion of which is one-time programmable by the user). An extra 16 KB sector is available as Shadow space usable for user option bits or censorship.

Bank 1 (Data flash 0) of the flash module supports 64 KB of user memory, plus 16 KB of test memory (a portion of which is one-time programmable by the user).

Bank 2 (Code flash 1) of the flash module supports up to 512 KB of user memory, plus 16 KB of test memory (a portion of which is one-time programmable by the user).

The sectorization of the flash module is shown in Table 17-1.

Table 17-1. Flash module sectorization

AP		Size [KB]	5602S	5604S	5606S	Sector	Region
Start Address	End Address						
On-chip Flash Memories (Code Flash)							
0x00000000	0x00007FFF	32	Yes	Yes	Yes[1]	B0F0	Code Flash Array 0
0x00008000	0x0000BFFF	16	Yes	Yes	Yes[1]	B0F1	Code Flash Array 0
0x0000C000	0x0000FFFF	16	Yes	Yes	Yes[1]	B0F2	Code Flash Array 0
0x00010000	0x00017FFF	32	Yes	Yes	Yes[1]	B0F3	Code Flash Array 0
0x00018000	0x0001FFFF	32	Yes	Yes	Yes[1]	B0F4	Code Flash Array 0
0x00020000	0x0003FFFF	128	Yes	Yes	Yes[1]	B0F5	Code Flash Array 0
0x00040000	0x0005FFFF	128	No	Yes	Yes[1]	B0F6	Code Flash Array 0
0x00060000	0x0007FFFF	128	No	Yes	Yes[1]	B0F7	Code Flash Array 0
0x00080000	0x0009FFFF	128	No	No	Yes[1]	B2F0	Code Flash Array 1
0x000A0000	0x000BFFFF	128	No	No	Yes[1]	B2F1	Code Flash Array 1
0x000C0000	0x000DFFFF	128	No	No	Yes[1]	B2F2	Code Flash Array 1
0x000E0000	0x000FFFFF	128	No	No	Yes[1]	B2F3	Code Flash Array 1
0x00100000	0x001FFFFF	1024	—	—	—	—	Reserved
On-chip Flash Memories (Shadow for Code Flash)							
0x00200000	0x00203FFF	16	Yes	Yes	Yes	—	Code Flash Array 0 Shadow block
0x00204000	0x003FFFFF	2032	—	—	—	—	Reserved
On-chip Flash Memories (Test for Code Flash)							

Table 17-1. Flash module sectorization (continued)

AP		Size [KB]	5602S	5604S	5606S	Sector	Region
Start Address	End Address						
0x00400000	0x00403FFF	16	Yes	Yes	Yes	—	Code Flash Array 0 Test Sector
0x00404000	0x0047FFFF	496	—	—	—	—	Reserved
0x00480000	0x00483FFF	16	No	No	Yes	—	Code Flash Array 1 Test Sector
0x00484000	0x007FFFFF	3568	—	—	—	—	Reserved
On-chip Flash Memories (Data Flash)							
0x00800000	0x00803FFF	16	Yes	Yes	Yes	B1F0	Data Flash Array 0
0x00804000	0x00807FFF	16	Yes	Yes	Yes	B1F1	Data Flash Array 0
0x00808000	0x0080BFFF	16	Yes	Yes	Yes	B1F2	Data Flash Array 0
0x0080C000	0x0080FFFF	16	Yes	Yes	Yes	B1F3	Data Flash Array 0
0x00810000	0x00BFFFFF	4032	—	—	—	—	Reserved
On-chip Flash Memories (Test for Data Flash)							
0x00C00000	0x00C03FFF	16	Yes	yes	Yes	—	Data Flash Array 0 Test Sector
0x00C04000	0x00FFFFFF	2032	—	—	—	—	Reserved

[1] Flash sector can be accessed via both slave ports. Arbitration logic required in case two different masters attempt to access the same address at the same time.

The flash module is divided into blocks also to implement independent erase/program protection. A software mechanism is provided to independently lock/unlock each block (as reported in Table 17-1) against program and erase.

17.2.4.2.1 Test flash block

The Test flash block exists outside the normal address space and is programmed and read independently of the other blocks. The independent Test flash block is included also to support systems which require non-volatile memory for security and/or to store system initialization information.

A section of the Test flash is reserved to store the non-volatile information related to redundancy, configuration, and protection.

The ECC is applied also to Test flash.

The use of the reserved Test flash sector is detailed in the following table.

Table 17-2. Test flash structure for Code flash 0 (block 0)

Name	Description	Addresses	Size
	User OTP Area	0x400000 to 0x401FFF	8192 byte
	Reserved	0x402000 to 0x403CFF	7424 byte
	User Reserved	0x403D00 to 0x403DE7	232 byte

Table 17-2. Test flash structure for Code flash 0 (block 0) (continued)

Name	Description	Addresses	Size
NVLML	Non-Volatile Low/Mid Address Space Block Locking register	0x403DE8 to 0x403DEF	8 byte
NVHBL	Non-Volatile High Address Space Block Locking Register	0x403DF0 to 0x403DF7	8 byte
NVSLL	Non-Volatile Secondary Low/Mid Address Space Block Lock Register	0x403DF8 to 0x403DFF	8 byte
	User Reserved	0x403E00 to 0x403EFF	256 byte
	Reserved	0x403F00 to 0x403FFF	256 byte

Table 17-3. Test flash structure for Data flash 0 (block 1)

Name	Description	Addresses	Size
	User OTP Area	0xC00000 to 0xC01FFF	8192 byte
	Reserved	0xC02000 to 0xC03CFF	7424 byte
	User Reserved	0xC03D00 to 0xC03DE7	232 byte
NVLML	Non-Volatile Low/Mid Address Space Block Locking Register	0xC03DE8 to 0xC03DEF	8 byte
NVHBL	Non-Volatile High Address space Block Locking Register	0xC03DF0 to 0xC03DF7	8 byte
NVSLL	Non-Volatile Secondary Low/Mid Address Space Block Lock Register	0xC03DF8 to 0xC03DFF	8 byte
	User Reserved	0xC03E00 to 0xC03EFF	256 byte
	Reserved	0xC03F00 to 0xC03FFF	256 byte

Table 17-4. Test flash structure for Code flash 1 (block 2)

Name	Description	Addresses	Size
	User OTP Area	0x480000 to 0x481FFF	8192 byte
	Reserved	0x482000 to 0x483CFF	7424 byte
	User Reserved	0x483D00 to 0x483DE7	232 byte
NVLML	Non-Volatile Low/Mid Address Space Block Locking Register	0x483DE8 to 0x483DEF	8 byte
NVHBL	Non-Volatile High Address Space Block Locking Register	0x483DF0 to 0x483DF7	8 byte
NVSLL	Non-Volatile Secondary Low/Mid Address Space Block Lock Register	0x483DF8 to 0x483DFF	8 byte
	User Reserved	0x483E00 to 0x483EFF	256 byte
	Reserved	0x483F00 to 0x483FFF	256 byte

Erase of Test flash block is always locked.

Program of the Test flash block has similar restriction as the array in terms of how ECC is calculated. Only one program is allowed per 64-bit ECC segment.

The first 8 KB of the Test flash block may be used for user defined functions (possibly to store serial numbers, other configuration words or factory process codes). Locations of the Test flash other than the first 8 KB of OTP Area cannot be programmed by the user application.

17.2.4.4.2.2 Shadow block

A Shadow block is present in Code flash 0 (Block 0).

The Shadow block can be enabled by the BIU.

When the Shadow space is enabled, all the operations are mapped to the Shadow block.

User mode program and erase of the Shadow block are enabled only when MCR.PEAS is high.

The Shadow block may be locked/unlocked against program or erase by using the LML.TSLK and SLL.STSLK registers.

Program of the Shadow block has similar restriction as the array in terms of how ECC is calculated. Only one program is allowed per 64-bit ECC segment between erases.

Erase of the Shadow block is done similarly as a sector erase.

The Shadow block contains specified data needed for user features.

The user area of the Shadow block may be used for user defined functions (possibly to store boot code, other configuration words, or factory process codes).

The use of the Shadow block is detailed in the following table:

Table 17-5. Shadow block structure

Name	Description	Addresses	Size
—	Start of Shadow block	0x200000 to 0x200007	8 byte
—	User Area	0x200008 to 0x203DCF	15816 byte
—	Reserved	0x203DD0 to 0x203DD7	8 byte
NVPWD0–1	Non-Volatile Private Censorship Password 0–1 Registers	0x203DD8 to 0x203DDF	8 byte
NVSCI0–1	Non-Volatile System Censorship Information 0–1 Registers	0x203DE0 to 0x203DE7	8 byte
—	Reserved	0x203DE8 to 0x203DFF	24 byte
NVBIU2-3	Non-Volatile Bus Interface Unit 2-3 Registers	0x203E00 to 0x203E0F	16 byte
—	Reserved	0x203E10 to 0x203E17	8 byte
NVUSRO	Non-Volatile User Options Register	0x203E18 to 0x203E1F	8 byte
—	Reserved	0x203E20 to 0x203FFF	480 byte

17.2.5 User mode operation

In User mode, the flash module may be read and written (register writes and interlock writes), programmed, or erased.

The default state of the flash module is read.

The main, shadow and test address space can be read only in the read state.

The flash memory registers are always available for read, also when the module is in power-down mode (except few documented registers).

The flash module enters the read state on reset.

The Module is in the read state under two sets of conditions:

- The read state is active when the Module is enabled (User mode read)
- The read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend).

Notice that no Read-While-Modify is available.

Flash core reads return 128 bits (1 page = 2 double words).

Registers reads return 32 bits (1 word).

Flash core reads are done through the Bus Interface Unit.

Registers reads to unmapped register address space will return all 0's.

Registers writes to unmapped register address space will have no effect.

Array reads attempted to invalid locations will result in indeterminate data. Invalid locations occur when addressing is done to blocks that do not exist in non 2^n array sizes.

Interlock writes attempted to invalid locations, will result in an interlock occurring, but attempts to program these blocks will not occur since they are forced to be locked. Erase will occur to selected and unlocked blocks even if the interlock write is to an invalid location.

Simultaneous read cycles on the flash matrix and read/write cycles on the registers are possible. On the contrary, register read/write accesses simultaneous to a flash matrix interlock write are forbidden.

WARNING

Software executing from flash must not write to registers that control flash behavior, e.g., wait state settings or prefetch enable/disable. Doing so can cause data corruption. On MPC5606S devices these registers include PFCR0, BIU1, and BIU2. Further, flash configuration registers should be written only with 32-bit write operations to avoid any issues associated with register "incoherency" caused by bit fields spanning smaller size (8- and 16-bit) boundaries.

17.2.5.1 Reset

A reset is the highest priority operation for the flash module and terminates all other operations.

The flash module uses reset to initialize register and status bits to their default reset values.

If the flash module is executing a program or erase operation (MCR.PGM = 1 or MCR.ERS = 1) and a reset is issued, the operation will be aborted and the module will disable the high voltage logic without

damage to the high voltage circuits. Reset aborts all operations and forces the flash module into User mode ready to receive accesses. Reset and power-off must not be used systematically to terminate a program or erase operation.

After reset is negated, read register access may be done, although it should be noted that registers that require updating from shadow information, or other inputs, may not read updated values until MCR.DONE transitions. MCR.DONE may be polled to determine if the flash module has transitioned out of reset. Notice that the registers cannot be written until MCR.DONE is high.

17.2.5.2 Power-Down mode

The Power-Down mode turns off all flash memory DC current sources, so that power dissipation is due only to leakage.

In Power-Down mode, no reads from or write to the module are possible.

The user may not read some registers (UMISR0–4, UT1–2 and part of UT0) until the Power-Down mode is exited. On the contrary, write access is locked on all the registers in Power-Down mode.

When enabled, the flash memory module returns to its previous state in all cases unless it was in the process of executing an erase high voltage operation at the time of entering Power-Down mode.

If the flash memory module enters Power-Down mode during an erase operation, the MCR[ESUS] bit is set. The user may resume the erase operation when the module exits Power-Down mode by clearing the MCR[ESUS] bit. MCR[EHV] must be high to resume the erase operation.

If the flash memory module is configured to enter Power-Down mode during a program operation, the operation will be completed and the Power-Down mode will be entered only after the programming ends.

If the flash memory module is put in Power-Down mode and the Vector Table remains mapped in the flash memory address space, the user must take care that the flash memory module will strongly increase the interrupt response time by adding several wait states.

It is forbidden to enter Low-Power mode when the Power-Down mode is active.

17.2.5.3 Low-Power mode

The Low-Power mode turns off most of the DC current sources within the flash module.

The module (flash core and registers) is not accessible for read or write after entering Low-Power mode.

The wakeup time from Low-Power mode is faster than the wakeup time from Power-Down mode.

The user may not read some registers (UMISR0-4, UT1-2 and part of UT0) until the Low-Power mode is exited. Write access is locked on all the registers in Low-Power mode.

When exiting from Low-Power mode the flash memory module returns to its previous state in all cases unless it was in the process of executing an erase high-voltage operation at the time of entering Low-Power mode.

If the flash memory module enters Low-Power mode during an erase operation, the MCR[ESUS] bit is set. The user may resume the erase operation when the module exits Low-Power mode by clearing the MCR[ESUS] bit. The MCR[EHV] bit must be high to resume the erase operation.

If the flash memory module is configured to enter Low-Power mode during a program operation, the operation will be completed and the Low-Power mode will be entered only after the programming ends.

It is forbidden to enter Power-Down mode when the Low-Power mode is active.

17.2.6 Register description

The flash memory user registers mapping is shown in Table 17-6.

Table 17-6. Flash 528 KB single bank registers

Register name	Address Offset	Location
Module Configuration Register (MCR)	0x0000	on page 555
Low/mid Address Space Block Locking Register (LML)	0x0004	on page 560
High Address Space Block Locking Register (HBL)	0x0008	on page 563
Secondary Low/mid Address Space Block Lock Register (SLL)	0x000C	on page 564
Low/mid Address Space Block Select Register (LMS)	0x0010	on page 567
High Address Space Block Select Register (HBS)	0x0014	on page 568
Address Register (ADR)	0x0018	on page 569
Bus Interface Unit Register 0 (BIU0)	0x001C	on page 571
Bus Interface Unit Register 1 (BIU1)	0x0020	on page 571
Bus Interface Unit Register 2 (BIU2)	0x0024	on page 572
Reserved	0x0028	
Reserved	0x002C	
Reserved	0x0030	
Reserved	0x0034	
Reserved	0x0038	
User Test Register 0 (UT0)	0x003C	on page 573
User Test Register 1 (UT1)	0x0040	on page 575
User Test Register 2 (UT2)	0x0044	on page 576
User Multiple Input Signature Register 0 (UMISR0)	0x0048	on page 576
User Multiple Input Signature Register 1 (UMISR1)	0x004C	on page 577
User Multiple Input Signature Register 2 (UMISR2)	0x0050	on page 578
User Multiple Input Signature Register 3 (UMISR3)	0x0054	on page 578
User Multiple Input Signature Register 4 (UMISR4)	0x0058	on page 579

In the following some non-volatile registers are described. Please notice that such entities are not Flip-Flops, but locations of Test flash or Shadow blocks with a special meaning.

During the flash memory initialization phase, the FPEC reads these non-volatile registers and updates their related volatile registers. When the FPEC detects ECC double errors in these special locations, it behaves in the following way:

- In case of a failing system locations (configurations, device options, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a fatal error is flagged.
- In case of failing user locations (protections, censorship, BIU, etc.), the volatile registers are filled with all 1s and the flash memory initialization ends, setting the PEG bit of MCR to 0.

In this section, the following abbreviations are used:

Table 17-7. Abbreviations

Case	Abbreviation	Description
read/write	rw	The software can read and write to these bits.
read/clear	rc	The software can read and clear to these bits.
read-only	r	The software can only read these bits.
write-only	w	The software should only write to these bits.

17.2.6.1 Module Configuration Register (MCR)

Address Offset: 0x0000

Reset value: 0x0220_0600 (code flash 0)
0x0210_0600 (code flash 1)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EDC	0	0	0	0	SIZE 2	SIZE 1	SIZE 0	0	LAS2	LAS1	LAS0	0	0	0	MAS
rc/0	r/0	r/0	r/0	r/0	r/0	r/1	r/0	r/0	r/0	r/1	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EER	RWE	0	0	PEAS	DONE	PEG	0	0	0	0	PGM	PSUS	ERS	ESUS	EHV
rc/0	rc/0	r/0	r/0	r/0	r/1	r/1	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-3. Module Configuration Register (MCR)

The Module Configuration Register is used to enable and monitor all the modify operations of the flash module.

Table 17-8. MCR field descriptions

Field	Description
0	**EDC:** *ECC Data Correction* (Read/Clear) EDC provides information on previous reads. If a ECC Single Error detection and correction occurred, the EDC bit will be set to 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user. In the event of a ECC Double Error detection, this bit will not be set. If EDC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EDC) were not corrected through ECC. Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect. The function of this bit is SoC dependent and it can be configured to be disabled. 0: Reads are occurring normally. 1: An ECC Single Error occurred and was corrected during a previous read.
1:4	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.
5:7	**SIZE2-0:** *array space SIZE 2-0* (Read Only) The value of SIZE field is dependent upon the size of the flash module, according to Table 17-9.
8	*Reserved* (Read Only). Write this bit has no effect and read this bit always outputs 0.
9:11	**LAS2-0:** *Low Address Space 2-* (Read Only) The value of the LAS field correspond to the configuration of the Low Address Space, according to Table 17-10.
12:14	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.
15	**MAS:** *Mid Address Space* (Read Only) The value of the MAS field correspond to the configuration of the Mid Address Space, according to Table 17-11.
16	**EER:** *ECC event ERror* (Read/Clear) EER provides information on previous reads. If a ECC Double Error detection occurred, the EER bit will be set to 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user. In the event of a ECC Single Error detection and correction, this bit will not be set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) were correct. Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect. 0: Reads are occurring normally. 1: An ECC double Error occurred during a previous read.

Table 17-8. MCR field descriptions (continued)

Field	Description
17	**RWE:** *Read-while-Write event Error* (Read/Clear) RWE provides information on previous reads when a Modify operation is on going. If a RWW Error occurs, the RWE bit will be set to 1. Read-While-Write Error means that a read access to the flash memory matrix has occurred while the FPEC was performing a program or erase operation or an array integrity check. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) were correct. Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect. 0: Reads are occurring normally. 1: A RWW error occurred during a previous read.
18:19	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.
20	**PEAS:** *Program/Erase Access Space* (Read Only) PEAS is used to indicate which space is valid for Program and Erase operations: main array space or shadow/test space. PEAS=0 indicates that the main address space is active for all flash module program and erase operations. PEAS=1 indicates that the test or shadow address space is active for program and erase. The value in PEAS is captured and held with the first interlock write done for Modify operations. The value of PEAS is retained between sampling events (for example, subsequent first interlock writes). 0: Shadow/Test address space is disabled for program/erase and main address space enabled. 1: Shadow/Test address space is enabled for program/erase and main address space disabled.
21	**DONE:** *modify operation DONE* (Read Only) DONE indicates if the flash module is performing a high voltage operation. DONE is set to 1 on termination of the flash module reset. DONE is cleared to 0 immediately after a 0 to 1 transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation. DONE is set to 1 at the end of program and erase high voltage sequences. DONE is set to 1 (within t_{PABT} or t_{EABT}, equal to P/E Abort Latency) after a 1 to 0 transition of EHV, which aborts a high voltage Program/Erase operation. DONE is set to 1 (within t_{ESUS}, time equals to Erase Suspend Latency) after a 0 to 1 transition of ESUS, which suspends an erase operation. 0: Flash is executing a high voltage operation. 1: Flash is not executing a high voltage operation.

Table 17-8. MCR field descriptions (continued)

Field	Description
22	**PEG:** *Program/Erase Good* (Read Only) The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation will cause PEG to be cleared to 0, indicating the sequence failed. PEG is set to 1 when the flash module is reset, unless a flash memory initialization error has been detected. The value of PEG is valid only when PGM=1 and/or ERS=1 and after DONE transitions from 0 to 1 due to an abort or the completion of a Program/Erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by ESUS being set to logic 1. If Program or Erase are attempted on blocks that are locked, the response will be PEG=1, indicating that the operation was successful, and the content of the block were properly protected from the Program or Erase operation. If a Program operation tries to program at 1 bits that are at 0, the program operation is correctly executed on the new bits to be programmed at 0, but PEG is cleared, indicating that the requested operation has failed. In Array Integrity Check or Margin mode, PEG is set to 1 when the operation is completed, regardless of the occurrence of any error. The presence of errors can be detected only by comparing the checksum value stored in UMIRS0–1. Aborting an Array Integrity Check or a Margin mode operation will cause PEG to be cleared to 0, indicating the sequence failed. 0: Program or Erase operation failed. 1: Program or Erase operation successful.
23:26	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.
27	**PGM:** *ProGraM* (Read/Write) PGM is used to setup the flash module for a Program operation. A 0 to 1 transition of PGM initiates a Program sequence. A 1 to 0 transition of PGM ends the Program sequence. PGM can be set only under User mode read (ERS is low and UT0.AIE is low). PGM can be cleared by the user only when EHV is low and DONE is high. PGM is cleared on reset. 0: Flash is not executing a Program sequence. 1: Flash is executing a Program sequence.
28	**PSUS:** *Program SUSpend* (Read/Write) Write this bit has no effect, but the written data can be read back.
29	**ERS:** *ERaSe* (Read/Write) ERS is used to setup the flash module for an Erase operation. A 0 to 1 transition of ERS initiates an Erase sequence. A 1 to 0 transition of ERS ends the Erase sequence. ERS can be set only under User mode read (PGM is low and UT0.AIE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0: Flash is not executing an Erase sequence. 1: Flash is executing an Erase sequence.

Table 17-8. MCR field descriptions (continued)

Field	Description
30	**ESUS:** *Erase SUSpend* (Read/Write) ESUS is used to indicate that the flash module is in Erase Suspend or in the process of entering a Suspend state. The flash module is in Erase Suspend when ESUS=1 and DONE=1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in Erase Suspend. The flash module enters Suspend within t_{ESUS} of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV=1 starts the sequence which clears DONE and returns the Module to Erase. The flash module cannot exit Erase Suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0: Erase sequence is not suspended. 1: Erase sequence is suspended.
31	**EHV:** *Enable High Voltage* (Read/Write) The EHV bit enables the flash module for a high voltage Program/Erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a Program/Erase sequence. EHV may be set under one of the following conditions: Erase (ERS=1, ESUS=0, UT0.AIE=0) Program (ERS=0, ESUS=0, PGM=1, UT0.AIE=0) In normal operation, a 1 to 0 transition of EHV with DONE high and ESUS low terminates the current Program/Erase high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the eventual Suspend bit low. An abort causes the value of PEG to be cleared, indicating a failing Program/Erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. A suspended operation cannot be aborted. Aborting a high voltage operation will leave the flash module addresses in an undeterminate data state. This may be recovered by executing an Erase on the affected blocks. EHV may be written during Suspend. EHV must be high to exit Suspend. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low. 0: Flash is not enabled to perform an high voltage operation. 1: Flash is enabled to perform an high voltage operation.

Table 17-9. Array Space Size

SIZE2-0	Array Space Size
000	128KB
001	256KB
010	512KB
011	Reserved (1024KB)
100	Reserved (1536KB)
101	Reserved (2048KB)
110	64KB
111	Reserved

Table 17-10. Low Address Space configuration

LAS2-0	Low Address Space Sectorization
000	Reserved
001	2×128 KB
010	32 KB + 2×16 KB + 2×32 KB + 128 KB
011	Reserved
100	Reserved
101	Reserved
110	4 x 16KB
111	Reserved

Table 17-11. Mid Address Space configuration

MAS	Mid Address Space Sectorization
0	2×128 KB or 0 KB
1	Reserved

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit by bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The flash module does not allow the user to write bits simultaneously which would put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in the following table.

Table 17-12. MCR Bits Set/Clear Priority Levels

Priority Level	MCR Bits
1	ERS
2	PGM
3	EHV
4	ESUS

If the user attempts to write two or more MCR bits simultaneously then only the bit with the lowest priority level will be written.

17.2.6.2 Low/Mid Address Space Block Locking Register (LML)

Address Offset: 0x0004

Reset value: 0x00XXXXXX, initially determined by NVLML value from test sector.

17.2.6.3 Non-Volatile Low/Mid Address Space Block Locking Register (NVLML)

Address Offset: 0x403DE8 Delivery value: 0xFFFFFFFF

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LME	0	0	0	0	0	0	0	0	0	0	TSLK	0	0	MLK1	MLK0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/X	r/0	r/0	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LLK15	LLK14	LLK13	LLK12	LLK11	LLK10	LLK9	LLK8	LLK7	LLK6	LLK5	LLK4	LLK3	LLK2	LLK1	LLK0
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-4. Non-Volatile Low/Mid Address Space Block Locking Register (NVLML)

The Low/Mid Address Space Block Locking register provides a means to protect blocks from being modified. These bits, along with bits in the SLL register, determine if the block is locked from Program or Erase. An “OR” of LML and SLL determine the final lock status.

The LML register has a related Non-volatile Low/Mid Address Space Block Locking register located in Test flash that contains the default reset value for LML: The NVLML register is read during the reset phase of the flash module and loaded into the LML.

The NVLML register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 17-13. LML field descriptions

Field	Description
0	**LME**: *Low/Mid Address Space Block Enable* (Read Only) This bit is used to enable the Lock registers (TSLK, MLK1-0 and LLK15-0) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the LME bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For LME the password 0xA1A11111 must be written to the LML register. 0: Low Address Locks are disabled: TSLK, MLK1-0 and LLK15-0 cannot be written. 1: Low Address Locks are enabled: TSLK, MLK1-0 and LLK15-0 can be written.
1:10	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-13. LML field descriptions (continued)

Field	Description
11	**TSLK**: *Test/Shadow address space block LocK* (Read/Write) This bit is used to lock the block of Test and Shadow Address Space from Program and Erase (Erase is any case forbidden for Test block). A value of 1 in the TSLK register signifies that the Test/Shadow block is locked for Program and Erase. A value of 0 in the TSLK register signifies that the Test/Shadow block is available to receive Program and Erase pulses. The TSLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the TSLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the TSLK register. The TSLK bit may be written as a register. Reset will cause the bit to go back to its Test flash block value. The default value of the TSLK bit (assuming erased fuses) would be locked. TSLK is not writable unless LME is high. 0: Test/Shadow Address Space Block is unlocked and can be modified (if also SLL.STSLK=0). 1: Test/Shadow Address Space Block is locked and cannot be modified.
12:13	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
14:15	**MLK1-0**: *Mid address space block LocK 1-0*(Read/Write) These bits are used to lock the blocks of Mid Address Space from Program and Erase. For Code flash 0, MLK1-0 are related to sectors B0F7-6, respectively. For Code flash 1, MLK1-0 are related to sectors B2F3-2, respectively. A value of 1 in a bit of the MLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the MLK register signifies that the corresponding block is available to receive Program and Erase pulses. The MLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the MLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the MLK registers. The MLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the MLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the MLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. MLK is not writable unless LME is high. 0: Mid Address Space Block is unlocked and can be modified (if also SLL.SMLK=0). 1: Mid Address Space Block is locked and cannot be modified.

Table 17-13. LML field descriptions (continued)

Field	Description
16:31	**LLK15-0**: *Low address space block LocK 15-0* (Read/Write) These bits are used to lock the blocks of Low Address Space from Program and Erase. For Code flash 0, LLK5-0 are related to sectors B0F5-0, respectively. LLK15-6 are not used for Code flash 0. For Code flash 1, LLK1-0 are related to sectors B2F1-0, respectively. LLK15-2 are not used for Code flash 1. A value of 1 in a bit of the LLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the LLK register signifies that the corresponding block is available to receive Program and Erase pulses. The LLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the LLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the LLK registers. The LLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the LLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the LLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In Code flash 0 bits LLK15-6 are read-only and locked at 1. In Code flash 1 bits LLK15-2 are read-only and locked at 1. LLK is not writable unless LME is high. 0: Low Address Space Block is unlocked and can be modified (if also SLL.SLK=0). 1: Low Address Space Block is locked and cannot be modified.

17.2.6.4 High Address Space Block Locking Register (HBL)

Address Offset: 0x0008

Reset value: 0x000000XX, initially determined by NVHBL, located in test sector.

17.2.6.5 Non-Volatile High Address Space Block Locking Register (NVHBL)

Address Offset: 0x403DF0 Delivery value: 0xFFFFFFFF

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HBE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	0	HLK5	HLK4	HLK3	HLK2	HLK1	HLK0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-5. Non-volatile High address space Block Locking register (NVHBL)

The High Address Space Block Locking register provides a means to protect blocks from being modified. The HBL register has a related Non-Volatile High Address Space Block Locking register located in Test flash that contains the default reset value for HBL: The NVHBL register is read during the reset phase of the flash module and loaded into the HBL.

The NVHBL register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 17-14. HBL field descriptions

Field	Description
0	*High address space Block Enable* (Read Only) This bit is used to enable the Lock registers (HLK5-0) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the HBE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For HBE the password 0xB2B22222 must be written to the HBL register. 0: High Address Locks are disabled: HLK5-0 cannot be written. 1: High Address Locks are enabled: HLK5-0 can be written.
1:25	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
26:31	**HLK5-0**: *High address space block LocK 5-0* (Read/Write) These bits are used to lock the blocks of High Address Space from Program and Erase. All the HLK5-0 are not used for both Code flash 0 and 1 that are all mapped in mid and low address space. A value of 1 in a bit of the HLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the HLK register signifies that the corresponding block is available to receive Program and Erase pulses.The HLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the HLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the HLK registers. The HLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the HLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the HLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In both Code flash 0 and 1, bits HLK5-0 are read-only and locked at 1. HLK is not writable unless HBE is high. 0: High Address Space Block is unlocked and can be modified. 1: High Address Space Block is locked and cannot be modified.

17.2.6.6 Secondary Low/Mid Address Space Block Locking Register (SLL)

Address Offset: 0x000C

Reset value: 0x00XXXXXX, initially determined by NVSLL, located in test sector.

17.2.6.7 Non-volatile Secondary Low/Mid Address Space Block Locking Register (NVSLL)

Address Offset: 0x403DF8 Delivery value: 0xFFFFFFFF

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLE	0	0	0	0	0	0	0	0	0	0	STSLK	0	0	SMK1	SMK0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/X	r/0	r/0	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SLK15	SLK14	SLK13	SLK12	SLK11	SLK10	SLK9	SLK8	SLK7	SLK6	SLK5	SLK4	SLK3	SLK2	SLK1	SLK0
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-6. Non-Volatile Secondary Low/Mid Address Space Block Locking Register (NVSLL)

The Secondary Low/Mid Address Space Block Locking register provides an alternative means to protect blocks from being modified. These bits, along with bits in the LML register, determine if the block is locked from Program or Erase. An “OR” of LML and SLL determine the final lock status.

The SLL register has a related Non-volatile Secondary Low/Mid Address Space Block Locking register located in Test flash that contains the default reset value for SLL: The NVSLL register is read during the reset phase of the flash module and loaded into the SLL.

The NVSLL register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 17-15. SLL field descriptions

Field	Description
0	**SLE**: *Secondary Low/mid address space block Enable* (Read Only) This bit is used to enable the Lock registers (STSLK, SMK1-0 and SLK15-0) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the SLE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For SLE the password 0xC3C33333 must be written to the SLL register. 0: Secondary Low/Mid Address Locks are disabled: STSLK, SMK1-0 and SLK15-0 cannot be written. 1: Secondary Low/Mid Address Locks are enabled: STSLK, SMK1-0 and SLK15-0 can be written.
1:10	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-15. SLL field descriptions (continued)

Field	Description
11	**STSLK**: *Secondary Test/Shadow address space block LocK* (Read/Write) This bit is used as an alternate means to lock the block of Test and Shadow Address Space from Program and Erase (Erase is any case forbidden for Test block). A value of 1 in the STSLK register signifies that the Test/Shadow block is locked for Program and Erase. A value of 0 in the STSLK register signifies that the Test/Shadow block is available to receive Program and Erase pulses. The STSLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the STSLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the STSLK register. The STSLK bit may be written as a register. Reset will cause the bit to go back to its Test flash block value. The default value of the STSLK bit (assuming erased fuses) would be locked. STSLK is not writable unless SLE is high. 0: Test/Shadow Address Space Block is unlocked and can be modified (if also LML.TSLK=0). 1: Test/Shadow Address Space Block is locked and cannot be modified.
12:13	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
14:15	**SMK1-0**: *Secondary Mid address space block locK 1-0* (Read/Write) These bits are used as an alternate means to lock the blocks of Mid Address Space from Program and Erase. For Code flash 0, SMK1-0 are related to sectors B0F7-6, respectively. For Code flash 1, SMK1-0 are related to sectors B2F3-2, respectively. A value of 1 in a bit of the SMK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the SMK register signifies that the corresponding block is available to receive Program and Erase pulses. The SMK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the SMK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the SMK registers. The SMK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the SMK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the SMK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. SMK is not writable unless SLE is high. 0: Mid Address Space Block is unlocked and can be modified (if also LML.MLK=0). 1: Mid Address Space Block is locked and cannot be modified.

Table 17-15. SLL field descriptions (continued)

Field	Description
16:31	**SLK15-0**: *Secondary Low address space block locK 15-0* (Read/Write) These bits are used as an alternate means to lock the blocks of Low Address Space from Program and Erase. For Code flash 0, SLK5-0 are related to sectors B0F5-0, respectively. For Code flash 1, SLK1-0 are related to sectors B2F1-0, respectively. A value of 1 in a bit of the SLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the SLK register signifies that the corresponding block is available to receive Program and Erase pulses. The SLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the SLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the SLK registers. The SLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the SLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the SLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In Code flash 0 bits SLK15-6 are read-only and locked at 1. In Code flash 1, bits SLK15-2 are read-only and locked at 1. SLK is not writable unless SLE is high. 0: Low Address Space Block is unlocked and can be modified (if also LML.LLK=0). 1: Low Address Space Block is locked and cannot be modified.

17.2.6.8 Low/Mid aDdress Space Block Select Register (LMS)

Address Offset: 0x00010 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	MSL1	MSL0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LSL15	LSL14	LSL13	LSL12	LSL11	LSL10	LSL9	LSL8	LSL7	LSL6	LSL5	LSL4	LSL3	LSL2	LSL1	LSL0
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-7. Low/Mid Address Space Block Select Register (LMS)

The Low/Mid Address Space Block Select register provides a means to select blocks to be operated on during erase.

Table 17-16. LMS field descriptions

Field	Description
0:13	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-16. LMS field descriptions (continued)

Field	Description
14:15	**MSL1-0**: *Mid address space block SeLect 1-0* (Read/Write) A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected. For Code flash 0, MSL1-0 are related to sectors B0F7-6, respectively. For Code flash 1, MSL1-0 are related to sectors B2F3-2, respectively. The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended. In the event that blocks are not present (due to configuration or total memory size), the corresponding MSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect. 0: Mid Address Space Block is unselected for Erase. 1: Mid Address Space Block is selected for Erase.
16:31	**LSL15-0**: *Low address space block SeLect 15-0* (Read/Write) A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected. For Code flash 0, LSL5-0 are related to sectors B0F5-0, respectively. For Code flash 1, LSL1-0 are related to sectors B2F1-0, respectively. The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended. In the event that blocks are not present (due to configuration or total memory size), the corresponding LSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect. In Code flash 0, bits LSL15-6 are read-only and locked at 0. In Code flash 1 bits LSL15-2 are read-only and locked at 0. 0: Low Address Space Block is unselected for Erase. 1: Low Address Space Block is selected for Erase.

17.2.6.9 High Address Space Block Select Register (HBS)

Address Offset: 0x00014 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	0	HSL5	HSL4	HSL3	HSL2	HSL1	HSL0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-8. High Address Space Block Select Register (HBS)

The High Address Space Block Select register provides a means to select blocks to be operated on during erase.

Table 17-17. HBS field descriptions

Field	Description
0:25	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
26:31	**HSL5-0**: *High address space block SeLect 5-0* (Read/Write) A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected. All the HSL5-0 are not used for both Code flash 0 and 1 that are all mapped in mid and low address space. The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended. In the event that blocks are not present (due to configuration or total memory size), the corresponding HSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect. In both Code flash 0 and 1, bits HSL5-0 are read-only and locked at 0. 0: High Address Space Block is unselected for Erase. 1: High Address Space Block is selected for Erase.

17.2.6.10 Address Register (ADR)

Address Offset: 0x00018 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	AD22	AD21	AD20	AD19	AD18	AD17	AD16
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	0	0	0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

Figure 17-9. Address Register (ADR)

The Address Register provides the first failing address in the event module failures (ECC, RWW or FPEC) or the first address at which a ECC single error correction occurs.

Table 17-18. ADR field descriptions

Field	Description
0:8	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-18. ADR field descriptions (continued)

Field	Description
9:28	**AD22-3**: *ADdress 22-3* (Read Only) The Address Register provides the first failing address in the event of ECC error (MCR.EER set) or the first failing address in the event of RWW error (MCR.RWE set), or the address of a failure that may have occurred in a FPEC operation (MCR.PEG cleared). The Address Register provides also the first address at which a ECC single error correction occurs (MCR.EDC set), if the SoC is configured to show this feature. The ECC double error detection takes the highest priority, followed by the RWW error, the FPEC error and the ECC single error correction. When accessed ADR will provide the address related to the first event occurred with the highest priority. The priorities between these 4 possible events is summarized in the following table. This address is always a Double Word address that selects 64 bits. In case of a simultaneous ECC Double Error Detection on both Double Words of the same page, bit AD3 will output 0. The same is valid for a simultaneous ECC Single Error Correction on both Double Words of the same page. In User mode, the Address Register is read only. **Note:** An erroneous update of the Address register (ADR) occurs whenever there is a sequence of three or more events affecting the ADR (ECC single or double bit errors or RWW error) and both the following conditions apply: — The priorities are ordered in such a way that only the first event should update ADR. — The last event, although it does not update ADR, sets the Read While Write Event Error (RWE) bit or the ECC Data Correction (EDC) bit in the Module Configuration Register (MCR). For this case, the ADR is wrongly updated with the address related to one of the intervening events. Example: If a sequence of two double-bit ECC errors is followed by a single-bit correction without clearing the ECC Event Error flag (EER) in the MCR, then the value found in ADR after the single-bit correction event is the one related to the second double-bit error (instead of the first one, as specified). See also Table 17-49
29:31	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-19. ADR content: priority list

Priority Level	Error Flag	ADR content
1	MCR.EER = 1	Address of first ECC Double Error
2	MCR.RWE = 1	Address of first RWW Error
3	MCR.PEG = 0	Address of first FPEC Error
4	MCR.EDC = 1	Address of first ECC Single Error Correction

17.2.6.11 Bus Interface Unit 0 register (BIU0)

Address Offset: 0x0001C Reset value: 0xXXXXXXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BI031	BI030	BI029	BI028	BI027	BI026	BI025	BI024	BI023	BI022	BI021	BI020	BI019	BI018	BI017	BI016
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BI015	BI014	BI013	BI012	BI011	BI010	BI009	BI008	BI007	BI006	BI005	BI004	BI003	BI002	BI001	BI000
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-10. Bus Interface Unit 0 register (BIU0)

The Bus Interface Unit 0 Register provides a mean for BIU specific information, or BIU configuration information to be stored. Please see Section 17.4.3.2.1, Platform Flash Configuration Register 0 (PFCR0), for more information about register description.

This register is present only in Code flash 0.

Table 17-20. BIU0 field descriptions

Field	Description
0:31	**BI031-00**: *Bus Interface unit 0 31-00* (Read/Write) The writability of the bits in this register can be locked.

NOTE

On this device, BIU0 and PFCR0 are the same register. Your software may refer to either register. For clarity, however, it is recommended that you use PFCR0.

17.2.6.12 Bus Interface Unit 1 register (BIU1)

Address Offset: 0x00020 Reset value: 0xXXXXXXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BI131	BI130	BI129	BI128	BI127	BI126	BI125	BI124	BI123	BI122	BI121	BI120	BI119	BI118	BI117	BI116
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BI115	BI114	BI113	BI112	BI111	BI110	BI109	BI108	BI107	BI106	BI105	BI104	BI103	BI102	BI101	BI100
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-11. Bus Interface Unit 1 register (BIU1)

The Bus Interface Unit 1 Register provides a mean for BIU specific information, or BIU configuration information to be stored. Please see Section 17.4.3.2.2, Platform Flash Configuration Register 1 (PFCR1), for more information about register description.

This register is present only in Code flash 0.

Table 17-21. BIU1 field descriptions

Field	Description
0:31	**BI131-00**: *Bus Interface unit 1 31-00* (Read/Write) The writability of the bits in this register can be locked. The use of this bus is SoC specific.

NOTE

On this device, BIU1 and PFCR1 are the same register. Your software may refer to either register. For clarity, however, it is recommended that you use PFCR1.

17.2.6.13 Bus Interface Unit 2 register (BIU2)

Address Offset: 0x00024

Reset value: 0xXXXXXXXX

17.2.6.14 Non-volatile Bus Interface Unit 2 register (NVBIU2)

Address Offset: 0x203E00 Delivery value: 0xXXXXXXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BI231	BI230	BI229	BI228	BI227	BI226	BI225	BI224	BI223	BI222	BI221	BI220	BI219	BI218	BI217	BI216
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BI215	BI214	BI213	BI212	BI211	BI210	BI209	BI208	BI207	BI206	BI205	BI204	BI203	BI202	BI201	BI200
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-12. Non-volatile Bus Interface Unit 2 register (NVBIU2)

The Bus Interface Unit 2 Register provides a mean for BIU specific information, or BIU configuration information to be stored. Please see Section 17.4.3.2.3, Platform Flash Access Protection Register (PFAPR), for more information about register description.

This register is present only in Code flash 0.

The BIU2 register has a related Non-volatile Bus Interface Unit 2 register located in Shadow block that contains the default reset value for BIU2: The NVBIU2 register is read during the reset phase of the flash module and loaded into the BIU2.

The NVBIU2 register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 17-22. BIU2 field descriptions

Field	Description
0:31	**BI231-00**: *Bus Interface unit 2 31-00* (Read/Write) The BI231-00 generic registers are reset based on the information stored in NVBIU2. The writability of the bits in this register can be locked. The use of this bus is SoC specific.

NOTE

On this device, BIU2 and PFAPR are the same register. Your software may refer to either register. For clarity, however, it is recommended that you use PFAPR.

17.2.6.15 User Test 0 register (UT0)

Address Offset: 0x0003C Reset value: 0x00000001

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UTE	0	0	0	0	0	0	0	DSI7	DSI6	DSI5	DSI4	DSI3	DSI2	DSI1	DSI0
rw/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	X	MRE	MRV	EIE	AIS	AIE	AID
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	r/1

Figure 17-13. User Test 0 register (UT0)

The User Test feature gives the user of the flash module the ability to perform test features on the flash memory. The User Test 0 Register allows to control the way in which the flash memory content check is done.

Bits MRE, MRV, AIS, EIE and DSI7-0 of the User Test 0 Register are not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-23. UT0 field descriptions

Field	Description
0	**UTE**: *User Test Enable* (Read/Clear) This status bit gives indication when User Test is enabled. All bits in UT0-2 and UMISR0-4 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. For UTE the password 0xF9F99999 must be written to the UT0 register.
1:7	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-23. UT0 field descriptions (continued)

Field	Description
8:15	**DSI7-0**: *Data Syndrome Input 7-0* (Read/Write) These bits represent the input of Syndrome bits of ECC logic used in the ECC Logic Check. The DSI7-0 correspond to the 8 syndrome bits on a double word. These bits are not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: The syndrome bit is forced at 0. 1: The syndrome bit is forced at 1.
16:24	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
25	*Reserved* (Read/Write). This bit can be written and its value can be read back, but there is no function associated. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.
26	**MRE**: *Margin Read Enable* (Read/Write) MRE enables margin reads to be done. This bit, combined with MRV, enables regular user mode reads to be replaced by margin reads. Margin reads are only active during Array Integrity Checks; Normal user reads are not affected by MRE. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: Margin reads are not enabled, all reads are User mode reads. 1: Margin reads are enabled.
27	**MRV**: *Margin Read Value* (Read/Write) If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: Zero's (programmed) margin reads are requested (if MRE=1). 1: One's (erased) margin reads are requested (if MRE=1).
28	**EIE**: *ECC data Input Enable* (Read/Write) EIE enables the ECC Logic Check operation to be done. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: ECC Logic Check is not enabled. 1: ECC Logic Check is enabled.
29	**AIS**: *Array Integrity Sequence* (Read/Write) AIS determines the address sequence to be used during array integrity checks or Margin mode. The default sequence (AIS=0) is meant to replicate sequences normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS=1) is just logically sequential. It should be noted that the time to run a sequential sequence is significantly shorter than the time to run the proprietary sequence. Only the sequential mode is allowed in Margin mode. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: Array Integrity sequence is proprietary sequence. 1: Array Integrity sequence or Margin mode sequence is sequential.

Table 17-23. UT0 field descriptions (continued)

Field	Description
31	**AIE**: *Array Integrity Enable* (Read/Write) AIE set to 1 starts the Array Integrity Check done on all selected and unlocked blocks. The pattern is selected by AIS, and the MISR (UMISR0-4) can be checked after the operation is complete, to determine if a correct signature is obtained. AIE can be set only if MCR.ERS, MCR.PGM and MCR.EHV are all low. 0: Array Integrity Checks, Margin mode, and ECC Logic Checks are not enabled. 1: Array Integrity Checks, Margin mode, and ECC Logic Checks are enabled.
31	**AID**: *Array Integrity Done* (Read Only) AID will be cleared upon an Array Integrity Check being enabled (to signify the operation is on-going). Once completed, AID will be set to indicate that the Array Integrity Check is complete. At this time the MISR (UMISR0-4) can be checked. 0: Array Integrity Check is on-going. 1: Array Integrity Check is done.

17.2.6.16 User Test 1 register (UT1)

Address Offset: 0x00040 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DAI31	DAI30	DAI29	DAI28	DAI27	DAI26	DAI25	DAI24	DAI23	DAI22	DAI21	DAI20	DAI19	DAI18	DAI17	DAI16
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DAI15	DAI14	DAI13	DAI12	DAI11	DAI10	DAI09	DAI08	DAI07	DAI06	DAI05	DAI04	DAI03	DAI02	DAI01	DAI00
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-14. User Test 1 register (UT1)

The User Test 1 Register allows to enable the checks on the ECC logic related to the 32 LSB of the Double Word.

The User Test 1 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-24. UT1 field descriptions

Field	Description
0:31	**DAI31-00**: *Data Array Input 31-0* (Read/Write) These bits represent the input of even word of ECC logic used in the ECC Logic Check. The DAI31-00 correspond to the 32 array bits representing Word 0 within the double word. 0: The array bit is forced at 0. 1: The array bit is forced at 1.

17.2.6.17 User Test 2 register (UT2)

Address Offset: 0x00044 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DAI63	DAI62	DAI61	DAI60	DAI59	DAI58	DAI57	DAI56	DAI55	DAI54	DAI53	DAI52	DAI51	DAI50	DAI49	DAI48
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DAI47	DAI46	DAI45	DAI44	DAI43	DAI42	DAI41	DAI40	DAI39	DAI38	DAI37	DAI36	DAI35	DAI34	DAI33	DAI32
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-15. User Test 2 register (UT2)

The User Test 2 Register allows to enable the checks on the ECC logic related to the 32 MSB of the Double Word.

The User Test 2 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-25. UT2 field descriptions

Field	Description
0:31	**DAI63-32**: *Data Array Input 63-32* (Read/Write) These bits represent the input of odd word of ECC logic used in the ECC Logic Check. The DAI63-32 correspond to the 32 array bits representing Word 1 within the double word. 0: The array bit is forced at 0. 1: The array bit is forced at 1.

17.2.6.18 User Multiple Input Signature Register 0 (UMISR0)

Address Offset: 0x00048 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS031	MS030	MS029	MS028	MS027	MS026	MS025	MS024	MS023	MS022	MS021	MS020	MS019	MS018	MS017	MS016
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS015	MS014	MS013	MS012	MS011	MS010	MS009	MS008	MS007	MS006	MS005	MS004	MS003	MS002	MS001	MS000
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-16. User Multiple Input Signature Register 0 (UMISR0)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.

The User Multiple Input Signature Register 0 represents the bits 31-0 of the whole 144 bits word (2 Double Words including ECC).

The UMISR0 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-26. UMSIR0 field descriptions

Field	Description
0:31	**MS031-000**: *Multiple input Signature 031-000* (Read/Write) These bits represent the MISR value obtained accumulating the bits 31-0 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR0 register.

17.2.6.19 User Multiple Input Signature Register 1 (UMISR1)

Address Offset: 0x0004C

Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS063	MS062	MS061	MS060	MS059	MS058	MS057	MS056	MS055	MS054	MS053	MS052	MS051	MS050	MS049	MS048
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS047	MS046	MS045	MS044	MS043	MS042	MS041	MS040	MS039	MS038	MS037	MS036	MS035	MS034	MS033	MS032
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-17. User Multiple Input Signature Register 1 (UMISR1)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.

The User Multiple Input Signature Register 1 represents the bits 63-32 of the whole 144 bits word (2 Double Words including ECC).

The UMISR1 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-27. UMISR1 field descriptions

Field	Description
0:31	**MS063-032**: *Multiple input Signature 063-032* (Read/Write) These bits represent the MISR value obtained accumulating the bits 63-32 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR1 register.

17.2.6.20 User Multiple Input Signature Register 2 (UMISR2)

Address Offset: 0x00050 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS095	MS094	MS093	MS092	MS091	MS090	MS089	MS088	MS087	MS086	MS085	MS084	MS083	MS082	MS081	MS080
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS079	MS078	MS077	MS076	MS075	MS074	MS073	MS072	MS071	MS070	MS069	MS068	MS067	MS066	MS065	MS064
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-18. User Multiple Input Signature Register 2 (UMISR2)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.
The User Multiple Input Signature Register 2 represents the bits 95-64 of the whole 144 bits word (2 Double Words including ECC).

The UMISR2 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-28. UMISR2 field descriptions

Field	Description
0:31	**MS095-064**: *Multiple input Signature 095-064* (Read/Write) These bits represent the MISR value obtained accumulating the bits 95-64 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR2 register.

17.2.6.21 User Multiple Input Signature Register 3 (UMISR3)

Address Offset: 0x00054 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS127	MS126	MS125	MS124	MS123	MS122	MS121	MS120	MS119	MS118	MS117	MS116	MS115	MS114	MS113	MS112
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS111	MS110	MS109	MS108	MS107	MS106	MS105	MS104	MS103	MS102	MS101	MS100	MS099	MS098	MS097	MS096
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-19. User Multiple Input Signature Register 3 (UMISR3)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.

The User Multiple Input Signature Register 3 represents the bits 127-96 of the whole 144 bits word (2 Double Words including ECC).

The UMISR3 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-29. UMISR3 field descriptions

Field	Description
0:31	**MS127-096**: *Multiple input Signature 127-096* (Read/Write) These bits represent the MISR value obtained accumulating the bits 127-96 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR3 register.

17.2.6.22 User Multiple Input Signature Register 4 (UMISR4)

Address Offset: 0x00058 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS159	MS158	MS157	MS156	MS155	MS154	MS153	MS152	MS151	MS150	MS149	MS148	MS147	MS146	MS145	MS144
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS143	MS142	MS141	MS140	MS139	MS138	MS137	MS136	MS135	MS134	MS133	MS132	MS131	MS130	MS129	MS128
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-20. User Multiple Input Signature Register 4 (UMISR4)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.

The User Multiple Input Signature Register 4 represents the ECC bits of the whole 144 bits word (2 Double Words including ECC): bits 8-15 are ECC bits for the odd Double Word and bits 24-31 are the ECC bits for the even Double Word; bits 4-5 and 20-21 of MISR are respectively the double and single ECC error detection for odd and even Double Word.

The UMISR4 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-30. UMISR4 field descriptions

Field	Description
0:31	**MS159-128**: *Multiple input Signature 159-128* (Read/Write) These bits represent the MISR value obtained accumulating: the 8 ECC bits for the even Double Word (on MS135-128); the single ECC error detection for even Double Word (on MS138); the double ECC error detection for even Double Word (on MS139); the 8 ECC bits for the odd Double Word (on MS151-144); the single ECC error detection for odd Double Word (on MS154); the double ECC error detection for odd Double Word (on MS155). The MS can be seeded to any value by writing the UMISR4 register.

17.2.6.23 Non-volatile private censorship PassWord 0 register (NVPWD0)

Address Offset: 0x203DD8 Reset value: 0xFEEDFACE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PWD 31	PWD 30	PWD 29	PWD 28	PWD 27	PWD 26	PWD 25	PWD 24	PWD 23	PWD 22	PWD 21	PWD 20	PWD 19	PWD 18	PWD 17	PWD 16
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PWD 15	PWD 14	PWD 13	PWD 12	PWD 11	PWD 10	PWD 09	PWD 08	PWD 07	PWD 06	PWD 05	PWD 04	PWD 03	PWD 02	PWD 01	PWD 00
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-21. Non-Volatile Private Censorship Password 0 Register (NVPWD0)

The Non-Volatile Private Censorship Password 0 Register contains the 32 LSB of the Password used to validate the Censorship information contained in NVSCI0–1 registers.

Table 17-31. NVPWD0 field descriptions

Field	Description
0:31	**PWD31-00**: *PassWorD 31-00* (Read/Write) The PWD31-00 registers represent the 32 LSB of the Private Censorship Password.

17.2.6.24 Non-Volatile Private Censorship Password 1 Register (NVPWD1)

Address Offset: 0x203DDC Reset value: 0xCAFEBEEF

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PWD 63	PWD 62	PWD 61	PWD 60	PWD 59	PWD 58	PWD 57	PWD 56	PWD 55	PWD 54	PWD 53	PWD 52	PWD 51	PWD 50	PWD 49	PWD 48
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PWD 47	PWD 46	PWd45	PWD 44	PWD 43	PWD 42	PWD 41	PWD 40	PWD 39	PWD 38	PWD 37	PWD 36	PWD 35	PWD 34	PWD 33	PWD 32
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-22. Non-Volatile Private Censorship Password 1 Register (NVPWD1)

The Non-volatile Private Censorship Password 1 Register contains the 32 MSB of the Password used to validate the Censorship information contained in NVSCI0–1 registers.

Table 17-32. NVPWD1 field descriptions

Field	Description
0:31	**PWD63-32**: *PassWorD 63-32* (Read/Write) The PWD63-32 registers represent the 32 MSB of the Private Censorship Password.

17.2.6.25 Non-volatile System Censoring Information 0 register (NVSCI0)

Address Offset: 0x203DE0 Delivery value: 0x55AA55AA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SC15	SC14	SC13	SC12	SC11	SC10	SC9	SC8	SC7	SC6	SC5	SC4	SC3	SC2	SC1	SC0
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CW15	CW14	CW13	CW12	CW11	CW10	CW9	CW8	CW7	CW6	CW5	CW4	CW3	CW2	CW1	CW0
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-23. Non-volatile System Censoring Information 0 register (NVSCI0)

The Non-volatile System Censoring Information 0 register stores the 32 LSB of the Censorship Control Word of the SoC.

The NVSCI0 is a non-volatile register located in Shadow block: it is read during the reset phase of the flash module and the protection mechanisms are activated consequently.

The parts are delivered uncensored to the user.

Table 17-33. NVSCI0 field descriptions

Field	Description
0:15	**SC15-0**: *Serial Censorship control word 15-0* (Read/Write) These bits represent the 16 LSB of the Serial Censorship Control Word (SCCW). If SC15-0 = 0x55AA and NVSCI1 = NVSCI0 the Public Access is disabled. If SC15-0 ≠ 0x55AA or NVSCI1 ≠ NVSCI0 the Public Access is enabled.
16:31	**CW15-0**: *Censorship control Word 15-0* (Read/Write) These bits represent the 16 LSB of the Censorship Control Word (CCW). If CW15-0 = 0x55AA and NVSCI1 = NVSCI0 the Censored mode is disabled. If CW15-0 ≠ 0x55AA or NVSCI1 ≠ NVSCI0 the Censored mode is enabled.

17.2.6.26 Non-Volatile System Censoring Information 1 register (NVSCI1)

Address Offset: 0x203DE4 Delivery value: 0x55AA55AA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SC31	SC30	SC29	SC28	SC27	SC26	SC25	SC24	SC23	SC22	SC21	SC20	SC19	SC18	SC17	SC16
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CW31	CW30	CW29	CW28	CW27	CW26	CW25	CW24	CW23	CW22	CW21	CW20	CW19	CW18	CW17	CW16
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-24. Non-Volatile System Censoring Information 1 register (NVSCI1)

The Non-volatile System Censoring Information 1 register stores the 32 MSB of the Censorship Control Word of the SoC.

The NVSCI1 is a non-volatile register located in Shadow block: it is read during the reset phase of the flash module and the protection mechanisms are activated consequently.

The parts are delivered uncensored to the user.

Table 17-34. NVSCI1 field descriptions

Field	Description
0:15	**SC32-16**: *Serial Censorship control word 32-16* (Read/Write) These bits represent the 16 MSB of the Serial Censorship Control Word (SCCW). If SC15-0 = 0x55AA and NVSCI1 = NVSCI0 the Public Access is disabled. If SC15-0 ≠ 0x55AA or NVSCI1 ≠ NVSCI0 the Public Access is enabled.
16:31	**CW32-16**: *Censorship control Word 32-16* (Read/Write) These bits represent the 16 MSB of the Censorship Control Word (CCW). If CW15-0 = 0x55AA and NVSCI1 = NVSCI0 the Censored mode is disabled. If CW15-0 ≠ 0x55AA or NVSCI1 ≠ NVSCI0 the Censored mode is enabled.

17.2.6.27 Non-Volatile User Options register (NVUSRO)

Address Offset: 0x203E18 Delivery value: 0xXXXXXXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UO31	UO30	UO29	UO28	UO27	UO26	UO25	UO24	UO23	UO22	UO21	UO20	UO19	UO18	UO17	UO16
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
UO15	UO14	UO13	UO12	UO11	UO10	UO09	UO08	UO07	UO06	UO05	UO04	UO03	PAD3V5V	OSCILLATOR_MARGIN	WATCHDOG_EN
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-25. Non-volatile User Options register (NVUSRO)

The Non-volatile User Options Register contains configuration information for the user application.

The NVUSRO register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

The availability of this register is SoC dependent.

Table 17-35. NVUSRO field descriptions

Field	Description
0:28	**UO31-03**: *User Options 31-03* (Read/Write) The UO31-03 generic registers are reset based on the information stored in NVUSRO. The use of this bus is SoC specific.
2	**PAD3V5V** 0: High voltage supply is 5.0 V 1: High voltage supply is 3.3 V Default manufacturing value before flash initialization is 1 (3.3 V) which should ensure correct min slope for boundary scan.
1	**OSCILLATOR_MARGIN** 0: Low consumption configuration (4 MHz / 8 MHz) 1: High margin configuration (4/16 MHz) Default manufacturing value before flash initialization is 1
0	**WATCHDOG_EN** 0: Disable after reset 1: Enable after reset Default manufacturing value before flash initialization is 1. If the device undergoes a non-destructive reset, the behavior of SWT after reset will again be controlled by this field. Any change in the value of this field will take effect only after the device goes through a Phase 0 (destructive reset sequence).

17.2.7 Programming considerations

17.2.7.1 Modify operation

All the modify operations of the flash module are managed through the flash user registers interface.

All the sectors of the flash module belong to the same partition (bank), therefore when a Modify operation is active on some sectors no read access is possible on any other sector (Read-While-Modify is not supported).

During a flash modify operation, any attempt to read any flash memory location will output invalid data and bit RWE of MCR will be automatically set. This means that the flash module is not fetchable when a modify operation is active and these commands must be executed from another memory (internal RAM or another flash module).

If a reset occurs during a modify operation, the operation is suddenly terminated and the macrocell is reset to Read mode. The data integrity of the flash memory section where the modify operation has been terminated is not guaranteed — the interrupted flash memory modify operation must be repeated.

In general, each modify operation is started through a sequence of three steps:

- The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR (PGM or ERS) or UT0 (MRE or EIE).
- The second step is the definition of the operands: the address and the data for programming or the Sectors for erase or margin read.
- The third instruction is used to start the modify operation by setting EHV in MCR or AIE in UT0.

Once selected, but not yet started, one operation can be canceled by resetting the operation selection bit.

A summary of the available flash memory modify operations are shown in Table 17-36.

Table 17-36. Flash modify operations

Operation	Select bit	Operands	Start bit
Double Word Program	MCR.PGM	Address and Data by Interlock Writes	MCR.EHV
Sector Erase	MCR.ERS	LMS, HBS	MCR.EHV
Array Integrity Check	None	LMS, HBS	UT0.AIE
Margin Read	UT0.MRE	UT0.MRV + LMS, HBS	UT0.AIE
ECC Logic Check	UT0.EIE	UT0.DSI, UT1, UT2	UT0.AIE

Once bit MCR.EHV (or UT0.AIE) is set, all the operands can no more be modified until bit MCR.DONE (or UT0.AID) is high.

In general , each modify operation is completed through a sequence of four steps:

1. Wait for operation completion: wait for bit MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UMISR0-4 with expected value).
3. Switch-Off FPEC by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE/EIE).

If the device embeds more than one flash memory macrocell and a modify operation is ongoing on one of them, then it is forbidden to start any other modify operation on the other flash memory macrocells.

In the following, all the possible modify operations are described and some examples of the sequences needed to activate them are presented.

17.2.7.1.1 Double word program

A flash memory program sequence operates on any double word within the flash core.

As many as to two words within the double word may be altered in a single program operation.

Whenever you program, ECC bits also get programmed (unless the selected address belongs to a sector in which the ECC has been disabled in order to allow bit manipulation). ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be of 64 bits. The programming operation should completely fill selected ECC segments within the double word.

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

Addresses in locked/disabled blocks cannot be programmed.

The user may program the values in any or all of 2 words, of a double word, with a single program sequence.

Double word-bound words have addresses which differ only in address bit 2.

The Program operation consists of the following sequence of events:

1. Change the value in the MCR.PGM bit from 0 to 1.
2. Ensure the block that contains the address to be programmed is unlocked.
   Write the first address to be programmed with the program data.
   The flash module latches address bits (22:3) at this time.
   The flash module latches data written as well.
   This write is referred to as a program data interlock write. An interlock write may be as large as 64 bits, and as small as 32 bits (depending on the CPU bus).
3. If more than 1 word is to be programmed, write the additional address in the Double Word with data to be programmed. This is referred to as a program data write.
   The flash modules ignores address bits (22:3) for program data writes.
   The eventual unwritten data word default to 0xFFFFFFFF.
4. Write a logic 1 to the MCR.EHV bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR.DONE bit goes high.
6. Confirm MCR.PEG=1.
7. Write a logic 0 to the MCR.EHV bit.
8. If more addresses are to be programmed, return to step 2.

9. Write a logic 0 to the MCR.PGM bit to terminate the program operation.

Program may be initiated with the 0 to 1 transition of the MCR.PGM bit or by clearing the MCR.EHV bit at the end of a previous program.

The first write after a program is initiated determines the page address to be programmed. This first write is referred to as an interlock write. The interlock write determines if the shadow, test or normal array space will be programmed by causing MCR.PEAS to be set/cleared.

An interlock write must be performed before setting MCR.EHV. The user may terminate a program sequence by clearing MCR.PGM prior to setting MCR.EHV.

After the interlock write, additional writes only affect the data to be programmed at the word location determined by address bit 2. Unwritten locations default to a data value of 0xFFFFFFFF. If multiple writes are done to the same location the data for the last write is used in programming.

While MCR.DONE is low and MCR.EHV is high, the user may clear EHV, resulting in a program abort. A Program abort forces the Module to step 8 of the program sequence.

An aborted program will result in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The data space being operated on before the abort will contain indeterminate data. This may be recovered by repeating the same program instruction or executing an erase of the affected blocks.

Example 17-1. Double word program of data 0x55AA55AA at address 0x00AAA8 and data 0xAA55AA55 at address 0x00AAAC.

```
MCR              = 0x00000010;                  /* Set PGM in MCR: Select Operation */
(0x00AAA8)       = 0x55AA55AA;                  /* Latch Address and 32 LSB data */
(0x00AAAC)       = 0xAA55AA55;                  /* Latch 32 MSB data */
MCR              = 0x00000011;                  /* Set EHV in MCR: Operation Start */
do                                              /* Loop to wait for DONE=1 */
{ tmp            = MCR;                         /* Read MCR */
} while ( !(tmp & 0x00000400) );
status           = MCR & 0x00000200;            /* Check PEG flag */
MCR              = 0x00000010;                  /* Reset EHV in MCR: Operation End */
MCR              = 0x00000000;                  /* Reset PGM in MCR: Deselect Operation */
```

17.2.7.1.2 Sector erase

Erase changes the value stored in all bits of the selected block(s) to logic 1.

An erase sequence operates on any combination of blocks (sectors) in the low, mid or high address space, or the Shadow block (if available). The test block cannot be erased.

The erase sequence is fully automated within the flash memory. The user only needs to select the blocks to be erased and initiate the erase sequence.

Locked/disabled blocks cannot be erased.

If multiple blocks are selected for erase during an erase sequence, no specific operation order must be assumed.

The erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from 0 to 1.
2. Select the block(s) to be erased by writing 1s to the appropriate register(s) in LMS or HBS registers.
   If the Shadow block is to be erased, this step may be skipped, and LMS and HBS are ignored.
   Note that Lock and Select are independent. If a block is selected and locked, no erase will occur.
3. Write to any address in flash memory. This is referred to as an erase interlock write.
4. Write a logic 1 to the MCR.EHV bit to start the internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR.DONE bit goes high.
6. Confirm MCR.PEG=1.
7. Write a logic 0 to the MCR.EHV bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR.ERS bit to terminate the erase operation.

After setting MCR.ERS, one write, referred to as an interlock write, must be performed before MCR.EHV can be set to 1. Data words written during erase sequence interlock writes are ignored.

The user may terminate the erase sequence by clearing ERS before setting EHV.

An erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low.

An erase abort forces the Module to step 8 of the erase sequence.

An aborted erase will result in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The block(s) being operated on before the abort contain indeterminate data. This may be recovered by executing an erase on the affected blocks.

The user may not abort an erase sequence while in erase suspend.

Example 17-2. Erase of sectors B0F1 and B0F2.

```
MCR              = 0x00000004;             /* Set ERS in MCR: Select Operation */
LMS              = 0x00000006;             /* Set LSL2-1 in LMS: Select Sectors to erase */
(0x000000)       = 0xFFFFFFFF;             /* Latch a Flash Address with any data */
MCR              = 0x00000005;             /* Set EHV in MCR: Operation Start */
do                                         /* Loop to wait for DONE=1 */
{ tmp            = MCR;                    /* Read MCR */
} while ( !(tmp & 0x00000400) );
status           = MCR & 0x00000200;       /* Check PEG flag */
MCR              = 0x00000004;             /* Reset EHV in MCR: Operation End */
MCR              = 0x00000000;             /* Reset ERS in MCR: Deselect Operation */
```

Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core.

It is not possible to program or to erase during an erase suspend.

During erase suspend, all reads to blocks targeted for erase return indeterminate data.

An erase suspend can be initiated by changing the value of the MCR.ESUS bit from 0 to 1. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0 to 1 transition of MCR.ESUS causes the Module to start the sequence which places it in erase suspend.

The user must wait until MCR.DONE = 1 before the Module is suspended and further actions are attempted. MCR.DONE will go high no more than t_{ESUS} after MCR.ESUS is set to 1.

Once suspended, the array may be read. flash core reads while MCR.ESUS = 1 from the block(s) being erased return indeterminate data.

Example 17-3. Sector erase suspend

```
MCR              = 0x00000007;                  /* Set ESUS in MCR: Erase Suspend */
do                                              /* Loop to wait for DONE=1 */
{ tmp            = MCR;                         /* Read MCR */
} while ( !(tmp & 0x00000400) );
```

Notice that there is no need to clear MCR.EHV and MCR.ERS in order to perform reads during erase suspend.

The Erase sequence is resumed by writing a logic 0 to MCR.ESUS.

MCR.EHV must be set to 1 before MCR.ESUS can be cleared to resume the operation.

The Module continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Example 17-4. Sector Erase Resume

```
MCR              = 0x00000005;                  /* Reset ESUS in MCR: Erase Resume */
```

17.2.7.1.3 User Test mode

User Test mode is a mode that customers can put the flash module in, to do specific tests to check the integrity of the flash module.

Three kinds of Test can be performed:

- Array Integrity Self Check
- Margin mode read
- ECC Logic Check

The User Test mode is equivalent to a Modify operation: read accesses attempted by the user during User Test mode generates a Read-While-Write Error (RWE of MCR set).

It is not allowed to perform User Test operations on the Test and Shadow blocks.

Array Integrity self check

Array Integrity is checked using a predefined address sequence (proprietary), and this operation is executed on selected and unlocked blocks. Once the operation is completed, the results of the reads can be

checked by reading the MISR value (stored in UMISR0-4), to determine if an incorrect read, or ECC detection was noted.

The internal MISR calculator is a 32-bit register.

The 128-bit data, the 16 ECC data and the single and double ECC errors of the two Double Words are therefore captured by the MISR through 5 different read accesses at the same location.

The whole check is done through five complete scans of the memory address space:

1. The first pass will scan only bits 31-0 of each page.
2. The second pass will scan only bits 63-32 of each page.
3. The third pass will scan only bits 95-64 of each page.
4. The fourth pass will scan only bits 127-96 of each page.
5. The fifth pass will scan only the ECC bits (8 + 8) and the single and double ECC errors (2 + 2) of both Double Words of each page.

The 128-bit data and the 16 ECC data are sampled before the eventual ECC correction, while the single and double error flags are sampled after the ECC evaluation.

Only data from existing and unlocked locations are captured by the MISR.

The MISR can be seeded to any value by writing the UMISR0-4 registers.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Select the block(s) to be checked by writing 1s to the appropriate register(s) in LMS or HBS registers.
   Note that Lock and Select are independent. If a block is selected and locked, no Array Integrity Check will occur.
3. Set eventually UT0.AIS bit for a sequential addressing only.
4. Write a logic 1 to the UT0.AIE bit to start the Array Integrity Check.
5. Wait until the UT0.AID bit goes high.
6. Compare UMISR0-4 content with the expected result.
7. Write a logic 0 to the UT0.AIE bit.
8. If more blocks are to be checked, return to step 2.

It is recommended to leave UT0.AIS at 0 and use the proprietary address sequence that checks the read path more fully, although this sequence takes more time.

During the execution of the Array Integrity Check operation it is forbidden to modify the content of Block Select (LMS, HBS) and Lock (LML, SLL, HBL) registers, otherwise the MISR value can vary in an unpredictable way.

While UT0.AID is low and UT0.AIE is high, the user may clear AIE, resulting in a Array Integrity Check abort.

UT0.AID must be checked to know when the aborting command has completed.

Example 17-5. Array Integrity check of sectors B0F1 and B0F2.

```
UT0              = 0xF9F99999;             /* Set UTE in UT0: Enable User Test */
LMS              = 0x00000006;             /* Set LSL2-1 in LMS: Select Sectors */
UT0              = 0x80000002;             /* Set AIE in UT0: Operation Start */
do                                         /* Loop to wait for AID=1 */
{ tmp            = UT0;                    /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0            = UMISR0;                 /* Read UMISR0 content*/
data1            = UMISR1;                 /* Read UMISR1 content*/
data2            = UMISR2;                 /* Read UMISR2 content*/
data3            = UMISR3;                 /* Read UMISR3 content*/
data4            = UMISR4;                 /* Read UMISR4 content*/
UT0              = 0x00000000;             /* Reset UTE and AIE in UT0: Operation End */
```

Margin read

Margin read procedure (either Margin 0 or Margin 1), can be run on unlocked blocks in order to unbalance the Sense Amplifiers, respect to standard read conditions, so that all the read accesses reduce the margin vs. 0 (UT0.MRV = 0) or vs. 1 (UT0.MRV = 1). Locked sectors are ignored by MISR calculation and ECC flagging. The results of the margin reads can be checked comparing checksum value in UMISR0-4.

Since Margin reads are done at voltages that differ than the normal read voltage, lifetime expectancy of the flash memory macrocell is impacted by the execution of Margin reads.

Doing Margin reads repetitively results in degradation of the flash memory array, and shorten expected lifetime experienced at normal read levels.

For these reasons the margin read usage is allowed only in Factory mode, while it is forbidden to use it inside the user application.

In any case the charge losses detected through the Margin mode cannot be considered failures of the device and no failure analysis will be opened on them.

The Margin Read Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Select the block(s) to be checked by writing 1’s to the appropriate register(s) in LMS or HBS registers.
   Note that Lock and Select are independent. If a block is selected and locked, no Array Integrity Check will occur.
3. Set eventually UT0.AIS bit for a sequential addressing only.
4. Change the value in the UT0.MRE bit from 0 to 1.
5. Select the Margin level: UT0.MRV=0 for 0’s margin, UT0.MRV=1 for 1’s margin.
6. Write a logic 1 to the UT0.AIE bit to start the Margin Read Setup or skip to step 6 to terminate.
7. Wait until the UT0.AID bit goes high.
8. Compare UMISR0-4 content with the expected result.
9. Write a logic 0 to the UT0.AIE, UT0.MRE and UT0.MRV bits.
10. If more blocks are to be checked, return to step 2.

It is recommended to leave UT0.AIS at 1 and use the linear address sequence that takes less

time.

During the execution of the Margin mode operation it is forbidden to modify the content of Block Select (LMS, HBS) and Lock (LML, SLL, HBL) registers, otherwise the MISR value can vary in an unpredictable way.

The read accesses will be done with the addition of a proper number of Wait States to guarantee the correctness of the result.

While UT0.AID is low and UT0.AIE is high, the User may clear AIE, resulting in a Array Integrity Check abort.

UT0.AID must be checked to know when the aborting command has completed.

Example 17-6. Margin Read Check versus 1s

```
UT0 = 0xF9F99999;                   /* Set UTE in UT0: Enable User Test */
LMS = 0x00000006;                   /* Set LSL2-1 in LMS: Select Sectors */
UT0 = 0x80000004;                   /* Set AIS in UT0: Select Operation */
UT0 = 0x80000024;                   /* Set MRE in UT0: Select Operation */
UT0 = 0x80000034;                   /* Set MRV in UT0: Select Margin versus 1's */
UT0 = 0x80000036;                   /* Set AIE in UT0: Operation Start */
do                                  /* Loop to wait for AID=1 */
{ tmp = UT0;                        /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0 = UMISR0;                     /* Read UMISR0 content*/
data1 = UMISR1;                     /* Read UMISR1 content*/
data2 = UMISR2;                     /* Read UMISR2 content*/
data3 = UMISR3;                     /* Read UMISR3 content*/
data4 = UMISR4;                     /* Read UMISR4 content*/
UT0 = 0x80000034;                   /* Reset AIE in UT0: Operation End */
UT0 = 0x00000000;                   /* Reset UTE, MRE, MRV, AIS in UT0: Deselect Op. */
```

ECC logic check

ECC logic can be checked by forcing the input of ECC logic: The 64 bits of data and the 8 bits of ECC syndrome can be individually forced and they will drive simultaneously at the same value the ECC logic of the whole page (2 Double Words).

The results of the ECC Logic Check can be verified by reading the MISR value.

The ECC Logic Check operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Write in UT1.DAI31-0 and UT2.DAI63-32 the Double Word Input value.
3. Write in UT0.DSI7-0 the Syndrome Input value.
4. Select the ECC Logic Check: write a logic 1 to the UT0.EIE bit.
5. Write a logic 1 to the UT0.AIE bit to start the ECC Logic Check.
6. Wait until the UT0.AID bit goes high.
7. Compare UMISR0-4 content with the expected result.
8. Write a logic 0 to the UT0.AIE bit.

Notice that when UT0.AID is low UMISR0-4, UT1-2 and bits MRE, MRV, EIE, AIS and DSI7-0 of UT0 are not accessible: reading returns undeterminate data and write has no effect.

Example 17-7. ECC logic check

```
UT0              = 0xF9F99999;              /* Set UTE in UT0: Enable User Test */
UT1              = 0x55555555;              /* Set DAI31-0 in UT1: Even Word Input Data */
UT2              = 0xAAAAAAAA;              /* Set DAI63-32 in UT2: Odd Word Input Data */
UT0              = 0x80FF0000;              /* Set DSI7-0 in UT0: Syndrome Input Data */
UT0              = 0x80FF0008;              /* Set EIE in UT0: Select ECC Logic Check */
UT0              = 0x80FF000A;              /* Set AIE in UT0: Operation Start */
do                                          /* Loop to wait for AID=1 */
{ tmp            = UT0;                     /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0            = UMISR0;                  /* Read UMISR0 content (expected 0x55555555) */
data1            = UMISR1;                  /* Read UMISR1 content (expected 0xAAAAAAAA) */
data2            = UMISR2;                  /* Read UMISR2 content (expected 0x55555555) */
data3            = UMISR3;                  /* Read UMISR3 content (expected 0xAAAAAAAA) */
data4            = UMISR4;                  /* Read UMISR4 content (expected 0x00FF00FF) */
UT0              = 0x00000000;              /* Reset UTE, AIE and EIE in UT0: Operation End */
```

17.2.7.2 Error Correction Code (ECC)

The flash memory macrocell provides a method to improve the reliability of the data stored in flash memory: the usage of an error correction code (ECC). The word size is fixed at 64 bits.

At each double word of 64 bits, there are associated 8 ECC bits that are programmed in such a way to guarantee a Single Error Correction and a Double Error Detection (SEC-DED).

17.2.7.2.1 ECC algorithms

The flash memory macrocell supports one ECC algorithm: “All 1s No Error”. A modified Hamming code is used that ensures the all-erased state (that is, 0xFFFF.....FFFF) data is a valid state, and will not cause an ECC error. This allows the user to perform a blank check after a sector erase operation.

17.2.7.3 Protection strategy

Two kind of protections are available: Modify Protection to avoid unwanted program/erase in flash memory sectors and censored mode to avoid piracy.

17.2.7.3.1 Modify protection

The flash modify protection information is stored in non-volatile flash memory cells located in the Test flash. This information is read once during the flash initialization phase following the exit from Reset and is stored in volatile registers that act as actuators.

The reset state of all the Volatile Modify Protection Registers is the protected state.

All the non-volatile Modify Protection registers can be programmed through a normal Double Word Program operation at the related locations in Test flash.

The non-volatile Modify Protection registers cannot be erased.

- The Non-volatile Modify Protection Registers are physically located in Test flash their bits can be programmed to 0 only once and they can no more be restored to 1.
- The Volatile Modify Protection Registers are Read/Write registers which bits can be written at 0 or 1 by the user application.

A software mechanism is provided to independently lock/unlock each Low, Mid and High Address Space Block against program and erase.

Software locking is done through the LML (Low/Mid Address Space Block Lock Register) or HBL (High Address Space Block Lock Register) registers.

An alternate means to enable software locking for blocks of Low Address Space only is through the SLL (Secondary Low/Mid Address Space Block Lock Register).

All these registers have a non-volatile image stored in Test flash (NVLML, NVHBL, NVSLL), so that the locking information is kept on reset.

On delivery the Test flash non-volatile image is at all 1s, meaning all sectors are locked.

By programming the non-volatile locations in Test flash the selected sectors can be unlocked.

Being the Test flash One Time Programmable (that is, not erasable), once unlocked the sectors cannot be locked again.

Of course, on the contrary, all the volatile registers can be written at 0 or 1 at any time, therefore the user application can lock and unlock sectors when desired.

17.2.7.3.2 Censored mode

The Censored mode information is stored in non-volatile flash memory cells located in the Shadow block. This information is read once during the flash memory initialization phase following the exit from Reset and is stored in Volatile registers that act as actuators.

The reset state of all the volatile censored mode registers is the protected state.

All the non-volatile Censored mode registers can be programmed through a normal double word program operation at the related locations in the Shadow block.

The non-volatile Censored mode registers can be erased by erasing the Shadow block.

- The non-volatile Censored mode registers are physically located in the Shadow block their bits can be programmed to 0 and eventually restored to 1 by erasing the Shadow block.
- The Volatile Censored mode registers are registers not accessible by the user application.

The flash memory macrocell provides two levels of protection against piracy:

- If bits CW15-0 of NVSCI0 are programmed at 0x55AA and NVSC1 = NVSCI0 the Censored mode is disabled, while all the other possible values enable the Censored mode.
- If bits SC15-0 of NVSCI0 are programmed at 0x55AA and NVSC1 = NVSCI0 the Public Access is disabled, while all the other possible values enable the Public Access.

The parts are delivered to the user with Censored mode and Public Access disabled.

The flash memory ECC algorithm allows to modify the censorship status without erasing the Shadow block, as shown in Table 17-37.

Table 17-37. Bit manipulation: censorship management

Censored mode	Public access	NVSCI0	NVSCI1
Enabled	Enabled	0xFFFF_FFFF	0xFFFF_FFFF
Disabled	Enabled	0xFFFF_55AA	0xFFFF_55AA
Enabled	Disabled	0x55AA_FFFF	0x55AA_FFFF
Disabled	Disabled	0x55AA_55AA	0x55AA_55AA
Enabled	Disabled	0x55AA_0000	0x55AA_0000
Disabled	Enabled	0x0000_55AA	0x0000_55AA
Enabled	Enabled	0x0000_0000	0x0000_0000

17.3 Data flash memory

17.3.1 Introduction

The primary function of the data flash module is to serve as electrically programmable and erasable non-volatile memory.

Non-volatile memory may be used for instruction and/or data storage.

The module is a non-volatile solid-state silicon memory device consisting of blocks (called also sectors) of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The data flash module is arranged as two functional units: the flash core and the memory interface.

The flash core is composed of arrayed non-volatile storage elements, sense amplifiers, row decoders, column decoders and charge pumps. The arrayed storage elements in the flash core are subdivided into physically separate units referred to as blocks (or sectors).

The memory interface contains the registers and logic which control the operation of the flash core. The memory interface is also the interface between the flash module and a Bus Interface Unit (BIU) and contains the ECC logic and redundancy logic.

A BIU connects the flash module to a system bus, and contains all system-level customization required for the SoC application.

17.3.2 Main features

- High Read parallelism (128 bits)
- Error Correction Code (SEC-DED) to enhance Data Retention
- Double Word Program (64 bits)

- Sector Erase
- Single Bank: Read-While-Modify not available
- Erase Suspend available (Program Suspend not available)
- Software programmable Program/Erase Protection to avoid unwanted writings
- Censored mode against piracy
- Not usable as main Code Memory
- Shadow block not available
- “OTP” area in Test flash block

17.3.3 Block diagram

The flash memory macrocell contains one matrix module, composed of a single bank: Bank 0, normally used for Code storage. No Read-While-Modify operations are possible.

The Modify operations are managed by an embedded Flash Program/Erase Controller (FPEC). Commands to the FPEC are given through a user registers Interface.

The read data bus is 128 bits wide, while the flash memory registers are on a separate 32-bit bus.

The high voltages needed for Program/Erase operations are internally generated addressed in user memory map.

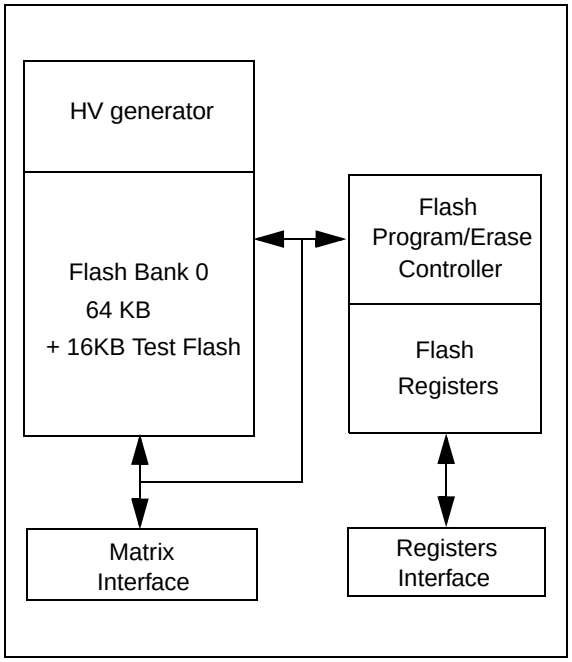


Figure 17-26. Flash macrocell structure

17.3.4 Functional description

17.3.4.1 Macrocell structure

The flash module is addressable by word (32 bits) or double word (64 bits) for program, and page (128 bits) for read. Reads done to the flash always return 128 bits, although read page buffering may be done in the platform BIU.

Each read of the flash module retrieves a page, or four consecutive words (128 bits) of information. The address for each word retrieved within a page differ from the other addresses in the page only by address bits (3:2).

The flash module supports fault tolerance through Error Correction Code (ECC) and/or error detection. The ECC implemented within the flash module will correct single bit failures and detect double bit failures.

The flash module uses an embedded hardware algorithm implemented in the Memory Interface to program and erase the flash core.

Control logic that works with the software block enables, and software lock mechanisms, is included in the embedded hardware algorithm to guard against accidental program/erase.

The hardware algorithm perform the steps necessary to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the flash module reads as logic level 0 (or low).

An erased bit in the flash module reads as logic level 1 (or high).

Program and erase of the flash module requires multiple system clock cycles to complete.

The erase sequence may be suspended.

The program and erase sequences may be aborted.

17.3.4.2 Flash module sectorization

The data flash module supports 64 KB of user memory, plus 16 KB of test memory (a portion of which is one-time programmable by the user).

The flash module is composed by a single Bank (Bank 0): Read-While-Modify is not supported.

Bank 0 of the 80 KB flash memory macrocell is divided in four sectors. Bank 0 contains also a reserved sector named Test flash in which some One Time Programmable user data are stored.

The sectorization of the 80 KB Matrix Module is shown in Table 17-38.

Table 17-38. 80 KB flash module sectorization

Bank	Sector	Addresses	Size	Address Space
B1	B1F0	0x00800000 to 0x00803FFF	16 KB	Low Address Space
B1	B1F1	0x00804000 to 0x00807FFF	16 KB	Low Address Space

Table 17-38. 80 KB flash module sectorization (continued)

Bank	Sector	Addresses	Size	Address Space
B1	B1F2	0x00808000 to 0x0080BFFF	16 KB	Low Address Space
B1	B1F3	0x0080C000 to 0x0080FFFF	16 KB	Low Address Space
B1	Reserved	0x00810000 to 0x00BFFFFF		Reserved
B1	B1TF	0x00C00000 to 0x00C03FFF	16 KB	Test Address Space

The flash module is divided into blocks also to implement independent Erase/Program protection. A software mechanism is provided to independently lock/unlock each block in low, mid and high address space against program and erase.

17.3.4.2.1 Test flash block

The Test flash block exists outside the normal address space and is programmed and read independently of the other blocks. The independent Test flash block is included also to support systems which require non-volatile memory for security and/or to store system initialization information.

A section of the Test flash is reserved to store the non-volatile information related to redundancy, configuration, and protection.

The ECC is applied also to Test flash.

The usage of reserved Test flash sector is detailed in the following table.

Table 17-39. Test flash structure

Name	Description	Addresses	Size
	User OTP Area	0x400000 to 0x401FFF	8192 byte
	Reserved	0x402000 to 0x403CFF	7424 byte
	User Reserved	0x403D00 to 0x403DE7	232 byte
NVLML	Non-Volatile Low/Mid Address Space Block Locking Register	0x403DE8 to 0x403DEF	8 byte
NVHBL	Non-Volatile High Address Space Block Locking Register	0x403DF0 to 0x403DF7	8 byte
NVSLL	Non-Volatile Secondary Low/Mid Address Space Block Lock Register	0x403DF8 to 0x403DFF	8 byte
	User Reserved	0x403E00 to 0x403EFF	256 byte
	Reserved	0x403F00 to 0x403FFF	256 byte

When the Test space is enabled, all the operations are mapped to the Test block.

User mode program of the test block are enabled only when MCR.PEAS is high, also if the Shadow block is available.

The Test flash block may be locked/unlocked against program by using the LML.TSLK and SLL.STSLK registers. Erase of the Test flash block is always locked in user mode.

Program of the Test flash block has similar restriction as the array in terms of how ECC is calculated. Only one program is allowed per 64-bit ECC segment.

The first 8 KB of the Test flash block may be used for user-defined functions (possibly to store boot code, other configuration words or factory process codes). Locations of the Test flash block marked as reserved cannot be programmed by the user application.

17.3.5 User mode operation

In User mode, the flash module may be read and written (register writes and interlock writes), programmed or erased.

The default state of the flash module is read.

The main and test address space can be read only in the read state.

The flash memory registers are always available for read, also when the Module is in Power-Down mode (except few documented registers).

The flash module enters the read state on reset.

The Module is in the read state under two sets of conditions:

- The read state is active when the Module is enabled (User mode read)
- The read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend).

Notice that no Read-While-Modify is available.

Flash core reads return 128 bits (1 page = 2 double words).

Registers reads return 32 bits (1 word).

Flash core reads are done through the Bus Interface Unit.

Registers reads to unmapped register address space will return all 0s.

Registers writes to unmapped register address space will have no effect.

Array reads attempted to invalid locations will result in indeterminate data. Invalid locations occur when addressing is done to blocks that do not exist in non 2^n array sizes.

Interlock writes attempted to invalid locations, will result in an interlock occurring, but attempts to program these blocks will not occur since they are forced to be locked. Erase will occur to selected and unlocked blocks even if the interlock write is to an invalid location.

Simultaneous Read cycle on the flash memory matrix and read/write cycles on the registers are possible. On the contrary, register read/write accesses simultaneous to a flash memory matrix interlock write are forbidden.

17.3.5.1 Reset

A reset is the highest priority operation for the flash module and terminates all other operations.

The flash module uses reset to initialize register and status bits to their default reset values.

If the flash module is executing a Program or Erase operation (MCR.PGM = 1 or MCR.ERS = 1) and a reset is issued, the operation will be terminated and the module will disable the high voltage logic without damage to the high voltage circuits. Reset terminates all operations and forces the flash module into User mode, ready to receive accesses. Reset and power-off must not be used systematically to terminate a Program or Erase operation.

After reset is negated, read register access may be done, although it should be noted that registers that require updating from shadow information, or other inputs, may not read updated values until MCR.DONE transitions. MCR.DONE may be polled to determine if the flash module has transitioned out of reset. Notice that the registers cannot be written until MCR.DONE is high.

17.3.5.2 Power-Down mode

The Power-Down mode turns off all flash memory DC current sources, so that power dissipation is due only to leakage.

In Power-Down mode no reads from or write to the Module are possible.

The user may not read some registers (UMISR0-4, UT1-2 and part of UT0) until the Disable mode is exited. On the contrary write access is locked on all the registers in Power-Down mode.

When enabled the flash module returns to its previous state in all cases unless in the process of executing an erase high voltage operation at the time of entering Power-Down mode.

If the flash memory module enters Power-Down mode during an erase operation, the MCR[ESUS] bit is set. The user may resume the erase operation when the module exits Power-Down mode by clearing the MCR[ESUS] bit. MCR[EHV] must be high to resume the erase operation.

If the flash memory module is configured to enter Power-Down mode during a program operation, the operation will be completed and the Power-Down mode will be entered only after the programming ends.

If the flash memory module is put in Power-Down mode and the Vector Table remains mapped in the flash memory address space, the user must take care that the flash memory macrocell will strongly increase the interrupt response time by adding several wait states.

It is forbidden to enter Low-Power mode when the Power-Down mode is active.

17.3.5.3 Low-Power mode

The Low-Power mode turns off most of the DC current sources within the flash memory module.

The module (flash core and registers) is not accessible for read or write after it enters Low-Power mode.

The wakeup time from Low-Power mode is faster than the wakeup time from Power-Down mode.

The user may not read some registers (UMISR0-4, UT1-2 and part of UT0) until the Low-Power mode is exited. Write access is locked on all the registers in Low-Power mode.

When exiting from Low-Power mode the flash memory module returns to its previous state in all cases unless it was in the process of executing an erase high voltage operation at the time of entering Low-Power mode.

If the flash memory module enters Low-Power mode during an erase operation, the MCR[ESUS] bit is set. The user may resume the erase operation when the module exits Low-Power mode by clearing the MCR[ESUS] bit. The MCR[EHV] bit must be high to resume the erase operation.

If the flash memory module is configured to enter Low-Power mode during a program operation, the operation will be completed and the Low-Power mode will be entered only after the programming ends.

It is forbidden to enter Power-Down mode when the Low-Power mode is active.

17.3.6 Register description

The flash memory user registers mapping is shown in the following table.

Table 17-40. Flash 528 KB single bank registers

Register name	Address Offset	Location
Module Configuration Register (MCR)	0x00	on page 601
Low/mid Address Space Block Locking Register (LML)	0x04	on page 605
High Address Space Block Locking Register (HBL)	0x08	on page 608
Secondary Low/Mid Address Space Block Lock Register (SLL)	0x0C	on page 609
Low/mid Address Space Block Select Register (LMS)	0x10	on page 612
High Address Space Block Select Register (HBS)	0x14	on page 613
Address Register (ADR)	0x18	on page 614
Reserved	0x1C–0x3B	—
User Test Register 0 (UT0)	0x3C	on page 616
User Test Register 1 (UT1)	0x40	on page 618
User Test Register 2 (UT2)	0x44	on page 618
User Multiple Input Signature Register 0 (UMISR0)	0x48	on page 619
User Multiple Input Signature Register 1 (UMISR1)	0x4C	on page 620
User Multiple Input Signature Register 2 (UMISR2)	0x50	on page 620
User Multiple Input Signature Register 3 (UMISR3)	0x54	on page 621
User Multiple Input Signature Register 4 (UMISR4)	0x58	on page 622

In the following some non-volatile registers are described. Please notice that such entities are not Flip-Flops, but locations of Test flash sector with a special meaning.

During the flash memory initialization phase, the FPEC reads these non-volatile registers and update their related Volatile Registers. When the FPEC detects ECC double errors in these special locations, it behaves in the following way:

- In case of a failing system locations (configurations, device options, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged.
- In case of failing user locations (protections, censorship, BIU, etc.), the Volatile Registers are filled with all 1s and the flash memory initialization ends setting low the PEG bit of MCR.

In this section, the following abbreviations are used

Table 17-41. Abbreviations

Case	Abbreviation	Description
read/write	rw	The software can read and write to these bits.
read/clear	rc	The software can read and clear to these bits.
read-only	r	The software can only read these bits.
write-only	w	The software should only write to these bits.

17.3.6.1 Module Configuration Register (MCR)

Address Offset: 0x0000 Reset value: 0x06600600

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EDC	0	0	0	0	SIZE 2	SIZE 1	SIZE 0	0	LAS2	LAS1	LAS0	0	0	0	MAS
rc/0	r/0	r/0	r/0	r/0	r/1	r/1	r/0	r/0	r/1	r/1	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EER	RWE	0	0	PEAS	DONE	PEG	0	0	0	0	PGM	PSUS	ERS	ESUS	EHV
rc/0	rc/0	r/0	r/0	r/0	r/1	r/1	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-27. Module Configuration Register (MCR)

The Module Configuration Register is used to enable and monitor all the Modify Operations of the flash module.

Table 17-42. MCR field descriptions

Field	Description
0	**EDC:** *ECC Data Correction* (Read/Clear) EDC provides information on previous reads. If a ECC Single Error detection and correction occurred, the EDC bit will be set to 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user. In the event of a ECC Double Error detection, this bit will not be set. If EDC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EDC) were not corrected through ECC. Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect. The function of this bit is SoC dependent and it can be configured to be disabled. 0: Reads are occurring normally. 1: An ECC Single Error occurred and was corrected during a previous read.

Table 17-42. MCR field descriptions (continued)

Field	Description
1:4	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.
5:7	**SIZE2-0:** *array space SIZE 2-0* (Read Only) 110 64 KB array space size
8	*Reserved* (Read Only). Write this bit has no effect and read this bit always outputs 0.
9:11	**LAS2-0:** *Low Address Space 2-0* (Read Only) 010 Low address space sectorization is 32 KB + 2×16 KB + 2×32 KB + 128 KB
12:14	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.
15	**MAS:** *Mid Address Space* (Read Only) 0 No mid address space
16	**EER:** *ECC event Error* (Read/Clear) EER provides information on previous reads. If a ECC Double Error detection occurred, the EER bit will be set to 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user. In the event of a ECC Single Error detection and correction, this bit will not be set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) were correct. Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect. 0: Reads are occurring normally. 1: An ECC Double Error occurred during a previous read.
17	**RWE:** *Read-while-Write event Error* (Read/Clear) RWE provides information on previous reads when a Modify operation is on going. If a RWW Error occurs, the RWE bit will be set to 1. Read-While-Write Error means that a read access to the flash memory matrix has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) were correct. Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect. 0: Reads are occurring normally. 1: A RWW Error occurred during a previous read.
18:19	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.

Table 17-42. MCR field descriptions (continued)

Field	Description
20	**PEAS:** *Program/Erase Access Space* (Read Only) PEAS is used to indicate which space is valid for Program and Erase operations: main array space or shadow/test space. PEAS=0 indicates that the main address space is active for all flash module program and erase operations. PEAS=1 indicates that the test or shadow address space is active for program and erase. The value in PEAS is captured and held with the first interlock write done for Modify operations. The value of PEAS is retained between sampling events (such as subsequent first interlock writes). 0: Shadow/Test address space is disabled for program/erase and main address space enabled. 1: Shadow/Test address space is enabled for program/erase and main address space disabled.
21	**DONE:** *modify operation DONE* (Read Only) DONE indicates if the flash module is performing a high voltage operation. DONE is set to 1 on termination of the flash module reset. DONE is cleared to 0 just after a 0 to 1 transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation. DONE is set to 1 at the end of program and erase high voltage sequences. DONE is set to 1 (within t_{PABT} or t_{EABT}, equal to P/E Abort Latency) after a 1 to 0 transition of EHV, which aborts a high voltage Program/Erase operation. DONE is set to 1 (within t_{ESUS}, time equals to Erase Suspend Latency) after a 0 to 1 transition of ESUS, which suspends an erase operation. 0 Flash is executing a high voltage operation. 1 Flash is not executing a high voltage operation.
22	**PEG:** *Program/Erase Good* (Read Only) The PEG bit indicates the completion status of the last flash program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the Program and Erase high voltage operations. Aborting a Program/Erase high voltage operation will cause PEG to be cleared to 0, indicating the sequence failed. PEG is set to 1 when the flash module is reset, unless a flash memory initialization error has been detected. The value of PEG is valid only when PGM=1 and/or ERS=1 and after DONE transitions from 0 to 1 due to an abort or the completion of a Program/Erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by ESUS being set to logic 1. If Program or Erase are attempted on blocks that are locked, the response will be PEG=1, indicating that the operation was successful, and the content of the block were properly protected from the Program or Erase operation. If a Program operation tries to program at 1 bits that are at 0, the program operation is correctly executed on the new bits to be programmed at 0, but PEG is cleared, indicating that the requested operation has failed. In Array Integrity Check or Margin mode, PEG is set to 1 when the operation is completed, regardless the occurrence of any error. The presence of errors can be detected only comparing checksum value stored in UMIRS0–1. Aborting an Array Integrity Check or a Margin mode operation will cause PEG to be cleared to 0, indicating the sequence failed. 0 Program or Erase operation failed. 1 Program or Erase operation successful.
23:26	*Reserved* (Read Only) Write these bits has no effect and read these bits always outputs 0.

Table 17-42. MCR field descriptions (continued)

Field	Description
27	**PGM:** *ProGraM* (Read/Write) PGM is used to setup the flash module for a Program operation. A 0 to 1 transition of PGM initiates a Program sequence. A 1 to 0 transition of PGM ends the Program sequence. PGM can be set only under User mode read (ERS is low and UT0.AIE is low). PGM can be cleared by the user only when EHV is low and DONE is high. PGM is cleared on reset. 0: Flash is not executing a Program sequence. 1: Flash is executing a Program sequence.
28	**PSUS:** *Program SUSpend* (Read/Write) Write this bit has no effect, but the written data can be read back.
29	**ERS:** *ERaSe* (Read/Write) ERS is used to setup the flash module for an Erase operation. A 0 to 1 transition of ERS initiates an Erase sequence. A 1 to 0 transition of ERS ends the Erase sequence. ERS can be set only under User mode read (PGM is low and UT0.AIE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0: Flash is not executing an Erase sequence. 1: Flash is executing an Erase sequence.
30	**ESUS:** *Erase SUSpend* (Read/Write) ESUS is used to indicate that the flash module is in Erase Suspend or in the process of entering a Suspend state. The flash module is in Erase Suspend when ESUS=1 and DONE=1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in Erase Suspend. The flash module enters Suspend within t_{ESUS} of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV=1 starts the sequence which clears DONE and returns the Module to Erase. The flash module cannot exit Erase Suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0: Erase sequence is not suspended. 1: Erase sequence is suspended.

Table 17-42. MCR field descriptions (continued)

Field	Description
31	**EHV:** *Enable High Voltage* (Read/Write) The EHV bit enables the flash module for a high voltage Program/Erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a Program/Erase sequence. EHV may be set under one of the following conditions: Erase (ERS=1, ESUS=0, UT0.AIE=0) Program (ERS=0, ESUS=0, PGM=1, UT0.AIE=0) In normal operation, a 1 to 0 transition of EHV with DONE high and ESUS low terminates the current Program/Erase high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the eventual Suspend bit low. An abort causes the value of PEG to be cleared, indicating a failing Program/Erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. A suspended operation cannot be aborted. Aborting a high voltage operation will leave the flash module addresses in an undeterminate data state. This may be recovered by executing an Erase on the affected blocks. EHV may be written during Suspend. EHV must be high to exit Suspend. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low. 0: Flash is not enabled to perform an high voltage operation. 1: Flash is enabled to perform an high voltage operation.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit by bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The flash module does not allow the user to write bits simultaneously which would put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in Table 17-43.

Table 17-43. MCR bits set/clear priority levels

Priority level	MCR bits
1	ERS
2	PGM
3	EHV
4	ESUS

If the user attempts to write two or more MCR bits simultaneously then only the bit with the lowest priority level will be written.

17.3.6.2 Low/Mid Address Space Block Locking Register (LML)

Address Offset: 0x0004

Reset value: 0x00XXXXXX, initially determined by NVLML value from test sector.

17.3.6.3 Non-Volatile Low/Mid Address Space Block Locking Register (NVLML)

Address Offset: 0x403DE8 Delivery value: 0xFFFFFFFF

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LME	0	0	0	0	0	0	0	0	0	0	TSLK	0	0	MLK1	MLK0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/X	r/0	r/0	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LLK15	LLK14	LLK13	LLK12	LLK11	LLK10	LLK9	LLK8	LLK7	LLK6	LLK5	LLK4	LLK3	LLK2	LLK1	LLK0
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-28. Non-Volatile Low/Mid Address Space Block Locking Register (NVLML)

The Low/Mid Address Space Block Locking register provides a means to protect blocks from being modified. These bits, along with bits in the SLL register, determine if the block is locked from Program or Erase. An “OR” of LML and SLL determine the final lock status.

The LML register has a related non-volatile Low/Mid Address Space Block Locking register located in Test flash that contains the default reset value for LML: The NVLML register is read during the reset phase of the flash module and loaded into the LML.

The NVLML register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 17-44. NVLML field descriptions

Field	Description
0	**LME**: *Low/Mid address space block Enable* (Read Only) This bit is used to enable the Lock registers (TSLK, MLK1-0 and LLK15-0) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the LME bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For LME the password 0xA1A11111 must be written to the LML register. 0: Low Address Locks are disabled: TSLK, MLK1-0 and LLK15-0 cannot be written. 1: Low Address Locks are enabled: TSLK, MLK1-0 and LLK15-0 can be written.
1:10	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-44. NVLML field descriptions (continued)

Field	Description
11	**TSLK**: *Test/Shadow address space block LocK* (Read/Write) This bit is used to lock the block of Test and Shadow Address Space from Program and Erase (Erase is any case forbidden for Test block). A value of 1 in the TSLK register signifies that the Test/Shadow block is locked for Program and Erase. A value of 0 in the TSLK register signifies that the Test/Shadow block is available to receive Program and Erase pulses. The TSLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the TSLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the TSLK register. The TSLK bit may be written as a register. Reset will cause the bit to go back to its Test flash block value. The default value of the TSLK bit (assuming erased fuses) would be locked. TSLK is not writable unless LME is high. 0: Test/Shadow Address Space Block is unlocked and can be modified (if also SLL.STSLK=0). 1: Test/Shadow Address Space Block is locked and cannot be modified.
12:13	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
14:15	**MLK1-0**: *Mid address space block LocK 1-0* (Read/Write) These bits are used to lock the blocks of Mid Address Space from Program and Erase. All the MLK1-0 are not used for this memory cut that is all mapped in low address space. A value of 1 in a bit of the MLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the MLK register signifies that the corresponding block is available to receive Program and Erase pulses. The MLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the MLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the MLK registers. The MLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the MLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the MLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In the 80 KB flash macrocell bits MLK1-0 are read-only and locked at 1. MLK is not writable unless LME is high. 0: Mid Address Space Block is unlocked and can be modified (if also SLL.SMLK=0). 1: Mid Address Space Block is locked and cannot be modified.

Table 17-44. NVLML field descriptions (continued)

Field	Description
16:31	**LLK15-0**: *Low address space block LocK 15-0* (Read/Write) These bits are used to lock the blocks of Low Address Space from Program and Erase. LLK3-0 are related to sectors B1F3-0, respectively. LLK15-4 are not used for this memory cut. A value of 1 in a bit of the LLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the LLK register signifies that the corresponding block is available to receive Program and Erase pulses. The LLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the LLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the LLK registers. The LLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the LLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the LLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In the 80 KB flash Macrocell bits LLK15-4 are read-only and locked at 1. LLK is not writable unless LME is high. 0: Low Address Space Block is unlocked and can be modified (if also SLL.SLK=0). 1: Low Address Space Block is locked and cannot be modified.

17.3.6.4 High Address Space Block Locking Register (HBL)

Address Offset: 0x0008

Reset value: 0x000000XX, initially determined by NVHBL, located in test sector.

17.3.6.5 Non-Volatile High Address Space Block Locking Register (NVHBL)

Address Offset: 0x403DF0 Delivery value: 0xFFFFFFFF

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HBE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	0	HLK5	HLK4	HLK3	HLK2	HLK1	HLK0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-29. Non-Volatile High Address Space Block Locking Register (NVHBL)

The High Address Space Block Locking register provides a means to protect blocks from being modified. The HBL register has a related non-volatile High Address Space Block Locking register located in Test flash that contains the default reset value for HBL: The NVHBL register is read during the reset phase of the flash module and loaded into the HBL.

The NVHBL register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 17-45. HBL field descriptions

Field	Description
0	*High address space Block Enable* (Read Only) This bit is used to enable the Lock registers (HLK5-0) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the HBE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For HBE the password 0xB2B22222 must be written to the HBL register. 0: High Address Locks are disabled: HLK5-0 cannot be written. 1: High Address Locks are enabled: HLK5-0 can be written.
1:25	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
26:31	**HLK5-0**: *High address space block LocK 5-0* (Read/Write) These bits are used to lock the blocks of High Address Space from Program and Erase. All the HLK5-0 are not used for this memory cut that is all mapped in low address space. A value of 1 in a bit of the HLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the HLK register signifies that the corresponding block is available to receive Program and Erase pulses. The HLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the HLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the HLK registers. The HLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the HLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the HLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In the 80 KB flash Macrocell bits HLK5-0 are read-only and locked at 1. HLK is not writable unless HBE is high. 0: High Address Space Block is unlocked and can be modified. 1: High Address Space Block is locked and cannot be modified.

17.3.6.6 Secondary Low/Mid Address Space Block Locking Register (SLL)

Address Offset: 0x000C

Reset value: 0x00XXXXXX, initially determined by NVSLL, located in test sector

17.3.6.7 Non-Volatile Secondary Low/Mid Address Space Block Locking Register (NVSLL)

Address Offset: 0x403DF8 Delivery value: 0xFFFFFFFF

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLE	0	0	0	0	0	0	0	0	0	0	STSLK	0	0	SMK1	SMK0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/X	r/0	r/0	rw/X	rw/X

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SLK15	SLK14	SLK13	SLK12	SLK11	SLK10	SLK9	SLK8	SLK7	SLK6	SLK5	SLK4	SLK3	SLK2	SLK1	SLK0
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 17-30. Non-Volatile Secondary Low/Mid Address Space Block Locking Register (NVSLL)

The Secondary Low/Mid Address Space Block Locking register provides an alternative means to protect blocks from being modified. These bits, along with bits in the LML register, determine if the block is locked from Program or Erase. An “OR” of LML and SLL determine the final lock status.

The SLL register has a related non-volatile Secondary Low/Mid Address Space Block Locking register located in Test flash that contains the default reset value for SLL: The NVSLL register is read during the reset phase of the flash module and loaded into the SLL.

The NVSLL register is a 64-bit register, the 32 most significant bits of which (bits 63-32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 17-46. NVSLL field descriptions

Field	Description
0	**SLE**: *Secondary Low/mid address space block Enable* (Read Only) This bit is used to enable the Lock registers (STSLK, SMK1-0 and SLK15-0) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the SLE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For SLE the password 0xC3C33333 must be written to the SLL register. 0: Secondary Low/Mid Address Locks are disabled: STSLK, SMK1-0 and SLK15-0 cannot be written. 1: Secondary Low/Mid Address Locks are enabled: STSLK, SMK1-0 and SLK15-0 can be written.
1:10	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-46. NVSLL field descriptions (continued)

Field	Description
11	**STSLK**: *Secondary Test/Shadow address space block LocK* (Read/Write) This bit is used as an alternate means to lock the block of Test and Shadow Address Space from Program and Erase (Erase is any case forbidden for Test block). A value of 1 in the STSLK register signifies that the Test/Shadow block is locked for Program and Erase. A value of 0 in the STSLK register signifies that the Test/Shadow block is available to receive Program and Erase pulses. The STSLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the STSLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the STSLK register. The STSLK bit may be written as a register. Reset will cause the bit to go back to its Test flash block value. The default value of the STSLK bit (assuming erased fuses) would be locked. STSLK is not writable unless SLE is high. 0: Test/Shadow Address Space Block is unlocked and can be modified (if also LML.TSLK=0). 1: Test/Shadow Address Space Block is locked and cannot be modified.
12:13	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
14:15	**SMK1-0**: *Secondary Mid address space block locK 1-0* (Read/Write) These bits are used as an alternate means to lock the blocks of Mid Address Space from Program and Erase. All the SMK1-0 are not used for this memory cut that is all mapped in low address space. A value of 1 in a bit of the SMK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the SMK register signifies that the corresponding block is available to receive Program and Erase pulses. The SMK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the SMK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the SMK registers. The SMK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the SMK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the SMK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In the 80 KB flash Macrocell bits SMK1-0 are read-only and locked at 1. SMK is not writable unless SLE is high. 0: Mid Address Space Block is unlocked and can be modified (if also LML.MLK=0). 1: Mid Address Space Block is locked and cannot be modified.

Table 17-46. NVSLL field descriptions (continued)

Field	Description
16:31	**SLK15-0**: *Secondary Low address space block locK 15-0* (Read/Write) These bits are used as an alternate means to lock the blocks of Low Address Space from Program and Erase. SLK3-0 are related to sectors B1F3-0, respectively. SLK15-4 are not used for this memory cut. A value of 1 in a bit of the SLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the SLK register signifies that the corresponding block is available to receive Program and Erase pulses. The SLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the SLK register is not writable if a high voltage operation is suspended. Upon reset, information from the Test flash block is loaded into the SLK registers. The SLK bits may be written as a register. Reset will cause the bits to go back to their Test flash block value. The default value of the SLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the SLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the Test flash block), and register writes will have no effect. In the 80 KB flash Macrocell bits SLK15-4 are read-only and locked at 1. SLK is not writable unless SLE is high. 0: Low Address Space Block is unlocked and can be modified (if also LML.LLK=0). 1: Low Address Space Block is locked and cannot be modified.

17.3.6.8 Low/Mid Address Space Block Select Register (LMS)

Address Offset: 0x00010 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	MSL1	MSL0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LSL15	LSL14	LSL13	LSL12	LSL11	LSL10	LSL9	LSL8	LSL7	LSL6	LSL5	LSL4	LSL3	LSL2	LSL1	LSL0
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-31. Low/Mid Address Space Block Select Register (LMS)

The Low/Mid Address Space Block Select register provides a means to select blocks to be operated on during erase.

Table 17-47. LMS field descriptions

Field	Description
0:13	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-47. LMS field descriptions (continued)

Field	Description
14:15	**MSL1-0**: *Mid address space block SeLect 1-0* (Read/Write) A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected. All the MSL1-0 are not used for this memory cut that is all mapped in low address space. The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended. In the event that blocks are not present (due to configuration or total memory size), the corresponding MSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect. In the 80 KB flash Macrocell bits MSL1-0 are read-only and locked at 0. 0: Mid Address Space Block is unselected for Erase. 1: Mid Address Space Block is selected for Erase.
16:31	**LSL15-0**: *Low address space block SeLect 15-0* (Read/Write) A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected. LSL3-0 are related to sectors B1F3-0, respectively. LSL15-4 are not used for this memory cut. The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended. In the event that blocks are not present (due to configuration or total memory size), the corresponding LSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect. In the 80 KB flash Macrocell bits LSL15-4 are read-only and locked at 0. 0: Low Address Space Block is unselected for Erase. 1: Low Address Space Block is selected for Erase.

17.3.6.9 High Address Space Block Select Register (HBS)

Address Offset: 0x00014 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	0	HSL5	HSL4	HSL3	HSL2	HSL1	HSL0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-32. High Address Space Block Select Register (HBS)

The High Address Space Block Select register provides a means to select blocks to be operated on during erase.

Table 17-48. HBS field descriptions

Field	Description
0:25	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
26:31	**HSL5-0**: *High address space block SeLect 5-0* (Read/Write) A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected. All the HSL5-0 are not used for this memory cut that is all mapped in low address space. The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended. In the event that blocks are not present (due to configuration or total memory size), the corresponding HSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect. In the 80 KB flash Macrocell bits HSL5-0 are read-only and locked at 0. 0: High Address Space Block is unselected for Erase. 1: High Address Space Block is selected for Erase.

17.3.6.10 Address Register (ADR)

Address Offset: 0x00018 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	AD22	AD21	AD20	AD19	AD18	AD17	AD16
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	0	0	0
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0

Figure 17-33. Address Register (ADR)

The Address Register provides the first failing address in the event module failures (ECC, RWW or FPEC) or the first address at which a ECC single error correction occurs.

Table 17-49. ADR field descriptions

Field	Description
0:8	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-49. ADR field descriptions (continued)

Field	Description
9:28	**AD22-3**: *ADdress 22-3* (Read Only) The Address Register provides the first failing address in the event of ECC error (MCR.EER set) or the first failing address in the event of RWW error (MCR.RWE set), or the address of a failure that may have occurred in a FPEC operation (MCR.PEG cleared). The Address Register provides also the first address at which a ECC single error correction occurs (MCR.EDC set), if the SoC is configured to show this feature. The ECC double error detection takes the highest priority, followed by the RWW error, the FPEC error and the ECC single error correction. When accessed ADR will provide the address related to the first event occurred with the highest priority. The priorities between these 4 possible events is summarized in the following table. This address is always a Double Word address that selects 64 bits. In case of a simultaneous ECC Double Error Detection on both Double Words of the same page, bit AD3 will output 0. The same is valid for a simultaneous ECC Single Error Correction on both Double Words of the same page. In User mode, the Address Register is read-only. **Note:** An erroneous update of the Address register (ADR) occurs whenever there is a sequence of three or more events affecting the ADR (ECC single or double bit errors or RWW error) and both the following conditions apply: — The priorities are ordered in such a way that only the first event should update ADR. — The last event, although it does not update ADR, sets the Read While Write Event Error (RWE) bit or the ECC Data Correction (EDC) bit in the Module Configuration Register (MCR). For this case, the ADR is wrongly updated with the address related to one of the intervening events. Example: If a sequence of two double-bit ECC errors is followed by a single-bit correction without clearing the ECC Event Error flag (EER) in the MCR, then the value found in ADR after the single-bit correction event is the one related to the second double-bit error (instead of the first one, as specified).
29:31	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.

Table 17-50. ADR content: priority list

Priority Level	Error Flag	ADR content
1	MCR.EER = 1	Address of first ECC Double Error
2	MCR.RWE = 1	Address of first RWW Error
3	MCR.PEG = 0	Address of first FPEC Error
4	MCR.EDC = 1	Address of first ECC Single Error Correction

17.3.6.11 User Test 0 register (UT0)

Address Offset: 0x0003C Reset value: 0x00000001

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UTE	0	0	0	0	0	0	0	DSI7	DSI6	DSI5	DSI4	DSI3	DSI2	DSI1	DSI0
rw/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	X	MRE	MRV	EIE	AIS	AIE	AID
r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	r/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	r/1

Figure 17-34. User Test 0 register (UT0)

The User Test feature gives the user of the flash module the ability to perform test features on the flash memory.

The User Test 0 Register allows to control the way in which the flash memory content check is done.

Bits MRE, MRV, AIS, EIE and DSI7-0 of the User Test 0 Register are not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-51. UT0 field descriptions

Field	Description
0	**UTE**: *User Test Enable* (Read/Clear) This status bit gives indication when User Test is enabled. All bits in UT0-2 and UMISR0-4 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. For UTE the password 0xF9F99999 must be written to the UT0 register.
1:7	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
8:15	**DSI7-0**: *Data Syndrome Input 7-0* (Read/Write) These bits represent the input of Syndrome bits of ECC logic used in the ECC Logic Check. The DSI7-0 correspond to the 8 syndrome bits on a double word. These bits are not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: The syndrome bit is forced at 0. 1: The syndrome bit is forced at 1.
16:24	*Reserved* (Read Only). Write these bits has no effect and read these bits always outputs 0.
25	*Reserved* (Read/Write). This bit can be written and its value can be read back, but there is no function associated. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-51. UT0 field descriptions (continued)

Field	Description
26	**MRE**: *Margin Read Enable* (Read/Write) MRE enables margin reads to be done. This bit, combined with MRV, enables regular user mode reads to be replaced by margin reads. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: Margin reads are not enabled, all reads are User mode reads. 1: Margin reads are enabled.
27	**MRV**: *Margin Read Value* (Read/Write) If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: Zero's (programmed) margin reads are requested (if MRE=1). 1: One's (erased) margin reads are requested (if MRE=1).
28	**EIE**: *ECC data Input Enable* (Read/Write) EIE enables the ECC Logic Check operation to be done. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: ECC Logic Check is not enabled. 1: ECC Logic Check is enabled.
29	**AIS**: *Array Integrity Sequence* (Read/Write) AIS determines the address sequence to be used during array integrity checks or Margin mode. The default sequence (AIS=0) is meant to replicate sequences normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS=1) is just logically sequential. It should be noted that the time to run a sequential sequence is significantly shorter than the time to run the proprietary sequence. Only the Sequential mode is allowed in Margin mode. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0: Array Integrity sequence is proprietary sequence. 1: Array Integrity sequence or Margin mode sequence is sequential.
31	**AIE**: *Array Integrity Enable* (Read/Write) AIE set to 1 starts the Array Integrity Check done on all selected and unlocked blocks. The pattern is selected by AIS, and the MISR (UMISR0-4) can be checked after the operation is complete, to determine if a correct signature is obtained. AIE can be set only if MCR.ERS, MCR.PGM and MCR.EHV are all low. 0: Array Integrity Checks, Margin mode, and ECC Logic Checks are not enabled. 1: Array Integrity Checks, Margin mode, and ECC Logic Checks are enabled.
31	**AID**: *Array Integrity Done* (Read Only) AID will be cleared upon an Array Integrity Check being enabled (to signify the operation is on-going). Once completed, AID will be set to indicate that the Array Integrity Check is complete. At this time the MISR (UMISR0-4) can be checked. 0: Array Integrity Check is on-going. 1: Array Integrity Check is done.

17.3.6.12 User Test 1 register (UT1)

Address Offset: 0x00040 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DAI31	DAI30	DAI29	DAI28	DAI27	DAI26	DAI25	DAI24	DAI23	DAI22	DAI21	DAI20	DAI19	DAI18	DAI17	DAI16
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DAI15	DAI14	DAI13	DAI12	DAI11	DAI10	DAI09	DAI08	DAI07	DAI06	DAI05	DAI04	DAI03	DAI02	DAI01	DAI00
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-35. User Test 1 register (UT1)

The User Test 1 Register allows to enable the checks on the ECC logic related to the 32 LSB of the Double Word.

The User Test 1 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-52. UT1 field descriptions

Field	Description
0:31	**DAI31-00**: *Data Array Input 31-0* (Read/Write) These bits represent the input of even word of ECC logic used in the ECC Logic Check. The DAI31-00 correspond to the 32 array bits representing Word 0 within the double word. 0: The array bit is forced at 0. 1: The array bit is forced at 1.

17.3.6.13 User Test 2 register (UT2)

Address Offset: 0x00044 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DAI63	DAI62	DAI61	DAI60	DAI59	DAI58	DAI57	DAI56	DAI55	DAI54	DAI53	DAI52	DAI51	DAI50	DAI49	DAI48
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DAI47	DAI46	DAI45	DAI44	DAI43	DAI42	DAI41	DAI40	DAI39	DAI38	DAI37	DAI36	DAI35	DAI34	DAI33	DAI32
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-36. User Test 2 register (UT2)

The User Test 2 Register allows to enable the checks on the ECC logic related to the 32 MSB of the Double Word.

The User Test 2 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-53. UT2 field descriptions

Field	Description
0:31	**DAI63-32**: *Data Array Input 63-32* (Read/Write) These bits represent the input of odd word of ECC logic used in the ECC Logic Check. The DAI63-32 correspond to the 32 array bits representing Word 1 within the double word. 0: The array bit is forced at 0. 1: The array bit is forced at 1.

17.3.6.14 User Multiple Input Signature Register 0 (UMISR0)

Address Offset: 0x00048 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS031	MS030	MS029	MS028	MS027	MS026	MS025	MS024	MS023	MS022	MS021	MS020	MS019	MS018	MS017	MS016
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS015	MS014	MS013	MS012	MS011	MS010	MS009	MS008	MS007	MS006	MS005	MS004	MS003	MS002	MS001	MS000
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-37. User Multiple Input Signature Register 0 (UMISR0)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.

The User Multiple Input Signature Register 0 represents the bits 31-0 of the whole 144 bits word (2 Double Words including ECC).

The UMISR0 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-54. UMSIR0 field descriptions

Field	Description
0:31	**MS031-000**: *Multiple input Signature 031-000* (Read/Write) These bits represent the MISR value obtained accumulating the bits 31-0 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR0 register.

17.3.6.15 User Multiple Input Signature Register 1 (UMISR1)

Address Offset: 0x0004C Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS063	MS062	MS061	MS060	MS059	MS058	MS057	MS056	MS055	MS054	MS053	MS052	MS051	MS050	MS049	MS048
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS047	MS046	MS045	MS044	MS043	MS042	MS041	MS040	MS039	MS038	MS037	MS036	MS035	MS034	MS033	MS032
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-38. User Multiple Input Signature Register 1 (UMISR1)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.

The User Multiple Input Signature Register 1 represents the bits 63-32 of the whole 144 bits word (2 Double Words including ECC).

The UMISR1 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-55. UMISR1 field descriptions

Field	Description
0:31	**MS063-032**: *Multiple input Signature 063-032* (Read/Write) These bits represent the MISR value obtained accumulating the bits 63-32 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR1 register.

17.3.6.16 User Multiple Input Signature Register 2 (UMISR2)

Address Offset: 0x00050 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS095	MS094	MS093	MS092	MS091	MS090	MS089	MS088	MS087	MS086	MS085	MS084	MS083	MS082	MS081	MS080
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS079	MS078	MS077	MS076	MS075	MS074	MS073	MS072	MS071	MS070	MS069	MS068	MS067	MS066	MS065	MS064
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-39. User Multiple Input Signature Register 2 (UMISR2)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.
The User Multiple Input Signature Register 2 represents the bits 95-64 of the whole 144 bits word (2 Double Words including ECC).

The UMISR2 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-56. UMISR2 field descriptions

Field	Description
0:31	**MS095-064**: *Multiple input Signature 095-064* (Read/Write) These bits represent the MISR value obtained accumulating the bits 95-64 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR2 register.

17.3.6.17 User Multiple Input Signature Register 3 (UMISR3)

Address Offset: 0x00054 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS127	MS126	MS125	MS124	MS123	MS122	MS121	MS120	MS119	MS118	MS117	MS116	MS115	MS114	MS113	MS112
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS111	MS110	MS109	MS108	MS107	MS106	MS105	MS104	MS103	MS102	MS101	MS100	MS099	MS098	MS097	MS096
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-40. User Multiple Input Signature Register 3 (UMISR3)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.
The User Multiple Input Signature Register 3 represents the bits 127-96 of the whole 144 bits word (2 Double Words including ECC).

The UMISR3 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-57. UMISR3 field descriptions

Field	Description
0:31	**MS127-096**: *Multiple input Signature 127-096* (Read/Write) These bits represent the MISR value obtained accumulating the bits 127-96 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR3 register.

17.3.6.18 User Multiple Input Signature Register 4 (UMISR4)

Address Offset: 0x00058 Reset value: 0x00000000

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MS159	MS158	MS157	MS156	MS155	MS154	MS153	MS152	MS151	MS150	MS149	MS148	MS147	MS146	MS145	MS144
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MS143	MS142	MS141	MS140	MS139	MS138	MS137	MS136	MS135	MS134	MS133	MS132	MS131	MS130	MS129	MS128
rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0	rw/0

Figure 17-41. User Multiple Input Signature Register 4 (UMISR4)

The Multiple Input Signature Register provides a mean to evaluate the Array Integrity.

The User Multiple Input Signature Register 4 represents the ECC bits of the whole 144 bits word (2 Double Words including ECC): bits 8-15 are ECC bits for the odd Double Word and bits 24-31 are the ECC bits for the even Double Word; bits 4-5 and 20-21 of MISR are respectively the double and single ECC error detection for odd and even Double Word.

The UMISR4 Register is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

Table 17-58. UMISR4 field descriptions

Field	Description
0:31	**MS159-128**: *Multiple input Signature 159-128* (Read/Write) These bits represent the MISR value obtained accumulating: the 8 ECC bits for the even Double Word (on MS135-128); the single ECC error detection for even Double Word (on MS138); the double ECC error detection for even Double Word (on MS139); the 8 ECC bits for the odd Double Word (on MS151-144); the single ECC error detection for odd Double Word (on MS154); the double ECC error detection for odd Double Word (on MS155). The MS can be seeded to any value by writing the UMISR4 register.

17.3.7 Programming considerations

17.3.7.1 Modify operation

All the Modify operations of the flash module are managed through the flash memory user registers interface.

All the sectors of the flash module belong to the same partition (Bank), therefore when a Modify operation is active on some sectors no read access is possible on any other sector (Read-While-Modify is not supported).

During a flash memory modify operation, any attempt to read any flash memory location will output invalid data and bit RWE of MCR will be automatically set. This means that the flash module is not fetchable when a Modify Operation is active: The Modify Operation commands must be executed from another Memory (internal Ram or external Memory).

If during a Modify Operation a reset occurs, the operation is suddenly terminated and the Macrocell is reset to Read mode. The data integrity of the flash memory section where the Modify Operation has been terminated is not guaranteed: The interrupted flash memory modify operation must be repeated.

In general each Modify Operation is started through a sequence of 3 steps:

1. The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR (PGM or ERS) or UT0 (MRE or EIE).
2. The second step is the definition of the operands: the Address and the Data for programming or the Sectors for erase or margin read.
3. The third instruction is used to start the Modify Operation, by setting EHV in MCR or AIE in UT0.

Once selected, but not yet started, one operation can be canceled by resetting the operation selection bit.

A summary of the available flash memory modify operations are shown in the following Table 17-36.

Table 17-59. Flash modify operations

Operation	Select bit	Operands	Start bit
Double Word Program	MCR.PGM	Address and Data by Interlock Writes	MCR.EHV
Sector Erase	MCR.ERS	LMS, HBS	MCR.EHV
Array Integrity Check	None	LMS, HBS	UT0.AIE
Margin Read	UT0.MRE	UT0.MRV + LMS, HBS	UT0.AIE
ECC Logic Check	UT0.EIE	UT0.DSI, UT1, UT2	UT0.AIE

In general each Modify Operation is completed through a sequence of four steps:

1. Wait for operation completion: wait for bit MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UMISR0-4 with expected value).
3. Switch-Off FPEC by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE/EIE).

If the device embeds more than one flash memory macrocell and a Modify operation is on-going on one of them, then it is forbidden to start any other Modify operation on the other flash memory macrocells.

In the following, all the possible Modify operations are described and some examples of the sequences needed to activate them are presented.

17.3.7.2 Double Word program

A flash memory program sequence operates on any double word within the flash core.

As many as two words within the Double word may be altered in a single program operation.

Whenever you program, ECC bits also get programmed (unless the selected address belongs to a sector in which the ECC has been disabled in order to allow bit manipulation). ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be of 64 bits. The programming operation should completely fill selected ECC segments within the Double Word.

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

Addresses in locked/disabled blocks cannot be programmed.

The user may program the values in any or all of two words, of a Double Word, with a single program sequence.

Double Word-bound words have addresses which differ only in address bit 2.

The Program operation consists of the following sequence of events:

1. Change the value in the MCR.PGM bit from 0 to 1.
2. Ensure the block that contains the address to be programmed is unlocked.
   Write the first address to be programmed with the program data.
   The flash module latches address bits (22:3) at this time.
   The flash module latches data written as well.
   This write is referred to as a program data interlock write. An interlock write may be as large as 64 bits, and as small as 32 bits (depending on the CPU bus).
3. If more than 1 word is to be programmed, write the additional address in the Double Word with data to be programmed. This is referred to as a program data write.
   The flash modules ignores address bits (22:3) for program data writes.
   The eventual unwritten data word default to 0xFFFFFFFF.
4. Write a logic 1 to the MCR.EHV bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR.DONE bit goes high.
6. Confirm MCR.PEG=1.
7. Write a logic 0 to the MCR.EHV bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR.PGM bit to terminate the program operation.

Program may be initiated with the 0 to 1 transition of the MCR.PGM bit or by clearing the MCR.EHV bit at the end of a previous program.

The first write after a program is initiated determines the page address to be programmed. This first write is referred to as an interlock write. The interlock write determines if the shadow, test or normal array space will be programmed by causing MCR.PEAS to be set/cleared.

An interlock write must be performed before setting MCR.EHV. The user may terminate a program sequence by clearing MCR.PGM prior to setting MCR.EHV.

After the interlock write, additional writes only affect the data to be programmed at the word location determined by address bit 2. Unwritten locations default to a data value of 0xFFFFFFFF. If multiple writes are done to the same location the data for the last write is used in programming.

While MCR.DONE is low and MCR.EHV is high, the user may clear EHV, resulting in a program abort. A Program abort forces the Module to step 8 of the program sequence.

An aborted program will result in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The data space being operated on before the abort will contain indeterminate data. This may be recovered by repeating the same program instruction or executing an erase of the affected blocks.

Example 17-8. Double Word program of data 0x55AA55AA at address 0x00AAA8 and data 0xAA55AA55 at address 0x00AAAC.

```
MCR               = 0x00000010;                 /* Set PGM in MCR: Select Operation */
(0x00AAA8)        = 0x55AA55AA;                 /* Latch Address and 32 LSB data */
(0x00AAAC)        = 0xAA55AA55;                 /* Latch 32 MSB data */
MCR               = 0x00000011;                 /* Set EHV in MCR: Operation Start */
do                                              /* Loop to wait for DONE=1 */
{ tmp             = MCR;                        /* Read MCR */
} while ( !(tmp & 0x00000400) );
status            = MCR & 0x00000200;           /* Check PEG flag */
MCR               = 0x00000010;                 /* Reset EHV in MCR: Operation End */
MCR               = 0x00000000;                 /* Reset PGM in MCR: Deselect Operation */
```

17.3.7.3 Sector erase

Erase changes the value stored in all bits of the selected block(s) to logic 1.

An erase sequence operates on any combination of blocks (sectors) in the low, mid or high address space, or the Shadow block (if available). The test block cannot be erased.

The erase sequence is fully automated within the flash memory. The user only needs to select the blocks to be erased and initiate the erase sequence.

Locked/disabled blocks cannot be erased.

If multiple blocks are selected for erase during an erase sequence, no specific operation order must be assumed.

The Erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from 0 to 1.
2. Select the block(s) to be erased by writing 1s to the appropriate register(s) in LMS or HBS registers.
   If the Shadow block is to be erased, this step may be skipped, and LMS and HBS are ignored.
   Note that Lock and Select are independent. If a block is selected and locked, no erase will occur.
3. Write to any address in flash memory. This is referred to as an erase interlock write.
4. Write a logic 1 to the MCR.EHV bit to start the internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR.DONE bit goes high.

6. Confirm MCR.PEG=1.
7. Write a logic 0 to the MCR.EHV bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR.ERS bit to terminate the erase operation.

After setting MCR.ERS, one write, referred to as an interlock write, must be performed before MCR.EHV can be set to 1. Data words written during erase sequence interlock writes are ignored.

The user may terminate the erase sequence by clearing ERS before setting EHV.

An erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low.

An erase abort forces the Module to step 8 of the erase sequence.

An aborted erase will result in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The block(s) being operated on before the abort contain indeterminate data. This may be recovered by executing an erase on the affected blocks.

The user may not abort an erase sequence while in erase suspend.

Example 17-9. Erase of sectors B0F1 and B0F2.

```
MCR               = 0x00000004;              /* Set ERS in MCR: Select Operation */
LMS               = 0x00000006;              /* Set LSL2-1 in LMS: Select Sectors to erase */
(0x000000)        = 0xFFFFFFFF;              /* Latch a Flash Address with any data */
MCR               = 0x00000005;              /* Set EHV in MCR: Operation Start */
do                                           /* Loop to wait for DONE=1 */
{ tmp             = MCR;                     /* Read MCR */
} while ( !(tmp & 0x00000400) );
status            = MCR & 0x00000200;        /* Check PEG flag */
MCR               = 0x00000004;              /* Reset EHV in MCR: Operation End */
MCR               = 0x00000000;              /* Reset ERS in MCR: Deselect Operation */
```

17.3.7.3.1 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core.

It is not possible to program or to erase during an erase suspend.

During erase suspend, all reads to blocks targeted for erase return indeterminate data.

An erase suspend can be initiated by changing the value of the MCR.ESUS bit from 0 to 1. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0 to 1 transition of MCR.ESUS causes the Module to start the sequence which places it in erase suspend.

The user must wait until MCR.DONE = 1 before the Module is suspended and further actions are attempted. MCR.DONE will go high no more than t_{ESUS} after MCR.ESUS is set to 1.

Once suspended, the array may be read. flash core reads while MCR.ESUS = 1 from the block(s) being erased return indeterminate data.

Example 17-10. Sector erase suspend

```
MCR              = 0x00000007;              /* Set ESUS in MCR: Erase Suspend */
do                                          /* Loop to wait for DONE=1 */
{ tmp            = MCR;                     /* Read MCR */
} while ( !(tmp & 0x00000400) );
```

Notice that there is no need to clear MCR.EHV and MCR.ERS in order to perform reads during erase suspend.

The Erase sequence is resumed by writing a logic 0 to MCR.ESUS.

MCR.EHV must be set to 1 before MCR.ESUS can be cleared to resume the operation.

The Module continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Example 17-11. Sector erase resume

```
MCR              = 0x00000005;                  /* Reset ESUS in MCR: Erase Resume */
```

17.3.7.4 User Test mode

User Test mode is a mode, that customers can put the flash module in, to do specific tests to check the integrity of the flash module.

Three kinds of Test can be performed:

- Array Integrity Self Check
- Margin mode Read
- ECC Logic Check

The User Test mode is equivalent to a Modify operation: read accesses attempted by the user during User Test mode generates a Read-While-Write Error (RWE of MCR set).

It is not allowed to perform User Test operations on the Test and Shadow blocks.

17.3.7.4.1 Array Integrity self check

Array Integrity is checked using a predefined address sequence (proprietary), and this operation is executed on selected and unlocked blocks. Once the operation is completed, the results of the reads can be checked by reading the MISR value (stored in UMISR0-4), to determine if an incorrect read, or ECC detection was noted.

The internal MISR calculator is a 32-bit register.

The 128 bit data, the 16 ECC data and the single and double ECC errors of the two Double Words are therefore captured by the MISR through 5 different read accesses at the same location.

The whole check is done through five complete scans of the memory address space:

1. The first pass will scan only bits 31-0 of each page.
2. The second pass will scan only bits 63-32 of each page.
3. The third pass will scan only bits 95-64 of each page.

4. The fourth pass will scan only bits 127-96 of each page.
5. The fifth pass will scan only the ECC bits (8 + 8) and the single and double ECC errors (2 + 2) of both Double Words of each page.

The 128 bits of data and the 16 ECC data bits are sampled before the eventual ECC correction, while the single and double error flags are sampled after the ECC evaluation.

Only data from existing and unlocked locations are captured by the MISR.

The MISR can be seeded to any value by writing the UMISR0–4 registers.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Select the block(s) to be checked by writing 1s to the appropriate register(s) in LMS or HBS registers.
   Note that Lock and Select are independent. If a block is selected and locked, no Array Integrity Check will occur.
3. Set eventually UT0.AIS bit for a sequential addressing only.
4. Write a logic 1 to the UT0.AIE bit to start the Array Integrity Check.
5. Wait until the UT0.AID bit goes high.
6. Compare UMISR0-4 content with the expected result.
7. Write a logic 0 to the UT0.AIE bit.
8. If more blocks are to be checked, return to step 2.

It is recommended to leave UT0.AIS at 0 and use the proprietary address sequence that checks the read path more fully, although this sequence takes more time.

During the execution of the Array Integrity Check operation it is forbidden to modify the content of Block Select (LMS, HBS) and Lock (LML, SLL, HBL) registers, otherwise the MISR value can vary in an unpredictable way.

While UT0.AID is low and UT0.AIE is high, the user may clear AIE, resulting in a Array Integrity Check abort.

UT0.AID must be checked to know when the aborting command has completed.

Example 17-12. Array Integrity check of sectors B0F1 and B0F2

```
UT0              = 0xF9F99999;              /* Set UTE in UT0: Enable User Test */
LMS              = 0x00000006;              /* Set LSL2-1 in LMS: Select Sectors */
UT0              = 0x80000002;              /* Set AIE in UT0: Operation Start */
do                                          /* Loop to wait for AID=1 */
{ tmp            = UT0;                     /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0            = UMISR0;                  /* Read UMISR0 content*/
data1            = UMISR1;                  /* Read UMISR1 content*/
data2            = UMISR2;                  /* Read UMISR2 content*/
data3            = UMISR3;                  /* Read UMISR3 content*/
data4            = UMISR4;                  /* Read UMISR4 content*/
UT0              = 0x00000000;              /* Reset UTE and AIE in UT0: Operation End */
```

17.3.7.4.2 Margin read

Margin read procedure (either Margin 0 or Margin 1), can be run on unlocked blocks in order to unbalance the Sense Amplifiers, respect to standard read conditions, so that all the read accesses reduce the margin vs. 0 (UT0.MRV = 0) or vs. 1 (UT0.MRV = 1). Locked sectors are ignored by MISR calculation and ECC flagging. The results of the margin reads can be checked comparing checksum value in UMISR0-4.

Since Margin reads are done at voltages that differ than the normal read voltage, lifetime expectancy of the flash memory macrocell is impacted by the execution of Margin reads. Doing Margin reads repetitively results in degradation of the flash memory array, and shorten expected lifetime experienced at normal read levels. For these reasons the Margin read usage is allowed only in Factory, while it is forbidden to use it inside the User Application. In any case the charge losses detected through the Margin mode cannot be considered failures of the device and no Failure Analysis will be opened on them.

The Margin Read Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in LMS or HBS registers.
   Note that Lock and Select are independent. If a block is selected and locked, no Array Integrity Check will occur.
3. Set eventually UT0.AIS bit for a sequential addressing only.
4. Change the value in the UT0.MRE bit from 0 to 1.
5. Select the Margin level: UT0.MRV=0 for 0's margin, UT0.MRV=1 for 1's margin.
6. Write a logic 1 to the UT0.AIE bit to start the Margin Read Setup or skip to step 6 to terminate.
7. Wait until the UT0.AID bit goes high.
8. Compare UMISR0-4 content with the expected result.
9. Write a logic 0 to the UT0.AIE, UT0.MRE and UT0.MRV bits.
10. If more blocks are to be checked, return to step 2.

It is recommended to leave UT0.AIS at 1 and use the linear address sequence that takes less time.

During the execution of the Margin mode operation it is forbidden to modify the content of Block Select (LMS, HBS) and Lock (LML, SLL, HBL) registers, otherwise the MISR value can vary in an unpredictable way.

The read accesses will be done with the addition of a proper number of Wait States to guarantee the correctness of the result.

While UT0.AID is low and UT0.AIE is high, the User may clear AIE, resulting in a Array Integrity Check abort.

UT0.AID must be checked to know when the aborting command has completed.

Example 17-13. Margin Read setup versus 1s

```
UT0     = 0xF9F99999;           /* Set UTE in UT0: Enable User Test */
LMS     = 0x00000006;           /* Set LSL2-1 in LMS: Select Sectors */
UT0     = 0x80000004;           /* Set AIS in UT0: Select Operation */
UT0     = 0x80000024;           /* Set MRE in UT0: Select Operation */
```

```
UT0      = 0x80000034;              /* Set MRV in UT0: Select Margin versus 1's */
UT0      = 0x80000036;              /* Set AIE in UT0: Operation Start */
do                                  /* Loop to wait for AID=1 */
{ tmp    = UT0;                     /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0    = UMISR0;                  /* Read UMISR0 content*/
data1    = UMISR1;                  /* Read UMISR1 content*/
data2    = UMISR2;                  /* Read UMISR2 content*/
data3    = UMISR3;                  /* Read UMISR3 content*/
data4    = UMISR4;                  /* Read UMISR4 content*/
UT0      = 0x80000034;              /* Reset AIE in UT0: Operation End */
UT0      = 0x00000000;              /* Reset UTE, MRE, MRV, AIS in UT0: Deselect Op. */
```

17.3.7.4.3 ECC logic check

ECC logic can be checked by forcing the input of ECC logic: The 64 bits of data and the 8 bits of ECC syndrome can be individually forced and they will drive simultaneously at the same value the ECC logic of the whole page (2 Double Words).

The results of the ECC Logic Check can be verified by reading the MISR value.

The ECC Logic Check operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Write in UT1.DAI31-0 and UT2.DAI63-32 the Double Word Input value.
3. Write in UT0.DSI7-0 the Syndrome Input value.
4. Select the ECC Logic Check: write a logic 1 to the UT0.EIE bit.
5. Write a logic 1 to the UT0.AIE bit to start the ECC Logic Check.
6. Wait until the UT0.AID bit goes high.
7. Compare UMISR0-4 content with the expected result.
8. Write a logic 0 to the UT0.AIE bit.

Notice that when UT0.AID is low UMISR0-4, UT1-2 and bits MRE, MRV, EIE, AIS and DSI7-0 of UT0 are not accessible: reading returns undeterminate data and write has no effect.

Example 17-14. ECC logic check

```
UT0              = 0xF9F99999;          /* Set UTE in UT0: Enable User Test */
UT1              = 0x55555555;          /* Set DAI31-0 in UT1: Even Word Input Data */
UT2              = 0xAAAAAAAA;          /* Set DAI63-32 in UT2: Odd Word Input Data */
UT0              = 0x80FF0000;          /* Set DSI7-0 in UT0: Syndrome Input Data */
UT0              = 0x80FF0008;          /* Set EIE in UT0: Select ECC Logic Check */
UT0              = 0x80FF000A;          /* Set AIE in UT0: Operation Start */
do                                      /* Loop to wait for AID=1 */
{ tmp            = UT0;                 /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0            = UMISR0;              /* Read UMISR0 content (expected 0x55555555) */
data1            = UMISR1;              /* Read UMISR1 content (expected 0xAAAAAAAA) */
data2            = UMISR2;              /* Read UMISR2 content (expected 0x55555555) */
data3            = UMISR3;              /* Read UMISR3 content (expected 0xAAAAAAAA) */
data4            = UMISR4;              /* Read UMISR4 content (expected 0x00FF00FF) */
UT0              = 0x00000000;          /* Reset UTE, AIE and EIE in UT0: Operation End */
```

17.3.8 Error Correction Code (ECC)

The flash memory macrocell provides a method to improve the reliability of the data stored in flash memory: the usage of an error correction code. The word size is fixed of 64 bits.

At each double word of 64 bits, there are associated 8 ECC bits that are programmed in such a way to guarantee a Single Error Correction and a Double Error Detection (SEC-DED).

ECC circuitry provides correction of single bit faults and is used to achieve automotive reliability targets. Some units will experience single bit corrections throughout the life of the product with no impact to product reliability.

17.3.8.1 ECC algorithm

The flash memory macrocell supports one ECC algorithm: “All 1s No Error”. This algorithm detects as valid any Double Word read on a just erased sector (all the 72 bits are 1s).

This option allows to perform a Blank Check after a Sector Erase operation.

17.3.8.2 Bit manipulation

The ECC algorithm allows some bit manipulations so that a Double Word can be rewritten several times without needing an erase of the sector. This allows to use a Double Word to store flags useful for the Eeprom Emulation.

As an example the ECC algorithm allows to start from an All 1s Double Word value and rewrite whichever of its four 16-bits Half-Words to an All 0s content by keeping the same ECC value.

Table 17-60 shows a set of Double Words sharing the same ECC value.

Table 17-60. Bit manipulation: double words with the same ECC value

Double Word	ECC
0xFFFF_FFFF_FFFF_FFFF	0xFF
0xFFFF_FFFF_FFFF_0000	
0xFFFF_FFFF_0000_FFFF	
0xFFFF_0000_FFFF_FFFF	
0x0000_FFFF_FFFF_FFFF	
0xFFFF_FFFF_0000_0000	
0xFFFF_0000_FFFF_0000	
0x0000_FFFF_FFFF_0000	
0xFFFF_0000_0000_FFFF	
0x0000_FFFF_0000_FFFF	
0x0000_0000_FFFF_FFFF	
0xFFFF_0000_0000_0000	
0x0000_FFFF_0000_0000	
0x0000_0000_0000_0000	

17.3.8.3 EEPROM emulation

When some flash memory sectors are used to perform an EEPROM emulation, it is recommended for safety reasons to reserve at least 3 sectors to this purpose.

17.3.9 Protection strategy

Two kind of protections are available: Modify Protection to avoid unwanted program/erase in flash memory sectors and Censored mode to avoid piracy.

17.3.9.1 Modify protection

The flash memory modify protection information is stored in non-volatile flash memory cells located in the Test flash. This information is read once during the flash memory initialization phase following the exit from Reset and is stored in volatile registers that act as actuators.

The reset state of all the volatile modify protection registers is the protected state.

All the non-volatile modify protection registers can be programmed through a normal double word program operation at the related locations in Test flash.

The non-volatile modify protection registers cannot be erased.

- The non-volatile modify protection registers are physically located in Test flash, their bits can be programmed to 0 only once and they can no more be restored to 1.

- The volatile modify protection registers are read/write registers whose bits can be written at 0 or 1 by the user application.

A software mechanism is provided to independently lock/unlock each Low, Mid and High Address Space Block against program and erase.

Software locking is done through the LML (Low/Mid Address Space Block Lock Register) or HBL (High Address Space Block Lock Register) registers.

An alternate means to enable software locking for blocks of Low Address Space only is through the SLL (Secondary Low/Mid Address Space Block Lock Register).

All these registers have a non-volatile image stored in Test flash (NVLML, NVHBL, NVSLL), so that the locking information is kept on reset.

On delivery the Test flash non-volatile image is at all 1s, meaning all sectors are locked.

By programming the non-volatile locations in Test flash, the selected sectors can be unlocked.

Being the Test flash one time programmable (that is, not erasable), once unlocked the sectors cannot be locked again.

Of course, on the contrary, all the volatile registers can be written at 0 or 1 at any time, therefore the user application can lock and unlock sectors when desired.

17.3.9.2 Censored mode

The 80K flash memory macrocell does not contain a Shadow block and all the associated features to manage the Censored mode. These must therefore be managed by the associated code flash memory macrocell embedded in the same SoC.

17.4 Platform flash controller (PFLASH2P_LCA)

17.4.1 Introduction

This section provides an introduction to the 2-port Platform Flash Controller (PFLASH2P_LCA). The PFLASH2P_LCA acts as the interface between two system bus master ports (AHB-Lite 2.v6) and up to three banks of integrated low-cost 90-nm flash memory arrays. It intelligently converts the protocols between the system bus ports and the dedicated flash array interfaces.

A block diagram of the e200z0h Power Architecture reduced product platform (RPP) reference design is shown below in Figure 17-42 with the PFLASH2P_LCA module and its attached off-platform flash memory arrays highlighted.

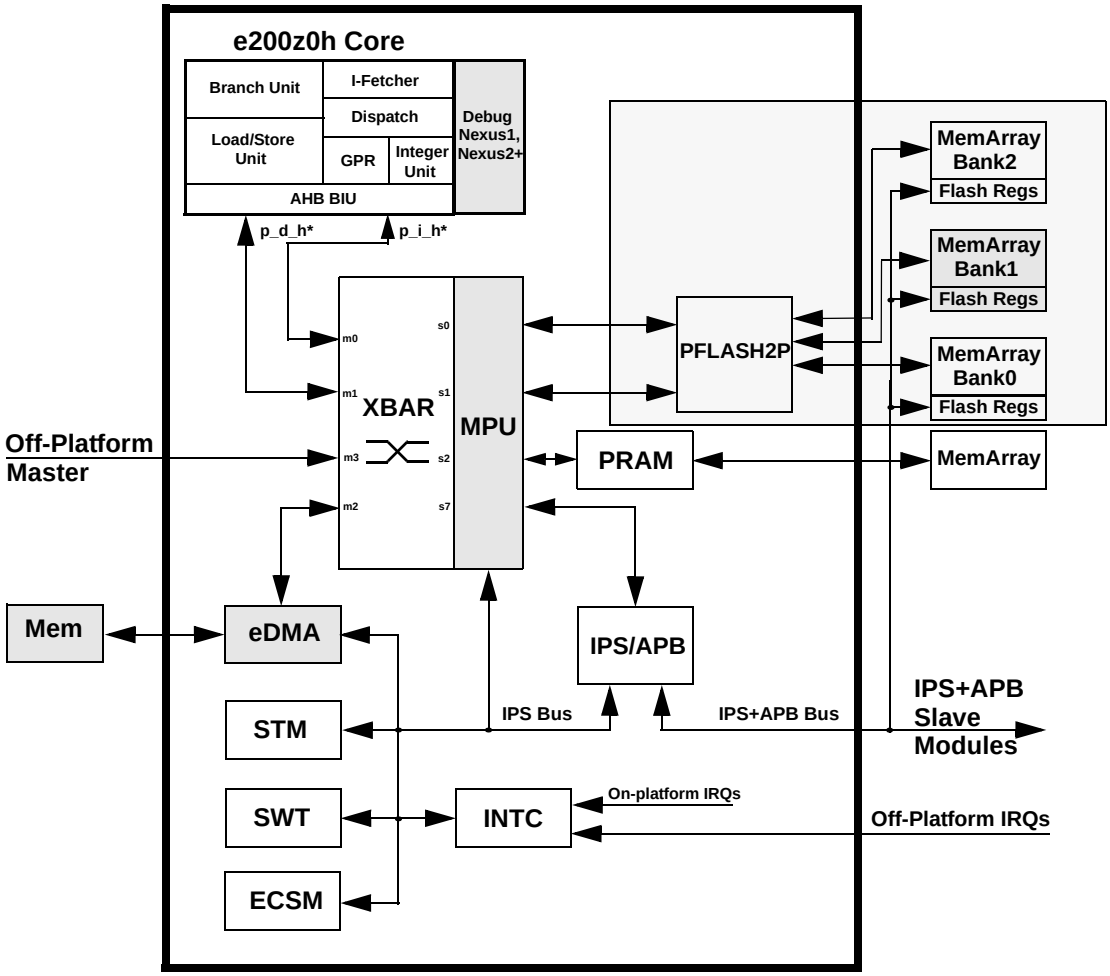


Figure 17-42. Power Architecture e200z0h RPP reference platform block diagram

As shown in this block diagram, there are a number of baseline and optional modules (shaded) supported. The module list includes:

- Power Architecture e200z0h core with Nexus1 or optional Nexus2+ debug
- Optional 16-channel second-generation Direct Memory Access (DMA)
- Optional off-platform bus master, e.g., FlexRay
- AHB crossbar switch (XBAR)
- Optional Memory Protection Unit (MPU)
- 2-port Platform Flash memory controller (PFLASH2P_LCA) with connections to 3 memory banks
- Platform RAM memory controller (PRAM)
- AHB-to-{IPS/APB} bus controller (PBRIDGE-Lite) for access to on- and off-platform slave modules
- Interrupt Controller (INTC)
- 4-channel System Timers (STM)
- Software Watchdog Timer (SWT)

- Error Correction Status Module (ECSM)

The resulting 32-bit Power Architecture e200z0h platform represents a *reference design*, where a single design description can be configured to generate multiple implementations by including/excluding various platform modules as required by a specific application.

Throughout this document, several important terms are used to describe the PFLASH2P_LCA module and its connections. These terms are defined here:

- **Port** — This is used to describe the AMBA-AHB connection(s) into the PFLASH2P_LCA. This flash controller supports 2 AHB ports. For these platform designs, *the PFLASH2P_LCA p0 port is always connected to the processor core and the p1 port is connected to the non-core bus masters*.
- **Bank** — This term is used to describe the attached flash memories. From the PFLASH2P_LCA's perspective, there may be two or three attached banks of flash memory. There are two "code flash" arrays required and they are attached to banks 0 and 2. The PFLASH2P_LCA treats banks 0 and 2 in a common manner with various configuration fields of the programming model shared across the two banks. Additionally, there may be a "data flash" attached to bank1. The PFLASH2P_LCA interface supports three separate connections, one to each memory bank.
- **Array** — Within each memory bank, there are one (or more) flash array instantiations. Recall the maximum capacity of the low-cost array is 512 KB, so devices with larger flash memory bank sizes require multiple instantiations of the array. Within a bank, the array instantiations are named array0, array1, etc. Since the PFLASH2P_LCA module supports interface signals for each bank, it is the responsibility of the SoC design to provide the required address decoding, control generation and read data muxing when there are multiple arrays within a bank. Regardless of the number of array instantiations or the number of populated banks, the operating configuration of the PFLASH2P_LCA is defined by the register values contained in bank0 array0.
- **Page** — This value defines the number of bits read from the flash array in a single access. For this controller and memory, the page size is 128 bits (16 bytes).

The nomenclature "page buffers and "line buffers" are used interchangeably.

From an architectural and programming model perspective, there are two "configuration variables" associated with the PFLASH2P_LCA. These variables define the 2 AHB input ports (p0 and p1) initiating transactions and the three destination flash memory banks (b0, optional b1, b2). The following abbreviations for these variables are used throughout the document:

p0	AHB port 0
p1	AHB port 1
b0, bk0	flash memory bank0
b1, bk1	flash memory bank1 (optional)
b2, bk2	flash memory bank2
b02	flash memory banks 0 and 2

Finally since the page buffers and temporary holding registers are associated with *both* an AHB input port and a flash bank, they use a *bx_py* nomenclature. For example, the b0_p0 page buffer refers to the bank0, port 0 storage elements.

17.4.1.1 Overview

The PFLASH2P_LCA supports a 32-bit data bus width at the two AHB ports and connections to 128-bit read data interfaces from three memory *banks*, where each bank contains one (or more) instantiations of the low-cost flash memory array. Typically, flash bank0 is connected to the first code flash memory, bank2 is connected to a second code flash memory, and bank1 is connected to the optional data flash memory. The memory controller capabilities vary between the banks with each bank's functionality optimized for the typical use cases associated with the attached flash memory. As an example, the PFLASH2P_LCA logic associated with bank0 contains 2 four-entry "page" buffers, one for each AHB input port, where each buffer entry contains 128 bits of data (1 flash page) plus an associated controller which prefetches sequential lines of data from the flash array into the buffer. This structure is repeated for bank2, providing a total of four copies of the 4-entry page buffer. The controller logic associated with bank1 is simpler and only supports two 128-bit registers (again, one for each AHB port) which serve as temporary page holding registers and no support of any prefetching. Prefetch buffer hits from any of the page buffers or temporary holding registers support zero-wait AHB data phase responses. AHB read requests which miss the buffers generate the needed flash array access and the read data is forwarded to the AHB port upon completion, typically incurring two wait-states at an operating frequency of 60–64 MHz. *The logic of the PFLASH2P_LCA is structured to support simultaneous AHB accesses from the two ports fully in parallel when the references are targeted to different memory banks.* If simultaneous AHB accesses reference the same bank, then arbitration logic within the PFLASH2P_LCA determines the order the references are granted access to the bank.

This memory controller is optimized for applications where a cacheless processor core, such as the e200z0h, is connected through the platform to on-chip memories, e.g., flash and RAM, where the processor and platform operate at the same frequency. For these applications, the 2-stage pipeline AMBA-AHB system bus is effectively mapped directly into stages of the processor's pipeline and zero wait-state responses for most memory accesses are critical for providing the required level of system performance.

17.4.1.2 Features

The following list summarizes the key features of the PFLASH2P_LCA:

- Triple bank interfaces support up to a total of 16 Mbytes of flash memory, partitioned as two 4 Mbyte code banks (0, 2) and a separate optional 8 Mbyte data bank (1)
- Dual AHB input port interfaces support a 32-bit data bus. All AHB aligned and unaligned reads within the 32-bit container are supported. Only aligned word writes are supported.
- Array interfaces support a 128-bit read data bus and a 64-bit write data bus for each of the 3 banks
- Internal hardware structure supports fully concurrent accesses from the dual AHB input ports when accessing different flash banks
  - If the AHB ports reference the same flash bank, there is arbitration logic which determines the order the accesses are granted access to the bank
  - Programmable arbitration allows the user to select fixed priority or round-robin
- Total flash page storage in the PFLASH2P_LCA includes four 4-entry page buffers (b0_p0, b0_p1, b2_p0, b2_p1) and two 128-bit temporary holding registers (b1_p0, b1_p1).

  - Each AHB input port provides configurable and independent read buffering and page prefetch support for banks 0 and 2
  - Each AHB input port includes four page read buffers (each 128 bits wide) and a prefetch controller to support single-cycle read responses (zero AHB data phase wait-states) for hits in the buffers. The buffers implement a least-recently-used replacement algorithm to maximize performance.
  - Each AHB input port interfaces to the optional data flash (bank1) includes a 128-bit register to temporarily hold a single flash page. This logic supports single-cycle read responses (zero AHB data phase wait-states) for accesses that hit in the holding register. There is no support for prefetching associated with this bank.
- Programmable response for read-while-write sequences including support for stall-while-write, optional stall notification interrupt, optional flash operation abort, and optional abort notification interrupt
- Separate and independent configurable access timing (common settings for banks 0 and 2, separate settings for bank1) to support use across a wide range of platforms and frequencies
- Support of address-based read access timing for emulation of other memory types
- Support for reporting of single- and multi-bit flash ECC events
- Typical operating configuration loaded into programming model by system reset

Figure 17-43 shows a simplified block diagram of the PFLASH2P_LCA memory controller.

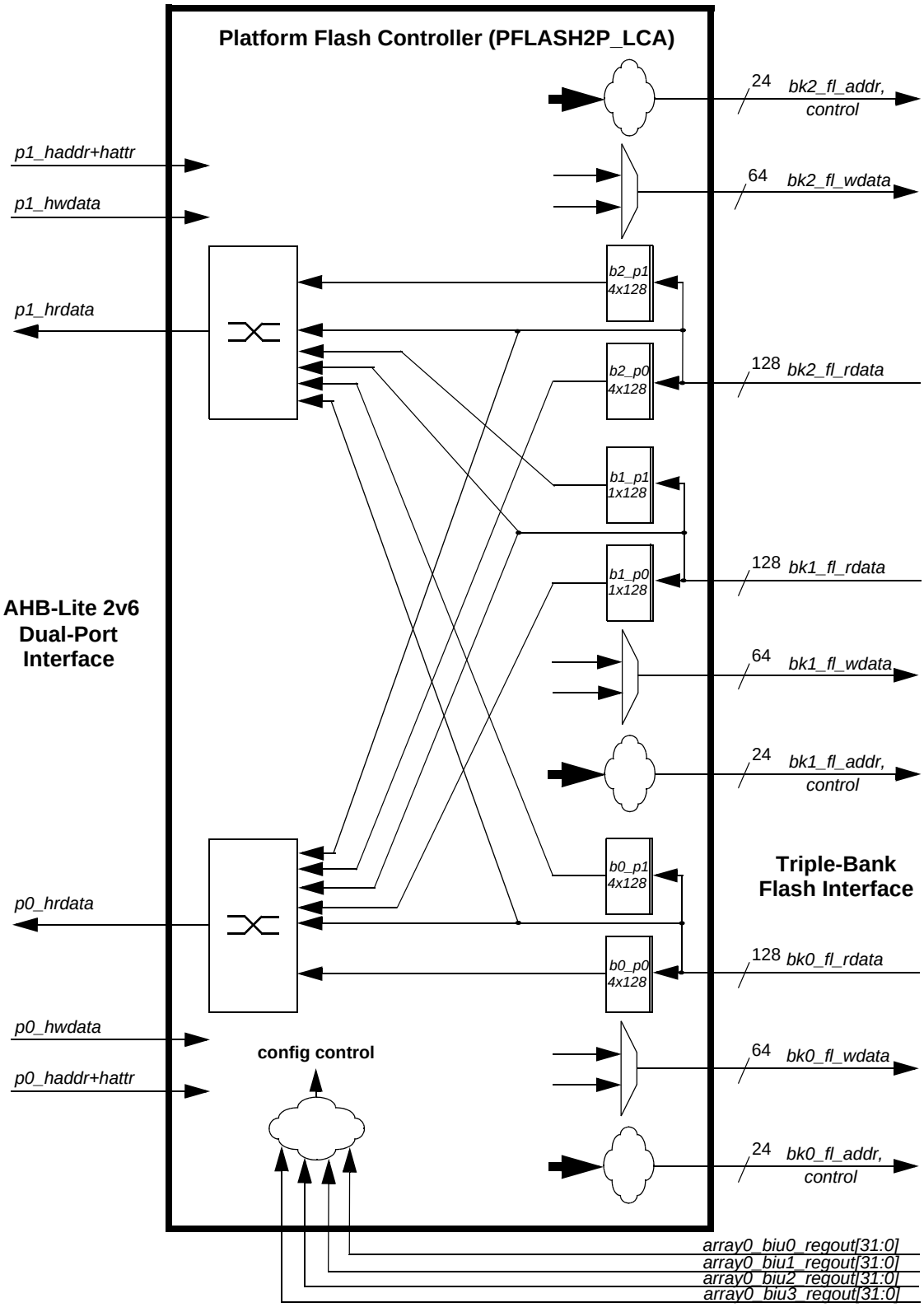


Figure 17-43. PFLASH2P_LCA memory controller block diagram

17.4.1.3 Modes of operation

The PFLASH2P_LCA module does not support any special modes of operation. Its operation is driven from the AMBA-AHB memory references it receives from the platform's bus masters. Its configuration is defined by the setting of its programming model registers, physically located as part of the flash array modules.

17.4.2 External signal descriptions

The PFLASH2P_LCA does not directly interface with any external signals. As shown in Figure 17-42 and Figure 17-43, its primary internal interfaces include two input connections from AMBA-AHB crossbar (or memory protection unit) slave ports and output connections with up to three banks (2 code and 1 data) of flash memory, each containing one or more instantiations of the low-cost flash array. Additionally, the operating configuration for the PFLASH2P_LCA is defined by the contents of certain bank0 array0 registers which are inputs to the module.

A summary of the major PFLASH2p_LCA internal connections is shown in Table 17-61.

Table 17-61. PFLASH2P_LCA Module Connections

PFLASH2P_LCA Connection	Description
Input p0	Processor Core
Input p1	Non-core Masters
Output b0	Bank0, Code Flash
Output b1	Bank1, Data Flash
Output b2	Bank2, Code Flash

17.4.3 Memory map and register definition

There are two memory maps associated with the PFLASH2P_LCA: one for the flash memory space and another for the program-visible control and configuration registers. The flash memory space is accessed via the AMBA-AHB ports while the program-visible registers are accessed via the slave peripheral bus. Details on both memory spaces are provided in Section 17.4.3.1, Memory map.

There are no program-visible registers that physically reside inside the PFLASH2P_LCA. Rather, the PFLASH2P_LCA receives control and configuration information from the flash array controller(s) to determine the operating configuration. These are part of the flash array's configuration registers mapped into its slave peripheral (IPS) address space but are described here.

17.4.3.1 Memory map

First, consider the flash memory space accessed via transactions from the PFLASH2P_LCA's AHB ports.

To support the three separate flash memory banks, the PFLASH2P_LCA controller uses address bits 23 and 19 (*haddr[23, 19]*) to steer the access to the appropriate memory bank. The address decode allocates two 4 Mbyte spaces for bank0 and bank2 and an 8 Mbyte space for bank1. In addition to the actual flash memory regions, there are shadow and test sectors included in the system memory map. The

program-visible control and configuration registers associated with each memory array are included in the slave peripheral address region. The system memory map defines up to 4 code flash arrays and 1 data flash array, although the address space for 3 additional data flash arrays is reserved. See Table 17-62.

Table 17-62. PFLASH2P_LCA decodes for flash-related regions in the system memory map

Start Address	End Address	Size [KB]	Region
0x0000_0000	0x0007_FFFF	512	Bank0 = Code flash array 0
0x0008_0000	0x000F_FFFF	512	Bank2 = Code flash array 1
0x0010_0000	0x0017_FFFF	512	Bank0 = Reserved for Code flash array 2
0x0018_0000	0x001F_FFFF	512	Bank2 = Reserved for Code flash array 3
0x0020_0000	0x0027_FFFF	512	Bank0 = Code flash array 0: Shadow block
0x0028_0000	0x002F_FFFF	512	Bank2 = Code flash array 1: Shadow block
0x0030_0000	0x0037_FFFF	512	Bank0 = Reserved for Code flash array 2: Shadow block
0x0038_0000	0x003F_FFFF	512	Bank2 = Reserved for Code flash array 3: Shadow block
0x0040_0000	0x0047_FFFF	512	Bank0 = Code flash array 0: test sector
0x0048_0000	0x004F_FFFF	512	Bank2 = Code flash array 1: test sector
0x0050_0000	0x0057_FFFF	512	Bank0 = Reserved for Code flash array 2: test sector
0x0058_0000	0x005F_FFFF	512	Bank2 = Reserved for Code flash array 3: test sector
0x0060_0000	0x007F_FFFF	2048	Reserved
0x0080_0000	0x0087_FFFF	512	Data flash array 0
0x0088_0000	0x008F_FFFF	512	Reserved for Data flash array 1
0x0090_0000	0x0097_FFFF	512	Reserved for Data flash array 2
0x0098_0000	0x009F_FFFF	512	Reserved for Data flash array 3
0x00A0_0000	0x00A7_FFFF	512	Data flash array 0: Shadow block
0x00A8_0000	0x00AF_FFFF	512	Reserved for Data flash array 1: Shadow block
0x00B0_0000	0x00B7_FFFF	512	Reserved for Data flash array 2: Shadow block
0x00B8_0000	0x00BF_FFFF	512	Reserved for Data flash array 3: Shadow block
0x00C0_0000	0x00C7_FFFF	512	Data flash array 0: test sector
0x00C8_0000	0x00CF_FFFF	512	Reserved for Data flash array 1: test sector
0x00D0_0000	0x00D7_FFFF	512	Reserved for Data flash array 2: test sector
0x00D8_0000	0x00DF_FFFF	512	Reserved for Data flash array 3: test sector
0x00E0_0000	0x00FF_FFFF	2048	Reserved
0x0100_0000	0x1FFF_FFFF	507904	Emulation Mapping
0xFFE8_8000	0xFFE8_BFFF	16	Code flash array 0 configuration[1]
0xFFE8_C000	0xFFE8_FFFF	16	Data flash array 0 configuration[1]
0xFFEB_0000	0xFFEB_3FFF	16	Code flash array 1 configuration[2]

Table 17-62. PFLASH2P_LCA decodes for flash-related regions in the system memory map (continued)

Start Address	End Address	Size [KB]	Region
0xFFEB_4000	0xFFEB_7FFF	16	Reserved for Code flash array 2 configuration[2]
0xFFEB_8000	0xFFEB_BFFF	16	Reserved for Code flash array 3 configuration[2]
1 This region is also aliased to address 0xC3F8_nnnn. 2 This region is also aliased to address 0xC3FB_nnnn.			

For additional information on the address-based read access timing for emulation of other memory types, see Section 17.4.4.12, Wait-State emulation.

Next, consider the memory map associated with the control and configuration registers.

There are multiple registers that control operation of the PFLASH2P_LCA. These registers are generically defined as “Bus Interface Unit *n* (BIU *n*) Register” in the flash array documentation, where *n* = 0,1,2,3 and are to be only referenced with 32-bit accesses. Note the first two flash array registers (BIU0, BIU1) are reset to an SoC-defined value, while the remaining two array registers (BIU2, BIU3) are loaded at reset from specific locations in the array’s shadow region.

Regardless of the number of populated banks or the number of flash arrays included in a given bank, *the configuration of the PFLASH2P_LCA is wholly specified by the BIU registers associated with bank0 array0*. These register settings define the operating behavior of **all** flash banks; it is recommended that the BIU registers for all physically present arrays be set to the bank0 array0 values.

NOTE

To perform program and erase operations, the control registers in the actual referenced flash array must be programmed, but the configuration of the PFLASH2P_LCA module is defined by the BIU*n* registers of bank0 array0.

The 32-bit memory map for the PFLASH2P_LCA control registers is shown in Table 17-63.

Table 17-63. PFLASH2P_LCA 32-bit memory map

Address	Register	Access	Reset Value	Location
0xFFE8_8000 + 0x01C	Platform Flash Configuration Register 0 (PFCR0)	R/W	0x1085_93ED	on page 642
0xFFE8_8000 + 0x020	Platform Flash Configuration Register 1 (PFCR1)	R/W	0x1085_8181	on page 646
0xFFE8_8000 + 0x024	Platform Flash Access Protection Register (PFAPR)	R/W	0xFFFF_FFFF	on page 647

17.4.3.2 Register descriptions

This section details the individual registers of the PFLASH2P_LCA. *To be consistent with the flash documentation, this description uses a LSB=0 vector bit numbering convention.*

Within the PFLASH2P_LCA's programming model, there are a variety of control and configuration fields. Some are associated with the operating configuration of the memory banks, while others are related to the behavior of the AHB master ports.

Due to limitations in the available register bits in the programming model, the PFLASH controllers (both the single and dual-ported versions) do *not* provide completely symmetric capabilities for the various memory banks. In fact, the PFLASH2P_LCA groups together the attributes of the two code flash arrays attached to bank0 and bank2 of the controller while the configuration of the data flash (bank1) is treated separately.

First, consider the operating configuration of the flash banks. In particular, there are 4 unique configuration fields that are associated with a bank. These include all the parameters associated with the timing (read and write wait states, address pipeline control) as well as the read-while-write control field. Accordingly, the programming model supports two separate sets of these 4 fields: one for banks 0 and 2 in PFCR0, and another for bank1 in PFCR1:

```
// per memory bank configuration controls
        b02_apc,  b1_apc  // address pipeline control
        b02_wwsc, b1_wwsc // write wait state control
        b02_rwsc, b1_rwsc //  read wait state control
        b02_rwwc, b1_rwwc // read-while-write control
```

where *b02* is used to refer to configuration and control information common to banks 0 and 2 while *b1* refers to bank1.

Second, there are a total of 6 configuration fields that relate to the operation of the PFLASH2P_LCA's page buffers. These fields are defined on a "per port" basis since the control needs to be associated *with the AHB master port* and *not the destination flash bank*. In addition, recall that bank1, connected to the data flash, does not support prefetching, etc., so the configuration controls for that bank are considerably reduced compared to banks 0 and 2. The resulting fields are:

```
// per ahb master port configuration controls
        b02_p0_bcfg,  b02_p1_bcfg// page buffer configuration
        b02_p0_dpfen, b02_p1_dpfen// data prefetch enable
        b02_p0_ipfen, b02_p1_ipfen// inst prefetch enable
        b02_p0_pflim, b02_p1_pflim// page buffer prefetch limit
        b02_p0_bfen,  b02_p1_bfen// page buffer enable for banks 0,2
        b1_p0_bfen,   b1_p1_bfen// page buffer enable for bank1
```

All these fields are located in the PFCR0 and PFCR1 registers described below.

17.4.3.2.1 Platform Flash Configuration Register 0 (PFCR0)

This register defines the configuration associated with flash memory banks 0 and 2. Collectively, this corresponds to the "code flash" and the operating configuration defined by certain fields applies to both

memory banks. Additionally, it includes fields that provide specific information for the two separate AHB ports (p0 and p1). The register is described below in Figure 17-44 and Table 17-64.

Offset 0x01C Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	B02_APC				B02_WWSC				B02_RWSC							B02_RWWC
Reset	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B02_RWWC	B02_P1_BCFG		B02_P1_DPFE	B02_P1_IPFE	B02_P1_PFLM		B02_P1_BFE	B02_RWWC	B0_P0_BCFG		B02_P0_DPFE	B02_P0_IPFE	B02_P0_PFLM		B02_P0_BFE
Reset	1	0	0	1	0	0	1	1	1	1	1	0	1	1	0	1

Figure 17-44. PFLASH Configuration Register 0 (PFCR0)

Table 17-64. PFLASH Configuration Register 0 field descriptions

Field	Description
B02_APC	Bank0+2 Address Pipelining Control. This field is used to control the number of cycles between flash array access requests. This field must be set to a value appropriate to the operating frequency of the PFLASH. The required settings are documented in Table 17-70. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to 0b00010 by hardware reset. 00000 Accesses may be initiated on consecutive (back-to-back) cycles 00001 Access requests require one additional hold cycle 00010 Access requests require two additional hold cycles ... 11110 Access requests require 30 additional hold cycles 11111 Access requests require 31 additional hold cycles
B02_WWSC	Bank0+2 Write Wait State Control. This field is used to control the number of wait-states to be added to the flash array access time for writes. This field must be set to a value appropriate to the operating frequency of the PFLASH. The required settings are documented in Table 17-70. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to 0b00010 by hardware reset. 00000 No additional wait-states are added 00001 1 additional wait-state is added 00010 2 additional wait-states are added ... 111111 31 additional wait-states are added

Table 17-64. PFLASH Configuration Register 0 field descriptions (continued)

Field	Description
B02_RWSC	Bank0+2 Read Wait State Control. This field is used to control the number of wait-states to be added to the flash array access time for reads. This field must be set to a value corresponding to the operating frequency of the PFLASH and the actual read access time of the PFLASH. The required settings are documented in the SoC specification. Higher operating frequencies require non-zero settings for this field for proper flash operation. 0 MHz, < 23 MHz APC=RWSC=0 23 MHz, < 45 MHz APC=RWSC=1 45 MHz, < 68 MHz APC=RWSC=2 68 MHz, < 90 MHz APC=RWSC=3 00000 No additional wait-states are added 00001 1 additional wait-state is added 00010 2 additional wait-states are added ... 111111 31 additional wait-states are added This field is set to 0b00010 by hardware reset.
B02_RWWC	Bank0+2 Read-While-Write Control. This 3-bit field defines the controller response to flash reads while the array is busy with a program (write) or erase operation. 0-- Terminate any attempted read while write/erase with an error response 111 Generate a bus stall for a read while write/erase, disable the stall notification interrupt, disable the abort + abort notification interrupt 110 Generate a bus stall for a read while write/erase, enable the stall notification interrupt, disable the abort + abort notification interrupt 101 Generate a bus stall for a read while write/erase, enable the operation abort, disable the abort notification interrupt 100 Generate a bus stall for a read while write/erase, enable the operation abort and the abort notification interrupt This field is set to 0b111 by hardware reset enabling the stall-while-write/erase and disabling the abort and notification interrupts.
B02_P1_BCFG	Bank0+2, Port 1 Page Buffer Configuration. This field controls the configuration of the four page buffers in the PFLASH controller. The buffers can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. If enabled, when a buffer miss occurs, it is allocated to the least-recently-used buffer within the group and the just-fetched entry then marked as most-recently-used. If the flash access is for the next-sequential line, the buffer is not marked as most-recently-used until the given address produces a buffer hit. 00 All four buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffers 0 and 1 allocated for instruction fetches and buffers 2 and 3 for data accesses. 11 The buffers are partitioned into two groups with buffers 0,1,2 allocated for instruction fetches and buffer 3 for data accesses.
B02_P1_DPFE	Bank0+2, Port 1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is set by hardware reset. 0 No prefetching is triggered by a data read access 1 If page buffers are enabled (B02_P1_BFE = 1), prefetching is triggered by any data read access
B02_P1_IPFE	Bank0+2, Port 1 Instruction Prefetch Enable. This field enables or disables prefetching initiated by an instruction fetch read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction fetch read access 1 If page buffers are enabled (B02_P1_BFE = 1), prefetching is triggered by any instruction fetch read access

Table 17-64. PFLASH Configuration Register 0 field descriptions (continued)

Field	Description
B02_P1_PFLM	Bank0+2, Port 1 Prefetch Limit. This field controls the prefetch algorithm used by the PFLASH controller. This field defines the prefetch behavior. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is set to 2b01 by hardware reset. 00 No prefetching is performed. 01 The referenced line is prefetched on a buffer miss, that is, *prefetch on miss*. 1- The referenced line is prefetched on a buffer miss, or the next sequential page is prefetched on a buffer hit (if not already present), that is, *prefetch on miss or hit*.
B02_P1_BFE	Bank0+2, Port 1 Buffer Enable. This bit enables or disables page buffer read hits. It is also used to invalidate the buffers. This bit is set by hardware reset, enabling the page buffers. 0 The page buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The page buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.
B02_P0_BCFG	Bank0+2, Port 0 Page Buffer Configuration. This field controls the configuration of the four page buffers in the PFLASH controller. The buffers can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. If enabled, when a buffer miss occurs, it is allocated to the least-recently-used buffer within the group and the just-fetched entry then marked as most-recently-used. If the flash access is for the next-sequential line, the buffer is not marked as most-recently-used until the given address produces a buffer hit. 00 All four buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffers 0 and 1 allocated for instruction fetches and buffers 2 and 3 for data accesses. 11 The buffers are partitioned into two groups with buffers 0,1,2 allocated for instruction fetches and buffer 3 for data accesses. This field is set to 2b11 by hardware reset.
B02_P0_DPFE	Bank0+2, Port 0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 If page buffers are enabled (B0_P0_BFE = 1), prefetching is triggered by any data read access
B02_P0_IPFE	Bank0+2, Port 0 Instruction Prefetch Enable. This field enables or disables prefetching initiated by an instruction fetch read access. This field is set by hardware reset. 0 No prefetching is triggered by an instruction fetch read access 1 If page buffers are enabled (B0_P0_BFE = 1), prefetching is triggered by any instruction fetch read access
B02_P0_PFLM	Bank0+2, Port 0 Prefetch Limit. This field controls the prefetch algorithm used by the PFLASH controller. This field defines the prefetch behavior. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is set to 2b10 by hardware reset. 00 No prefetching is performed. 01 The referenced line is prefetched on a buffer miss, that is, *prefetch on miss*. 1- The referenced line is prefetched on a buffer miss, or the next sequential page is prefetched on a buffer hit (if not already present), that is, *prefetch on miss or hit*.
B02_P0_BFE	Bank0+2, Port 0 Buffer Enable. This bit enables or disables page buffer read hits. It is also used to invalidate the buffers. This bit is set by hardware reset. 0 The page buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The page buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

17.4.3.2.2 Platform Flash Configuration Register 1 (PFCR1)

This register defines the configuration associated with flash memory bank1. This typically corresponds to the optional “data flash”. If bank1 is not present, the contents of this register are ignored. The register is described below in Figure 17-45 and Table 17-65.

Offset 0x020 Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	B1_APC				B1_WWSC				B1_RWSC							B1_RWWC
W																
Reset	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	B1_RWWC	0	0	0	0	0	0	B1_P1_BFE	B1_RWWC	0	0	0	0	0	0	B1_P0_BFE
W																
Reset	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1

Figure 17-45. PFLASH Configuration Register 1 (PFCR1)

Table 17-65. PFLASH Configuration Register 1 field descriptions

Field	Description
B1_APC	Bank1 Address Pipelining Control. This field is used to control the number of cycles between flash array access requests. This field must be set to a value appropriate to the operating frequency of the PFLASH. The required settings are documented in the SoC specification. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to 0b00010 by hardware reset. 00000 Accesses may be initiated on consecutive (back-to-back) cycles 00001 Access requests require one additional hold cycle 00010 Access requests require two additional hold cycles ... 11110 Access requests require 30 additional hold cycles 11111 Access requests require 31 additional hold cycles This field is ignored in single bank flash configurations.
B1_WWSC	Bank1 Write Wait State Control. This field is used to control the number of wait-states to be added to the flash array access time for writes. This field must be set to a value appropriate to the operating frequency of the PFLASH. The required settings are documented in the SoC specification. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to an appropriate value by hardware reset. This field is set to 0b00010 by hardware reset. 00000 No additional wait-states are added 00001 1 additional wait-state is added 00010 2 additional wait-states are added ... 111111 31 additional wait-states are added This field is ignored in single bank flash configurations.

Table 17-65. PFLASH Configuration Register 1 field descriptions (continued)

Field	Description
B1_RWSC	Bank1 Read Wait State Control. This field is used to control the number of wait-states to be added to the flash array access time for reads. This field must be set to a value corresponding to the operating frequency of the PFLASH and the actual read access time of the PFLASH. The required settings are documented in the SoC specification. Higher operating frequencies require non-zero settings for this field for proper flash operation. Shown below are the maximum operating frequencies for legal APC and RWSC settings based on estimated low-cost flash access times at 150C. The integrator is strongly encouraged to verify these settings based on actual silicon results. 0 MHz, < 23 MHz APC=RWSC=0 23 MHz, < 45 MHz APC=RWSC=1 45 MHz, < 68 MHz APC=RWSC=2 68 MHz, < 90 MHz APC=RWSC=3 00000 No additional wait-states are added 00001 1 additional wait-state is added 00010 2 additional wait-states are added ... 111111 31 additional wait-states are added This field is ignored in single bank flash configurations. This field is set to 0b00010 by hardware reset.
B1_RWWC	Bank1 Read-While-Write Control. This 3-bit field defines the controller response to flash reads while the array is busy with a program (write) or erase operation. 0-- Terminate any attempted read while write/erase with an error response 111 Generate a bus stall for a read while write/erase, disable the stall notification interrupt, disable the abort + abort notification interrupt 110 Generate a bus stall for a read while write/erase, enable the stall notification interrupt, disable the abort + abort notification interrupt 101 Generate a bus stall for a read while write/erase, enable the operation abort, disable the abort notification interrupt 100 Generate a bus stall for a read while write/erase, enable the operation abort and the abort notification interrupt This field is set to 0b111 by hardware reset enabling the stall-while-write/erase and disabling the abort and notification interrupts. This field is ignored in single bank flash configurations.
B1_P1_BFE	Bank1, Port 1 Buffer Enable. This bit enables or disables read hits from the 128-bit holding register. It is also used to invalidate the contents of the holding register. This bit is set by hardware reset, enabling the use of the holding register. 0 The holding register is disabled from satisfying read requests. 1 The holding register is enabled to satisfy read requests on hits.
B1_P0_BFE	Bank1, Port 0 Buffer Enable. This bit enables or disables read hits from the 128-bit holding register. It is also used to invalidate the contents of the holding register. This bit is set by hardware reset, enabling the use of the holding register. 0 The holding register is disabled from satisfying read requests. 1 The holding register is enabled to satisfy read requests on hits.

17.4.3.2.3 Platform Flash Access Protection Register (PFAPR)

The PFLASH Access Protection Register (PFAPR) is used to control read and write accesses to the flash based on system master number. Prefetching capabilities are defined on a per master basis. This register also defines the arbitration mode between the 2 AHB ports for the PFLASH2P_LCA. The register is described below in Figure 17-46 and Table 17-66.

The contents of the register are loaded from location 0x203E00 of the shadow region in the code flash (bank0) array at reset. To temporarily change the values of any of the fields in the PFAPR, a write to the IPS-mapped register is performed. To change the values loaded into the PFAPR *at reset*, the word location at address 0x203E00 of the shadow region in the flash array must be programmed using the normal sequence of operations. The reset value shown in Figure 17-46 reflects an erased or unprogrammed value from the shadow region.

Offset 0x024 Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	ARBM		M7PFD	M6PFD	M5PFD	M4PFD	M3PFD	M2PFD	M1PFD	M0PFD
W																
Reset	*	*	*	*	*	*	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M7AP		M6AP		M5AP		M4AP		M3AP		M2AP		M1AP		M0AP	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 17-46. PFLASH Access Protection Register (PFAPR)

Table 17-66. PFLASH Access Protection Register field descriptions

Field	Description
ARBM	Arbitration mode. This 2-bit field controls the arbitration for PFLASH controllers supporting 2 AHB ports. The port arbitration mode is used only when accesses from the 2 AHB ports attempt to simultaneously reference the same flash bank. Simultaneous references to different memory banks are processed concurrently. 00 Fixed priority arbitration with AHB p0 > p1 01 Fixed priority arbitration with AHB p1 > p0 1- Round-robin arbitration
MxPFD	Master x Prefetch Disable (x = 0,1,2,...,7). These bits control whether prefetching may be triggered based on the master number of the requesting AHB master. This field is further qualified by the PFCRn[B02_Px_DPFE, B02_Px_IPFE, Bx_Py_BFE] bits. 0 Prefetching may be triggered by this master 1 No prefetching may be triggered by this master
MxAP	Master x Access Protection (x = 0,1,2,...,7). These fields control whether read and write accesses to the flash are allowed based on the master number of the initiating module. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

17.4.4 Functional description

References to the PFLASH2P_LCA block diagram shown in Figure 17-43 will assist in understanding much of the discussion in this section.

The PFLASH2P_LCA interfaces between 2 AHB-Lite 2.v6 system bus master ports and three banks of low-cost flash memory arrays.

The PFLASH2P_LCA generates three sets of interface signals for the flash banks, including read and write enables, the flash array address, write size, and write data as inputs to each flash bank. The PFLASH2P_LCA captures read data from the flash banks and drives it onto the AHB. Each flash bank includes data storage for fetched pages on a per AHB port basis, either in the form of 4-entry page buffers (banks 0 and 2) or a 1-entry temporary holding register (bank 1). Pages may be prefetched in advance of being requested by the AHB interface, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Multiple prefetch control algorithms are available for controlling page read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

Recall the logic of the PFLASH2P_LCA is structured to support simultaneous AHB accesses from the two ports fully in parallel when the references are targeted to different memory banks. If simultaneous AHB accesses reference the same bank, then arbitration logic within the PFLASH2P_LCA determines the order the references are granted access to the bank. For more information, see Section 17.4.4.10, Input port arbitration.

17.4.4.1 Access protections

The PFLASH2P_LCA provides programmable configurable access protections for both read and write cycles from masters via the PFLASH Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in Section 17.4.3.2.3, Platform Flash Access Protection Register (PFAPR). Detection of a protection violation results in an error response from the PFLASH2P_LCA on the AHB transfer.

17.4.4.2 Read cycles—buffer miss

Read cycles from the flash array are initiated by driving a valid access address on *bkn_fl_addr[23:0]* and asserting *bkn_fl_rd_en* for the required setup (and hold) time before (and after) the rising edge of *hclk*. The PFLASH2P_LCA then waits for the programmed number of read wait states before sampling the read data on *bkn_fl_rdata[127:0]*. This data is normally stored in the least-recently updated page read buffer for banks 0 and 2 in parallel with the requested data being forwarded to the AHB. For bank1, the data is captured in the page-wide temporary holding register as the requested data is forwarded to the AHB bus. Timing diagrams of basic read accesses from the flash array are shown in Figure 17-47 through Figure 17-50.

If the flash access was the direct result of an AHB transaction, the page buffer is marked as most-recently-used as it is being loaded. If the flash access was the result of a speculative prefetch to the next sequential line, it is first loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs.

17.4.4.3 Read cycles—buffer hit

Single cycle read responses to the AHB are possible with the PFLASH2P_LCA when the requested read access was previously loaded into one of the page buffers associated with banks 0 and 2. In these “buffer hit” cases, read data is returned to the AHB data phase with a zero wait-state response.

Likewise, the bank1 logic includes 128-bit temporary holding registers (one per AHB port) and sequential accesses which “hit” in these registers are also serviced with a zero wait-state response.

17.4.4.4 Write cycles

In a write cycle, address, write data, and control signals are launched off the same edge of *hclk* at the completion of the first AHB data phase cycle. Write cycles to the flash array are initiated by driving a valid access address on *bkn_fl_addr[23:0]*, driving write data on *bkn_fl_wdata[63:0]*, and asserting *bkn_fl_wr_en*. Again, the controller drives the address and control information for the required setup time before the rising edge of *hclk*, and provides the required amount of hold time. The PFLASH2P_LCA then waits for the appropriate number of write wait-states before terminating the write operation. On the cycle following the programmed wait state value, the PFLASH2P_LCA asserts *hready_out* to indicate to the AHB port that the cycle has terminated.

17.4.4.5 Error termination

The PFLASH2P_LCA follows the standard procedure when an AHB bus cycle is terminated with an ERROR response. First, the PFLASH2P_LCA asserts *hresp[0]* and negates *hready_out* to signal an error has occurred. On the following clock cycle, the PFLASH2P_LCA asserts *hready_out* and holds both *hresp[0]* and *hready_out* asserted until *hready_in* is asserted.

The first case that can cause an error response to the AHB is when an access is attempted by an AHB master whose corresponding Read Access Control or Write Access Control settings do not allow the access, thus causing a protection violation. In this case, the PFLASH2P_LCA does not initiate a flash array access.

The second case that can cause an error response to the AHB is when an access is performed to the flash array and is terminated with a flash error response. See Section 17.4.4.7, Flash error response operation. This may occur for either a read or a write operation.

The third case that can cause an error response to the AHB is when a write access is attempted to the flash array and is disallowed by the state of the *bkn_fl_ary_access* control input. This case is similar to case 1.

A fourth case involves an attempted read access while the flash array is busy doing a write (program) or erase operation if the appropriate read-while-write control field is programmed for this response. The 3-bit read-while-write control allows for immediate error termination of an attempted read, or various combinations involving stalls with optional notification interrupts while program/erase operations are occurring.

The PFLASH2P_LCA can also terminate the current AHB access if *hready_in* is asserted before the end of the current bus access. While this circumstance should not occur, this does not result in an error condition being reported, as this behavior is initiated by the AHB master. In this circumstance, the

PFLASH2P_LCA control state machine completes any flash array access in progress (without signaling the AHB) before handling a new access request.

17.4.4.6 Access pipelining

The PFLASH2P_LCA controller does not support access pipelining since this capability is not supported by the low-cost flash array. As a result, the APC (Address Pipelining Control) field is typically set to the same value as the RWSC (Read Wait State Control) field for best performance, that is, Bn_APC = Bn_RWSC. It cannot be less than the RWSC.

17.4.4.7 Flash error response operation

The flash array may signal an error response by asserting *bkn_fl_xfr_err* to terminate a requested access with an error. This may occur due to an uncorrectable ECC error, or because of improper sequencing during program/erase operations. When an error response is received, the PFLASH2P_LCA does *not* update or validate a bank 0 or 2 page read buffer nor the bank1 temporary holding register. An error response may be signaled on read or write operations. For more information on the specifics related to signaling of errors, including flash ECC, refer to the low-cost flash array documentation. For additional information on the system registers which capture the faulting address, attributes, data and ECC information, see the ECSM chapter.

17.4.4.8 Bank 0 and 2 page read buffers and prefetch operation

The logic associated with banks 0 and 2 of the PFLASH2P_LCA contains four page read buffers which are used to hold data read from the flash array. Each buffer stores 4 pages (4 x 128b storage) operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

The organization of each page buffer is described below in a pseudo-code representation. The hardware structure includes the buffer address and valid bit, along with 128 bits of page read data and several error flags.

```
struct {                    // bx_py_page_buffer
   reg   addr[23:4];        // page address
   reg   valid;             // valid bit
   reg   rdata[127:0];      // page read data
   reg   xfr_error;         // transfer error indicator from flash array
   reg   multi_ecc_error;   // multi-bit ECC error indicator from flash array
   reg   single_ecc_error; // single-bit correctable ECC indicator from flash array
}  bx_py_page_buffer[4];
```

Given this definition, the PFLASH2P_LCA includes four instantiations of the basic 4 x 128b page buffer. These are named: b0_p0, b0_p1, b2_p0 and b2_p1.

For the general case, a page buffer is written at the completion of an error-free flash access and the valid bit asserted. Subsequent flash accesses that “hit” the buffer, that is, the current access address matches the address stored in the buffer, can be serviced in 0 AHB wait-states as the stored read data is routed from the given page buffer back to the requesting bus master.

As noted in Section 17.4.4.7, Flash error response operation, a page buffer is *not* marked as valid if the flash array access terminated with any type of transfer error. However, the result is that flash array accesses that are tagged with a single-bit correctable ECC event are loaded into the page buffer and validated. For additional comments on this topic, see Section 17.4.4.8.4, Buffer invalidation.

Prefetch triggering is controllable on a per-master and access-type basis. Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access to the PFLASH2P_LCA may trigger a prefetch to the next sequential page of array data on the first idle cycle following the request. The access address is incremented to the next-higher 16-byte boundary, and a flash array prefetch is initiated if the data is not already resident in a page buffer. Prefetched data is always loaded into the least-recently-used buffer.

Buffers may be in one of six states, listed here in prioritized order:

1. Invalid—the buffer contains no valid data
2. Used—the buffer contains valid data which has been provided to satisfy an AHB burst type read
3. Valid—the buffer contains valid data which has been provided to satisfy an AHB single type read
4. Prefetched—the buffer contains valid data which has been prefetched to satisfy a potential future AHB access
5. Busy AHB—the buffer is currently being used to satisfy an AHB burst read
6. Busy Fill—the buffer has been allocated to receive data from the flash array, and the array access is still in progress

Selection of a buffer to be loaded on a miss is based on the following replacement algorithm:

1. First, the buffers are examined to determine if there are any invalid buffers. If there are multiple invalid buffers, the one to be used is selected using a simple numeric priority, where buffer 0 is selected first, then buffer 1, etc.
2. If there are no invalid buffers, the least-recently-used buffer is selected for replacement.

Once the candidate page buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the just-loaded buffer is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Multiple algorithms are available for prefetch control which trade off performance versus power. They are defined by the Bx_Py_PFLM (prefetch limit) register field. More aggressive prefetching increases power slightly due to the number of wasted (discarded) prefetches, but may increase performance by lowering average read latency.

In order for prefetching to occur, a number of control bits must be enabled. Specifically, the global buffer enable (Bx_Py_BFE) must be set, the prefetch limit (Bx_Py_PFLM) must be non-zero and either instruction prefetching (Bx_Py_IPFE) or data prefetching (Bx_Py_DPFE) enabled. Recall the prefetch

and buffer enables are defined on a per AHB port in the PFCR0 and PFCR1 registers. Refer to Section 17.4.3.2, Register descriptions, for a description of these control fields.

17.4.4.8.1 Instruction/data prefetch triggering

Prefetch triggering may be enabled for instruction reads via the Bx_Py_IPFE control field, while prefetching for data reads is enabled via the Bx_Py_DPFE control field. Additionally, the Bx_Py_PFLIM field must also be set to enable prefetching. Prefetches are never triggered by write cycles.

17.4.4.8.2 Per-master prefetch triggering

Prefetch triggering may be also controlled for individual bus masters. AHB accesses indicate the requesting master via the *hmaster[3:0]* inputs. Refer to PFAPR description for details on these controls.

17.4.4.8.3 Buffer allocation

Allocation of the page read buffers is controlled via page buffer configuration (Bx_Py_BCFG) field. This field defines the operating organization of the four page buffers. The buffers can be organized as a “pool” of available resources (with all four buffers in the pool) or with a fixed partition between buffers allocated to instruction or data accesses. For the fixed partition, two configurations are supported. In one configuration, buffers 0 and 1 are allocated for instruction fetches and buffers 2 and 3 for data accesses. In the second configuration, buffers 0, 1 and 2 are allocated for instruction fetches and buffer 3 reserved for data accesses.

17.4.4.8.4 Buffer invalidation

The page read buffers may be invalidated under hardware or software control.

Any falling edge transition of the array’s *bkn_fl_done* signal causes the page read buffers to be marked as invalid. This input is negated by the flash array at the beginning of all program/erase operations as well as in certain other cases. Buffer invalidation occurs at the next AHB non-sequential access boundary, but does not affect a burst from a page read buffer which is in progress.

Software may invalidate the buffers by clearing the Bx_Py_BFE bit, which also disables the buffers. Software may then re-assert the Bx_Py_BFE bit to its previous state, and the buffers will have been invalidated.

One special case needing software invalidation relates to page buffer “hits” on flash data which was tagged with a single-bit ECC event on the original array access. Recall that the page buffer structure includes a status bit signaling the array access detected and corrected a single-bit ECC error. On all subsequent buffer hits to this type of page data, a single-bit ECC event is signaled by the PFLASH2P_LCA. Depending on the specific hardware configuration, this reporting of a single-bit ECC event may generate an ECC alert interrupt. *In order to prevent repeated ECC alert interrupts, the page buffers need to be invalidated by software after the first notification of the single-bit ECC event.*

Finally, the buffers are invalidated by hardware on any non-sequential access with a non-zero value on *haddr[28:24]* to support wait-state emulation.

17.4.4.9 Bank1 temporary holding registers

Recall the bank1 logic within the PFLASH2P_LCA includes two 128-bit data registers (one for each AHB port), used for capturing read data. Since this bank does not support prefetching, the read data for the referenced address is bypassed directly back to the AHB data bus. The page is also loaded into the appropriate temporary data register and subsequent accesses to this page can hit from this register, if it is enabled (B1_Py_BFE).

The organization of the temporary holding register is described below in a pseudo-code representation. The hardware structure includes the buffer address and valid bit, along with 128 bits of page read data and several error flags and is the same as an individual bank 0 or 2 page buffer.

```
struct {                    // b1_py_page_buffer
   reg   addr[23:4];        // page address
   reg   valid;             // valid bit
   reg   rdata[127:0];      // page read data
   reg   xfr_error;         // transfer error indicator from flash array
   reg   multi_ecc_error;   // multi-bit ECC error indicator from flash array
   reg   single_ecc_error; // single-bit correctable ECC indicator from flash array
}  b1_py_page_buffer;
```

Given this definition, the PFLASH2P_LCA includes two instantiations of this temporary holding register for bank 1. These are named: b1_p0 and b1_p1.

For the general case, a temporary holding register is written at the completion of an error-free flash access and the valid bit asserted. Subsequent flash accesses that “hit” the buffer, that is, the current access address matches the address stored in the temporary holding register, can be serviced in 0 AHB wait-states as the stored read data is routed from the temporary register back to the requesting bus master.

The contents of the holding register are invalidated by the falling edge transition of *b1_fl_done* and on any non-sequential access with a non-zero value on *haddr[28:24]* (to support wait-state emulation) in the same manner as the bank0 page buffers. Additionally, the B1_Py_BFE register bit can be cleared by software to invalidate the contents of the holding register.

As noted in Section 17.4.4.7, Flash error response operation, the temporary holding register is *not* marked as valid if the flash array access terminated with any type of transfer error. However, the result is that flash array accesses that are tagged with a single-bit correctable ECC event are loaded into the temporary holding register and validated. Accordingly, one special case needing software invalidation relates to holding register “hits” on flash data which was tagged with a single-bit ECC event. Depending on the specific hardware configuration, the reporting of a single-bit ECC event may generate an ECC alert interrupt. *In order to prevent repeated ECC alert interrupts, the temporary holding registers need to be invalidated by software after the first notification of the single-bit ECC event*.

Each bank1 temporary holding register effectively operates like a single page buffer.

17.4.4.10 Input port arbitration

For maximum system performance, the PFLASH2P_LCA fully supports concurrent flash accesses from the two AHB input ports when the references are targeted to different flash banks. This is expected to be

the typical use-case where AHB p0 (the processor core) mainly accesses bank0 while the non-core AHB masters on p1 mainly reference bank2.

In the event that both AHB ports reference the same flash bank, there is arbitration logic in the module to determine the order the references are granted access to the targeted bank. The 2-bit PFAPR[ARBM] field defines the port arbitration mode and this field can define a fixed priority scheme with either p0 > p1 or p1 > p0 or a round-robin mode where the port given priority simply toggles on every simultaneous bank conflict.

17.4.4.11 Read-While-Write functionality

The PFLASH2P_LCA supports various programmable responses for read accesses while the flash is busy performing a write (program) or erase operation. For all situations, the PFLASH2P_LCA uses the state of the flash array's *bkn_fl_done* output to determine if it is busy performing some type of high-voltage operation, namely, if *bkn_fl_done* = 0, the array is busy.

Specifically, there are two 3-bit read-while-write (Bn_RWWC) control register fields which define the PFLASH2P_LCA's response to these types of access sequences. There are 5 unique responses that are defined by the Bn_RWWC setting: one immediately reports an error on an attempted read and four settings that support various stall-while-write capabilities. Consider the details of these settings.

- Bn_RWWC = 0b0--
  - For this mode, any attempted flash read to a busy array is immediately terminated with an AHB error response and the read is blocked in the controller and not seen by the flash array.
- Bn_RWWC = 0b111
  - This defines the basic stall-while-write capability and represents the default reset setting. For this mode, the PFLASH2P_LCA module simply stalls any read reference until the flash has completed its program/erase operation. If a read access arrives while the array is busy or if a falling-edge on *bkn_fl_done* occurs while a read is still in progress, the AHB data phase is stalled by negating *hready_out* and saving the address and attributes into holding registers. Once the array has completed its program/erase operation, the PFLASH2P_LCA uses the saved address and attribute information to create a pseudo address phase cycle to "retry" the read reference and sends the registered information to the array as *bkn_fl_rd_en* is asserted. Once the retried address phase is complete, the read is processed normally and once the data is valid, it is forwarded to the AHB bus and *hready_out* negated to terminate the system bus transfer.
- Bn_RWWC = 0b110
  - This setting is similar to the basic stall-while-write capability provided when Bn_RWWC = 0b111 with the added ability to generate a notification interrupt if a read arrives while the array is busy with a program/erase operation. There are two notification interrupts, one for each bank.
- Bn_RWWC = 0b101
  - Again, this setting provides the basic stall-while-write capability with the added ability to abort any program/erase operation if a read access is initiated. For this setting, the read request is captured and retried as described for the basic stall-while-write, plus the program/erase operation is aborted by the PFLASH2P_LCA's assertion of the *bkn_fl_abort* signal. The

bkn_fl_abort signal remains asserted until *bkn_fl_done* is driven high. For this setting, there are no notification interrupts generated.

- Bn_RWWC = 0b100
  - This setting provides the basic stall-while-write capability with the ability to abort any program/erase operation if a read access is initiated plus the generation of an abort notification interrupt. For this setting, the read request is captured and retried as described for the basic stall-while-write, the program/erase operation is aborted by the PFLASH2P_LCA's assertion of the *bkn_fl_abort* signal and an abort notification interrupt generated. There are two abort notification interrupts, one for each bank.

As detailed above, there are a total of 4 interrupt requests associated with the stall-while-write functionality. These interrupt requests are captured as part of ECSM's Interrupt Register and logically summed together to form a single request to the interrupt controller.

Table 17-67. PFLASH2P_LCA Stall-While-Write interrupts

MIR[n]	Interrupt Description
ECSM.MIR[7]	Platform flash bank0 abort notification, MIR[FB0AI]
ECSM.MIR[6]	Platform flash bank0 stall notification, MIR[FB0SI]
ECSM.MIR[5]	Platform flash bank1 abort notification, MIR[FB1AI]
ECSM.MIR[4]	Platform flash bank1 stall notification, MIR[FB1S1]

For example timing diagrams of the stall-while-write and abort-while-write operations, see Figure 17-51 and Figure 17-52 respectively.

17.4.4.12 Wait-State emulation

Emulation of other memory array timings are supported by the PFLASH2P_LCA on read cycles to the flash. This functionality may be useful to maintain the access timing for blocks of memory which were used to overlay flash blocks for the purpose of system calibration or tuning during code development.

The PFLASH2P_LCA inserts additional wait-states according to the values of *haddr[28:24]*. When these inputs are non-zero, additional cycles are added to AHB read cycles. Write cycles are not affected. In addition, no page read buffer prefetches are initiated, and buffer hits are ignored.

Table 17-68 and Table 17-69 show the relationship of *haddr[28:24]* to the number of additional primary wait-states. These wait-states are applied to the initial access of a burst fetch or to single-beat read accesses on the AHB system bus.

Note that the wait-state specification consists of two components: *haddr[28:26]* and *haddr[25:24]* and effectively extends the flash read by (8 * *haddr[25:24]* + *haddr[28:26]*) cycles.

Table 17-68. Additional Wait-State encoding

Memory Address haddr[28:26]	Additional wait-states
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Table 17-69 shows the relationship of *haddr[25:24]* to the number of additional wait-states. These are applied in addition to those specified by *haddr[28:26]* and thus extend the total wait-state specification capability.

Table 17-69. Extended additional Wait-State encoding

Memory address haddr[25:24]	Additional Wait-states (added to those specified by haddr[28:26])
00	0
01	8
10	16
11	24

17.4.4.13 Timing diagrams

Since PFLASH2P_LCA controller is typically used in platform configurations with a cacheless core, the operation of the processor accesses to the platform memories, e.g., flash and SRAM, plays a major role in the overall system performance. Given the core/platform pipeline structure, the platform's memory controllers (PFLASH, PRAM) are designed to provide a zero wait-state data phase response to maximize processor performance. The following diagrams illustrate operation of various cycle types and responses referenced earlier in this chapter including stall-while-read (Figure 17-51) and abort-while-read (Figure 17-52) diagrams.

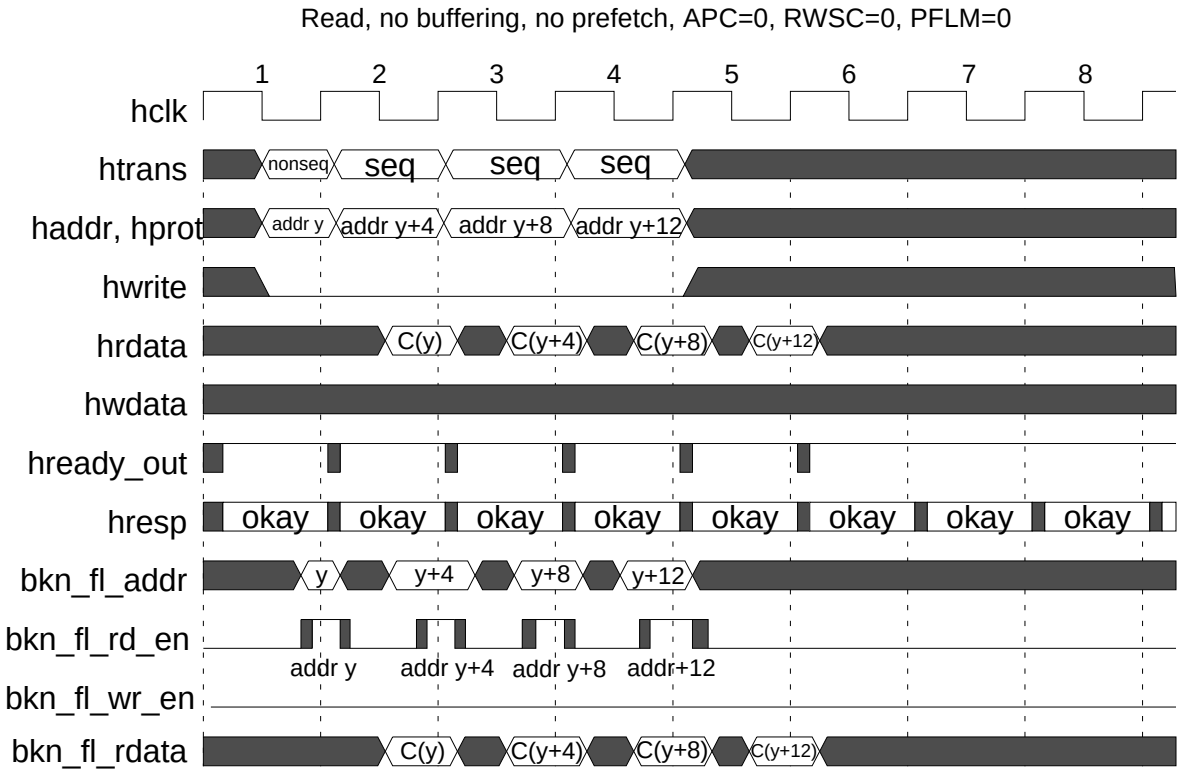


Figure 17-47. 1-cycle access, no buffering, no prefetch

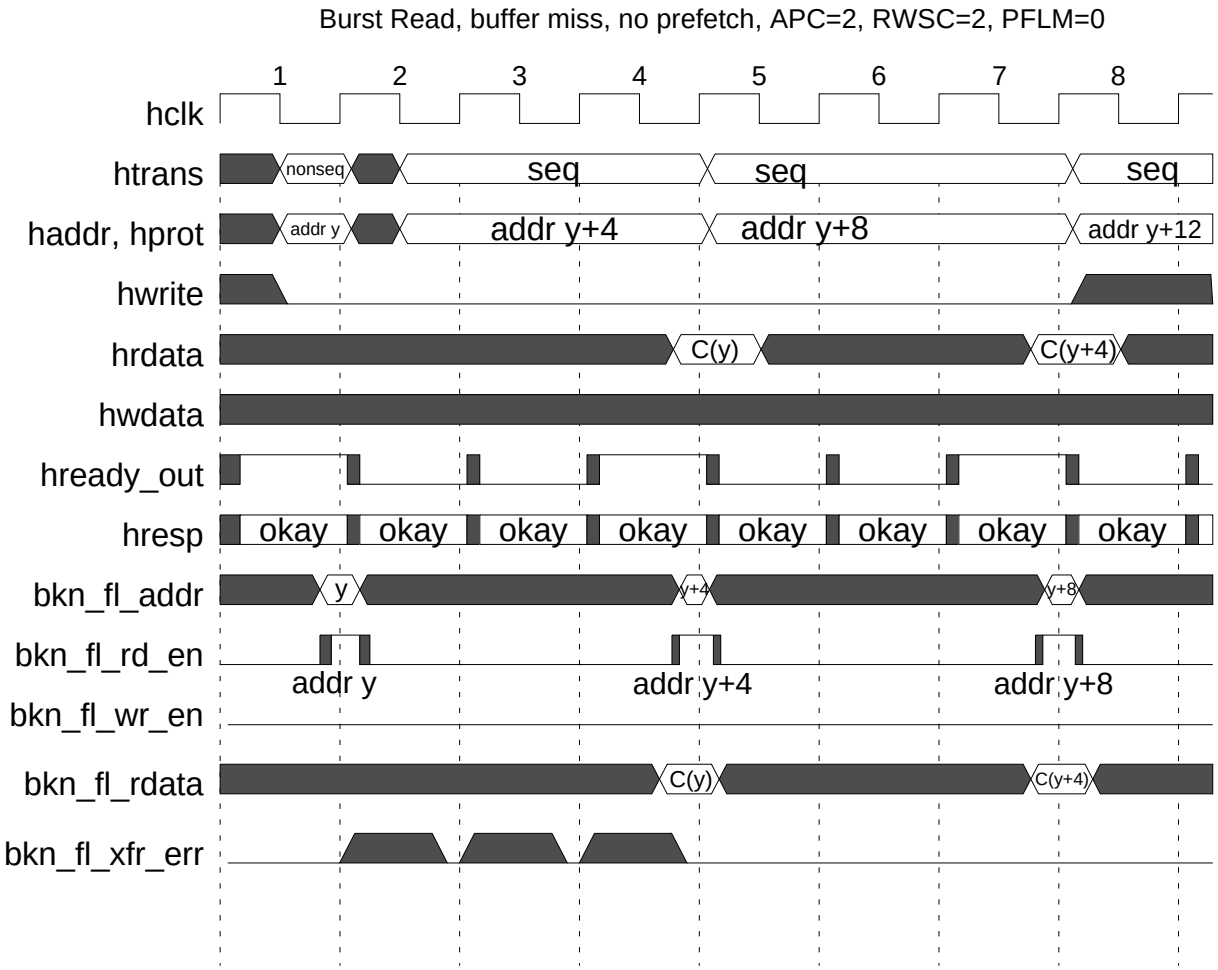


Figure 17-48. 3-cycle access, no prefetch, buffering disabled

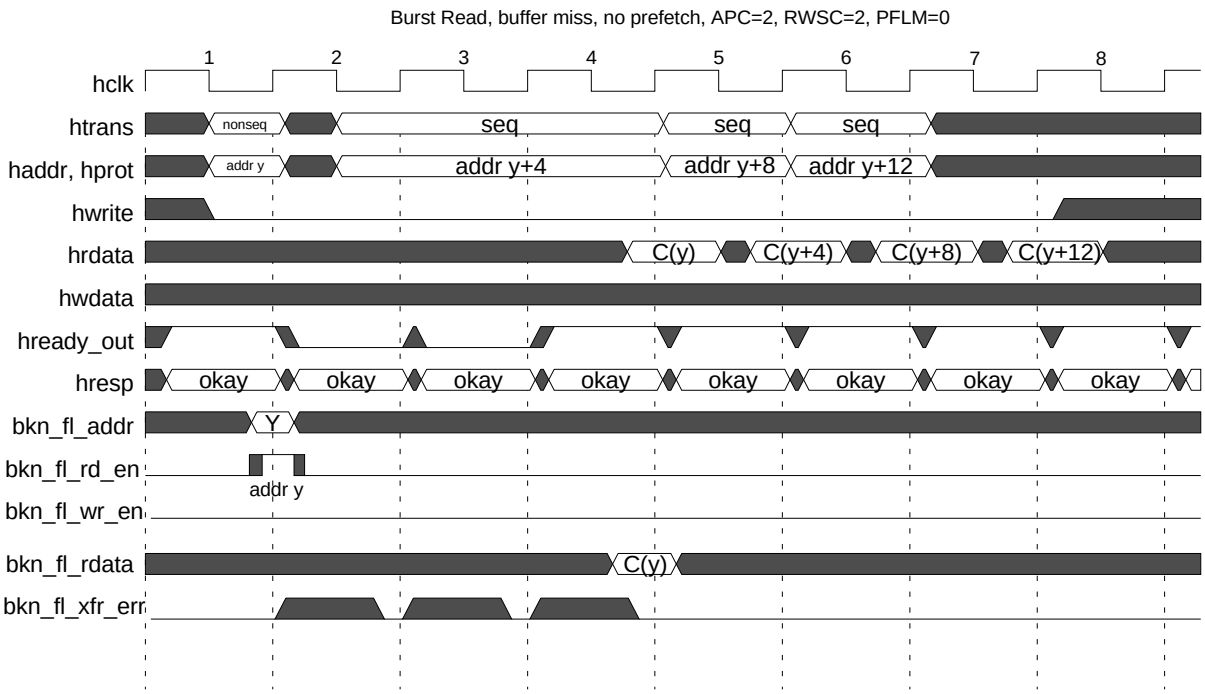


Figure 17-49. 3-cycle access, no prefetch, buffering enabled

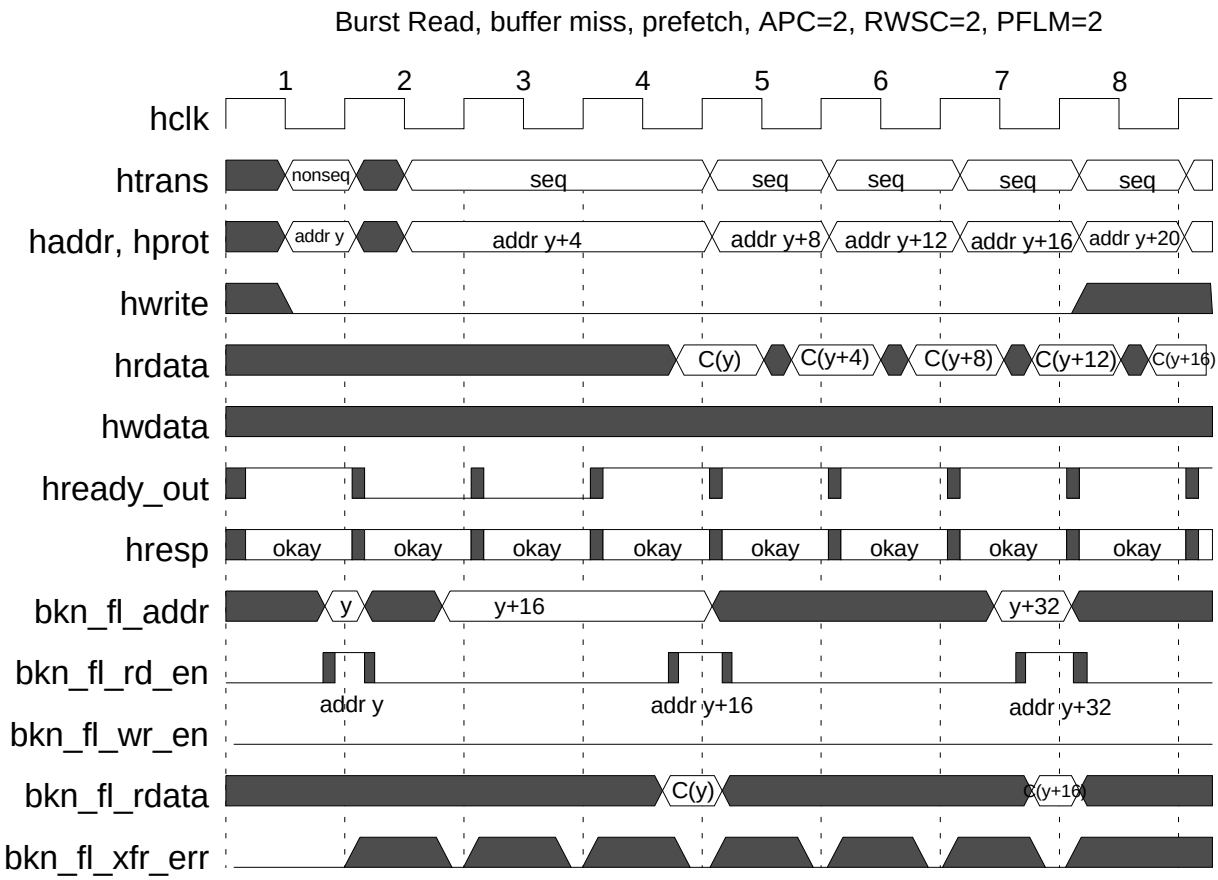


Figure 17-50. 3-cycle access, prefetch and buffering enabled

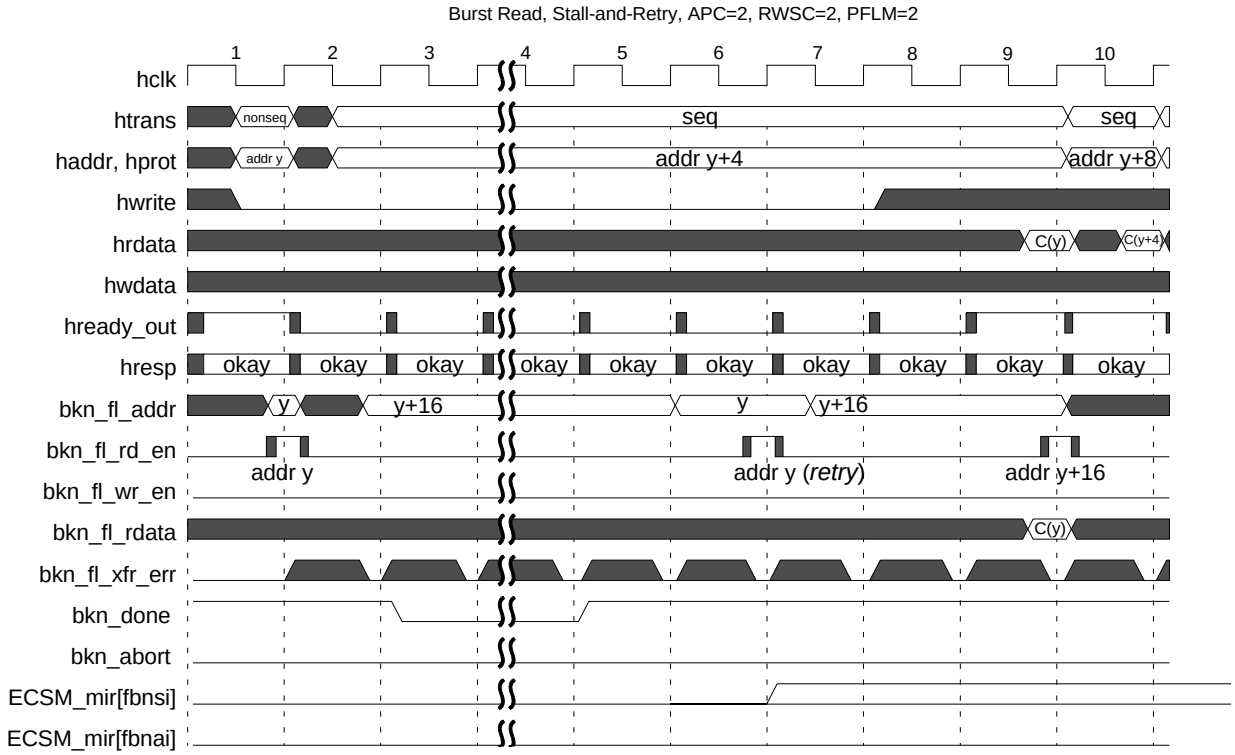


Figure 17-51. 3-cycle access, Stall-and-Retry with Bn_RWWC = 11x

As shown in Figure 17-51, the 3-cycle access to address y is interrupted when an operation causes the *bkn_done* signal to be negated signaling that the array bank is busy with a high-voltage program or erase event. Eventually, this array operation completes (at the end of cycle 4) and *bkn_done* returns to a logical 1. In cycle 6, the PFLASH2P_LCA module retries the read to address y which was interrupted by the negation of *bkn_done* in cycle 3. Note that throughout cycles 2-9, the AHB bus pipeline is stalled with a read to address y in the AHB data phase and a read to address y+4 in the address phase. Depending on the state of the least-significant-bit of the Bn_RWWC control field, the hardware may also signal a stall notification interrupt (if Bn_RWWC = 110). The stall notification interrupt is shown as the optional assertion of ECSM's MIR[FBnSI] (flash bank n stall interrupt).

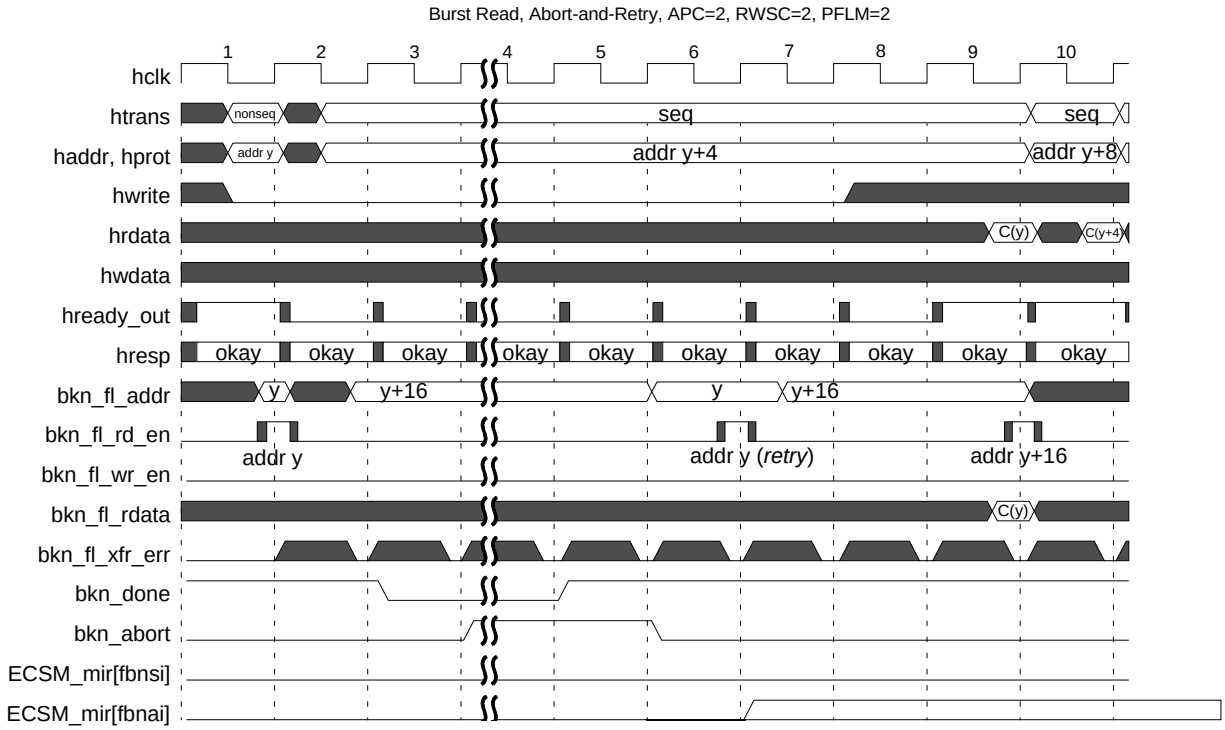


Figure 17-52. 3-cycle access, Abort-and-Retry with Bn_RWWC = 10x

Figure 17-52 shows the abort-while-write timing diagram. In this example, the 3-cycle access to address y is interrupted when an operation causes the *bkn_done* signal to be negated signaling that the array bank is busy with a high-voltage program or erase event. Based on the setting of Bn_RWWC, once the *bkn_done* signal is detected as negated, the PFLASH2P_LCA asserts *bkn_abort* which forces the flash array to cancel the high-voltage program or erase event. The array operation completes (at the end of cycle 4) and *bkn_done* returns to a logical 1. It should be noted that the time spent in cycle 4 for Figure 17-52 is considerably less than the time in the same cycle in Figure 17-51 (because of the abort operation). In cycle 6, the PFLASH2P_LCA module retries the read to address y which was interrupted by the negation of *bkn_done* in cycle 3. Note that throughout cycles 2-9, the AHB bus pipeline is stalled with a read to address y in the AHB data phase and a read to address y+4 in the address phase. Depending on the state of the least-significant-bit of the Bn_RWWC control field, the hardware may also signal an abort notification interrupt (if Bn_RWWC = 100). The stall notification interrupt is shown as the optional assertion of ECSM's MIR[FBnAI] (flash bank n abort interrupt).

17.5 Initialization / application information

17.5.1 Background

Flash array access is relatively slow compared to a full speed system clock based on the PLL. To prevent wait states on every flash access, line buffers are implemented. While wait states are required between the flash array and line buffer, no wait states are required between a line buffer and the system bus. For example, if the core is accessing sequential instructions starting at location 0, the first 32 bits (one line)

fetched will require wait states. The number of wait states is based on system clock frequency. However, subsequent instructions contained in that 128-bit line buffer can be accessed without wait states.

Furthermore, with prefetching configured, the next sequential instructions outside the current line buffer can be prefetched to different line buffer. After fetching all the instructions in current line buffer, the next instruction is fetched for the next line buffer without delay.

Prefetching only helps performance when sequential accesses typically occur, such as for instructions. Since data typically is not arranged sequentially (except for perhaps graphic data) prefetching for data generally is not recommended.

The flash module on this device has two ports. Port 0 is always connected to the core. Port 1 is connected to the other non-core masters (DCU and eDMA).

Configuring the flash bus interface parameters is done by writing to the Platform Flash Configuration Registers PFCR0:1 and Platform Flash Access Protection Register PFAPR.

17.5.2 Flash memory setting recommendations

Table 17-70 provides an example of recommended settings for a common scenario with this device. This example assumes Port 0 (core) instruction accesses are typically sequential, but not data. Port 1 (DCU and eDMA) will not have any instruction accesses. For illustration, this example assumes port 1 accesses have a significant amount of sequential data (such as for graphics) which are larger than a line buffer, so prefetching data would make sense. If graphic data were not in the internal flash, then prefetching data on port 1 would not be expected to be a benefit.

Table 17-70. General flash memory setting recommendations for 64 MHz system clock[1]

Access	Parameter	General Recommendations			
		Code flash (banks 0 and 2) 4 line buffers per port		Data flash (bank 1) 1 line buffer per port	
		Parameter symbol in register PFCR0	Comments	Parameter symbolin register PFCR1	Comments
Port 0 (Core only	Page Buffer Enable	B0_P0_BFE = 1	Enable port's buffers	B1_P0_BFE = 1	Enable port's buffer
	Instruction Prefetch Enable	B0_P0_IPFE = 1	Instructions are mostly sequential, so prefeching can improve performance.	—	—
	Data Prefetch Enable	B0_P0_DPFE = 0	Data accesses are expected to generally be random, not sequential	—	—
	Prefetch Limit	B0_P0_PFLIM = 3	Prefetch on hit or miss	—	—
	Page Buffer Configuration	B0_P0_BCFG = 3	Allocate 3 line buffers for instructions, 1 for data	—	—
Port 1 (DCU, eDMA)	Page Buffer Enable	B0_P1_BFE = 1	Enable port's buffers	B1_P1_BFE = 1	Enable port's buffer
	Instruction Prefetch Enable	B0_P1_IPFE = 0	No instruction access on port 1	—	—
	Data Prefetch Enable	B0_P1_DPFE = 1	Enable prefetching assuming there is significant sequential data	—	—
	Prefetch Limit	B0_P1_PFLIM = 1	Prefetch on miss only (allows more bandwidth for core)	—	—
	Page Buffer Configuration	B0_P1_BCFG = 0	All 4 line buffers available for any access	—	—

Table 17-70. General flash memory setting recommendations for 64 MHz system clock[1] (continued)

Access	Parameter	General Recommendations			
		Code flash (banks 0 and 2) 4 line buffers per port		Data flash (bank 1) 1 line buffer per port	
		Parameter symbol in register PFCR0	Comments	Parameter symbolin register PFCR1	Comments
Array Access (for 64 MHz)	Read Wait States	BK0_RWSC = 2	Values are system clock frequency dependent	BK1_RWSC = 2	Values are system clock frequency dependent
	Write Wait States	BK0_WWSC = 2		BK1_WWSC = 2	
	Adv. Pipeline Ctl.	BK0_APC = 2		BK1_APC = 2	
	Read While Write Ctl.	BK0_RRWC = 0	Terminate RWW attempt with error response. Assumes software must first check if any program or erase commands are in progress.	BK1_RRWC = 0	Terminate RWW attempt with error response. Assumes software must first check if any program or erase commands are in progress.

1 Result value for recommendations in PFCR0 = 0x1084_126E, PFCR1 = 0x1084_0101

Table 17-71 illustrates flash access and protection by master. Note that PFAPR's initial value is loaded from shadow flash location 0x20 3E00 after reset. The "Master" numbers correspond to the crossbar masters, which for this device are:

- Master 0: e200z0 core instructions
- Master 1: e200z0 core data
- Master 2: eDMA
- Master 4: DCU

Table 17-71. Access and protection setting recommendations[1]

Parameter	Parameter Symbol in register PFAPR	Comments
Arbitration mode	ARBM = 3	Start with round-bin (2 or 3).Change to fixed priority if application analysis indicates improved performance.
Master n Prefetch Disable	MnPFD = 0 for core instructions, eDMA and DCU; 1 for core data	Start with allowing prefetching (0) for core instructions since it is expected the core will have mostly sequential instruction accesses. Also allow prefetching for eDMA and DCU, assuming there are large blocks of graphic data accessed.
Master n Access Protection	MnAP = 3 for core data, 1 for core instructions, eDMA and DCU	Assuming only the core will program flash, allow read and write access (3) for the core data bus, but read access only (1) for core instructions, eDMA and DCU.

1 Result value for recommendations in PFAPR = 0x03F2 005D

Chapter 18
FlexCAN

18.1 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. A general block diagram is shown in Figure 18-1, which describes the main subblocks implemented in the FlexCAN module. These include two embedded memories, one for storing Message Buffers (MB) and another one for storing Rx Individual Mask Registers. Support for up to 64 Message Buffers is provided. The functions of the submodules are described in subsequent sections.

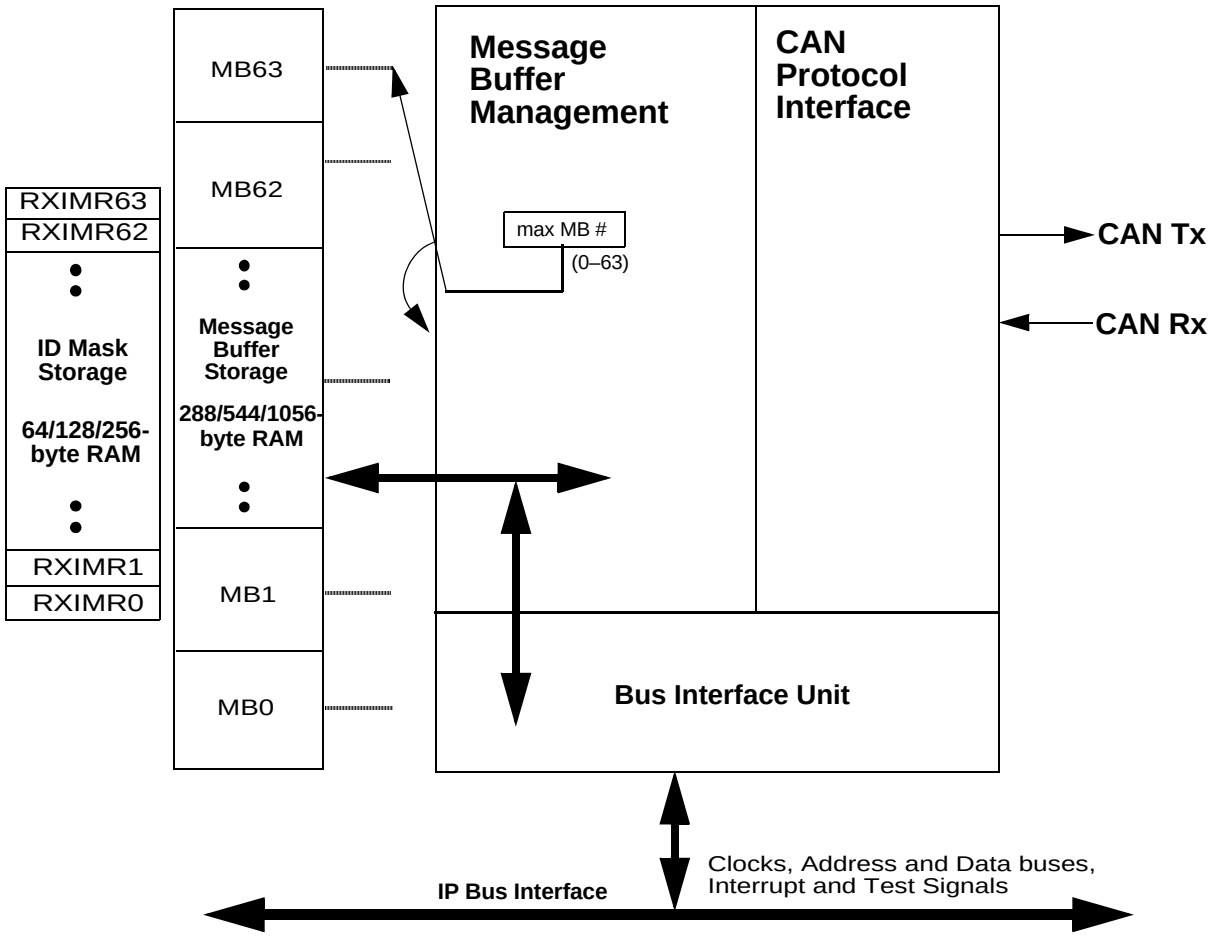


Figure 18-1. FlexCAN block diagram

18.1.1 Overview

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth. The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B [Ref. 1], which supports both standard and extended

message frames. A flexible number of Message Buffers (16, 32, or 64) is also supported. The Message Buffers are stored in an embedded RAM dedicated to the FlexCAN module.

The CAN Protocol Interface (CPI) submodule manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages, and performing error handling. The Message Buffer Management (MBM) submodule handles Message Buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The Bus Interface Unit (BIU) submodule controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs, and test signals are accessed through the Bus Interface Unit.

18.1.2 FlexCAN module features

The FlexCAN module includes these features:

- Full implementation of the CAN protocol specification, Version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - 0 to 8 bytes data length
  - Programmable bit rate up to 1 Mb/s
  - Content-related addressing
- Flexible Message Buffers (up to 64) of 0 to 8 bytes data length
- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask registers per Message Buffer
- Includes either 1056 bytes (64 MBs) of RAM used for MB storage
- Includes either 256 bytes (64 MBs) of RAM used for individual Rx Mask registers
- Full featured Rx FIFO with storage capacity for six frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard, or 32 partial (8 bits) IDs, with individual masking capability
- Selectable backwards compatibility with previous FlexCAN version
- Programmable clock source to the CAN Protocol Interface (CPI), either bus clock or crystal oscillator
- Unused MB and Rx Mask register space can be used as general-purpose RAM space
- Listen-only mode capability
- Programmable loopback mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages

- Low-power modes

18.1.3 Modes of operation

The FlexCAN module has four functional modes: Normal mode (User and Supervisor), Freeze mode, Listen-Only mode and Loopback mode. There is also a low-power mode, Disable mode.

- Normal mode (User or Supervisor):

  In Normal mode, the module operates receiving and/or transmitting message frames. Errors are handled normally and all the CAN protocol functions are enabled. User and Supervisor modes differ in the access to some restricted control registers.
- Freeze mode:

  It is enabled when the FRZ bit in the MCR Register is asserted. If enabled, Freeze mode is entered when the HALT bit in MCR is set or when the MCU is stopped by a debugger. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Section 18.4.9.1, Freeze mode for more information.
- Listen-Only mode:

  The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.
- Loopback mode:

  The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FlexCAN performs an internal loopback that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.
- Module Disable mode:

  This low-power mode is entered when the MDIS bit in the MCR Register is asserted. When disabled, the module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management submodules. Exit from this mode is done by negating the MDIS bit in the MCR Register. See Section 18.4.9.2, Module Disable mode, for more information.

18.2 External signal description

18.2.1 Overview

The FlexCAN module has two I/O signals connected to the external MCU pins. These signals are summarized in Table 18-1 and described in more detail in the next subsections.

Table 18-1. FlexCAN Signals

Signal Name[1]	Direction	Description
CAN Rx	Input	CAN Receive Pin
CAN Tx	Output	CAN Transmit Pin

1 The actual MCU pins may have different names. Please consult the Device User Guide for the actual signal names.

18.2.2 Signal descriptions

18.2.2.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

18.2.2.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

18.3 Memory map and register description

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the MCU. The addresses presented here are relative to the base address.

The address space occupied by FlexCAN has 96 bytes for registers starting at the module base address, followed by MB storage space in embedded RAM starting at address 0x0060, and an extra ID Mask storage space in a separate embedded RAM starting at address 0x0880.

18.3.1 FlexCAN memory mapping

The complete memory map for a FlexCAN module with 64 MB capability is shown in Table 18-2. Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV bit in the MCR Register. These registers are identified as S/U in the access column of Table 18-2.

The IFRH and IMRH registers are considered reserved space when FlexCAN is configured with 16 or 32 MBs. The Rx Global Mask (RXGMASK), Rx Buffer 14 Mask (RX14MASK), and the Rx Buffer 15 Mask (RX15MASK) registers are provided for backwards compatibility, and are not used when the BCC bit in MCR is asserted.

The address ranges 0x0060–0x047F and 0x0880–0x097F are occupied by two separate embedded memories. These two ranges are completely occupied by RAM (1056 and 256 bytes, respectively) only when FlexCAN is configured with 64 MBs. When it is configured with 16 MBs, the memory sizes are 288

and 64 bytes, so the address ranges 0x0180–0x047F and 0x08C0–0x097F are considered reserved space. When it is configured with 32 MBs, the memory sizes are 544 and 128 bytes, so the address ranges 0x0280–0x047F and 0x0900–0x097F are considered reserved space. Furthermore, if the BCC bit in MCR is negated, then the whole Rx Individual Mask Registers address range (0x0880–0x097F) is considered reserved space.

Table 18-2. Module memory map

Address	Use	Access type	Affected by hard reset	Affected by soft reset	Location
Base + 0x0000	Module Configuration (MCR)	S	Yes	Yes	on page 679
Base + 0x0004	Control Register (CTRL)	S/U	Yes	No	on page 683
Base + 0x0008	Free Running Timer (TIMER)	S/U	Yes	Yes	on page 686
Base + 0x000C	Reserved				
Base + 0x0010	Rx Global Mask (RXGMASK)	S/U	Yes	No	on page 687
Base + 0x0014	Rx Buffer 14 Mask (RX14MASK)	S/U	Yes	No	on page 688
Base + 0x0018	Rx Buffer 15 Mask (RX15MASK)	S/U	Yes	No	on page 688
Base + 0x001C	Error Counter Register (ECR)	S/U	Yes	Yes	on page 688
Base + 0x0020	Error and Status Register (ESR)	S/U	Yes	Yes	on page 690
Base + 0x0024	Interrupt Mask Register High (IMRH)	S/U	Yes	Yes	on page 692
Base + 0x0028	Interrupt Mask Register Low (IMRL)	S/U	Yes	Yes	on page 693
Base + 0x002C	Interrupt Flag Register High (IFRH)	S/U	Yes	Yes	on page 694
Base + 0x0030	Interrupt Flag Register Low (IFRL)	S/U	Yes	Yes	on page 694
Base + 0x0034–0x005F	Reserved				
Base + 0x0060–0x007F	Reserved				
Base + 0x0080–0x017F	Message Buffers MB0–MB15	S/U	No	No	—
Base + 0x0180–0x027F	Message Buffers MB16–MB31	S/U	No	No	—
Base + 0x0280–0x047F	Message Buffers MB32–MB63	S/U	No	No	—
Base + 0x0480-087F	Reserved				
Base + 0x0880–0x08BF	Rx Individual Mask Registers RXIMR0–RXIMR15	S/U	No	No	on page 696
Base + 0x08C0–0x08FF	Rx Individual Mask Registers RXIMR16–RXIMR31	S/U	No	No	on page 696
Base + 0x0900–0x097F	Rx Individual Mask Registers RXIMR32–RXIMR63	S/U	No	No	on page 696

The FlexCAN module stores CAN messages for transmission and reception using a Message Buffer structure. Each individual MB is formed by 16 bytes mapped on memory as described in Table 18-3. Table 18-3 shows a Standard/Extended Message Buffer (MB0) memory map, using 16 bytes total space (0x80–0x8F).

Table 18-3. Message Buffer MB0 memory mapping

Address offset	MB field
0x80	Control and Status (C/S)
0x84	Identifier Field
0x88–0x8F	Data Field 0–Data Field 7 (1 byte each)

18.3.2 Message Buffer structure

The Message Buffer structure used by the FlexCAN module is represented in Figure 18-2. Both Extended and Standard frames (29-bit identifier and 11-bit identifier, respectively) used in the CAN specification (Version 2.0 Part B) are represented.

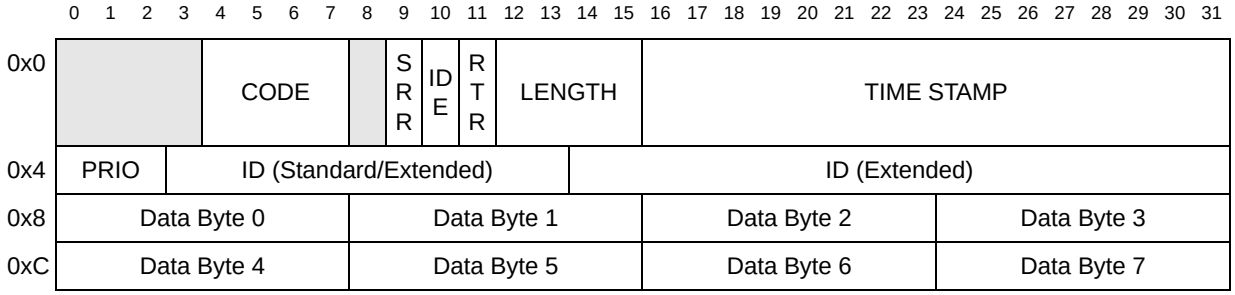


Figure 18-2. Message Buffer Structure

Table 18-4. Message Buffer Structure field descriptions

Field	Description
CODE	Message Buffer Code This 4-bit field can be accessed (read or write) by the CPU and by the Flexcan module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 18-5 and Table 18-6. See Section 18.4, Functional description for additional information.
SRR	Substitute Remote Request Fixed recessive bit, used only in extended format. It must be set to 1 by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as arbitration loss. 0 Dominant is not a valid value for transmission in Extended Format frames 1 Recessive value is compulsory for transmission in Extended Format frames
IDE	ID Extended Bit This bit identifies whether the frame format is standard or extended. 0 Frame format is standard 1 Frame format is extended

Table 18-4. Message Buffer Structure field descriptions (continued)

Field	Description
RTR	Remote Transmission Request This bit is used for requesting transmissions of a data frame. If FlexCAN transmits this bit as 1 (recessive) and receives it as 0 (dominant), it is interpreted as arbitration loss. If this bit is transmitted as 0 (dominant), then if it is received as 1 (recessive), then the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission. 0 Indicates the current MB has a Data frame to be transmitted 1 Indicates the current MB has a Remote frame to be transmitted
LENGTH	Length of Data in Bytes This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see Table 18-2). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the frame to be transmitted is a Remote frame and does not include the data field, regardless of the length field.
TIME STAMP	Free-Running Counter Time Stamp This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.
PRIO	Local priority This 3-bit field is used only when the LPRIO_EN bit is set in MCR and it only makes sense for Tx buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Section 18.4.3, Arbitration process.
ID	Frame Identifier In Standard Frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.
DATA	Data Field Up to 8 bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 18-5. Message Buffer code for Rx buffers

Rx code before Rx new frame	Description	Rx code after Rx new frame	Comment
0000	INACTIVE: MB is not active.	—	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty.	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.

Table 18-5. Message Buffer code for Rx buffers (continued)

Rx code before Rx new frame	Description	Rx code after Rx new frame	Comment
0010	FULL: MB is full.	0010	The act of reading the Control and Status word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the Control and Status word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame is overwritten to this MB before the CPU had time to read it, the code is automatically updated to OVERRUN. Refer to Section 18.4.5, Matching process for details about overrun behavior.
0110	OVERRUN: a frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the CPU reads the Control and Status word and then unlocks the MB, when a new frame is written to the MB the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Section 18.4.5, Matching process for details about overrun behavior.
0XY1[1]	BUSY: Flexcan is updating the contents of the MB. The CPU must not access the MB.	0010	An EMPTY buffer was written with a new frame (XY was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

[1] Note that for Tx MBs (see Table 18-6), the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register.

Table 18-6. Message Buffer Code for Tx Buffers

RTR	Initial Tx code	Code after successful transmission	Description
X	1000	—	INACTIVE: MB does not participate in the arbitration process.
X	1001	—	ABORT: MB was configured as Tx and CPU aborted the transmission. This code is only valid when AEN bit in MCR is asserted. MB does not participate in the arbitration process.
0	1100	1000	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1	1100	0100	Transmit remote frame unconditionally once. After transmission, the MB automatically becomes an Rx MB with the same ID.

Table 18-6. Message Buffer Code for Tx Buffers (continued)

RTR	Initial Tx code	Code after successful transmission	Description
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This MB participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs this MB is allowed to participate in the current arbitration process and the Code field is automatically updated to 1110 to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the Code automatically returns to 1010 to restart the process again.
0	1110	1010	This is an intermediate code that is automatically written to the MB by the MBM as a result of match to a remote request frame. The data frame will be transmitted unconditionally once and then the code will automatically return to 1010. The CPU can also write this code with the same effect.

18.3.3 Rx FIFO Structure

When the FEN bit is set in the MCR, the memory area from 0x80 to 0xFF (which is normally occupied by MBs 0 to 7) is used by the reception FIFO engine. Figure 18-3 shows the Rx FIFO data structure. The region 0x0–0xC contains an MB structure which is the port through which the CPU reads data from the FIFO (the oldest frame received and not read yet). The region 0x10–0xDF is reserved for internal use of the FIFO engine. The region 0xE0–0xFF contains an 8-entry ID table that specifies filtering criteria for accepting frames into the FIFO. Figure 18-4 shows the three different formats that the elements of the ID table can assume, depending on the IDAM field of the MCR. Note that all elements of the table must have the same format. See Section 18.4.7, Rx FIFO for more information.

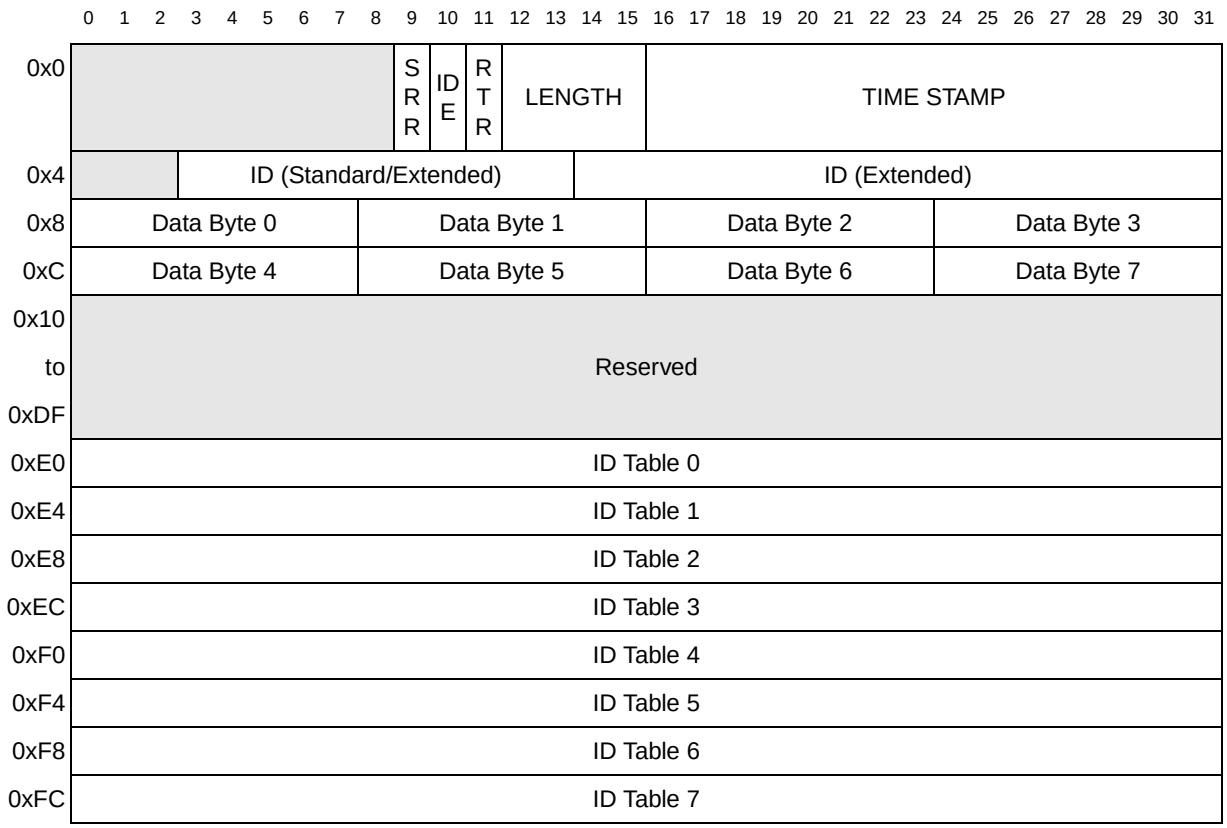


Figure 18-3. Rx FIFO Structure

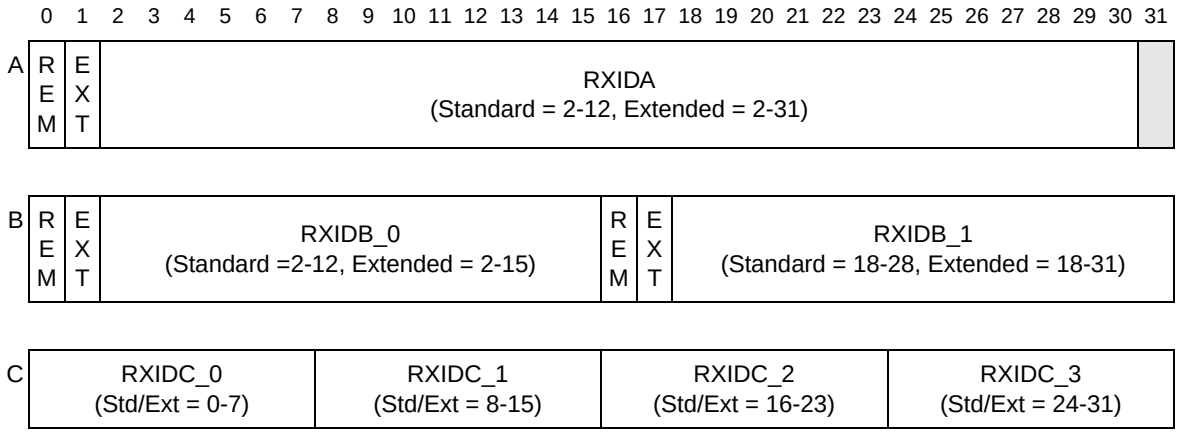


Figure 18-4. ID Table 0–7

Table 18-7. Rx FIFO Structure field descriptions

Field	Description
REM	Remote Frame This bit specifies if Remote frames are accepted into the FIFO if they match the target ID. 0 Remote frames are rejected and data frames can be accepted 1 Remote frames can be accepted and data frames are rejected
EXT	Extended Frame Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID. 0 Extended frames are rejected and standard frames can be accepted 1 Extended frames can be accepted and standard frames are rejected
RXIDA	Rx Frame Identifier (Format A) Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (3 to 13) are used for frame identification. In the extended frame format, all bits are used.
RXIDB_0, RXIDB_1	Rx Frame Identifier (Format B) Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (3 to 13) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.
RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3	Rx Frame Identifier (Format C) Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

18.3.4 Register descriptions

The FlexCAN registers are described in this section in ascending address order.

18.3.4.1 Module Configuration Register (MCR)

This register defines global system configurations, such as the module operation mode (e.g., low-power) and maximum message buffer configuration. Most of the fields in this register can be accessed at any time, except the MAXMB field, which should only be changed while the module is in Freeze mode.

Base + 0x0000

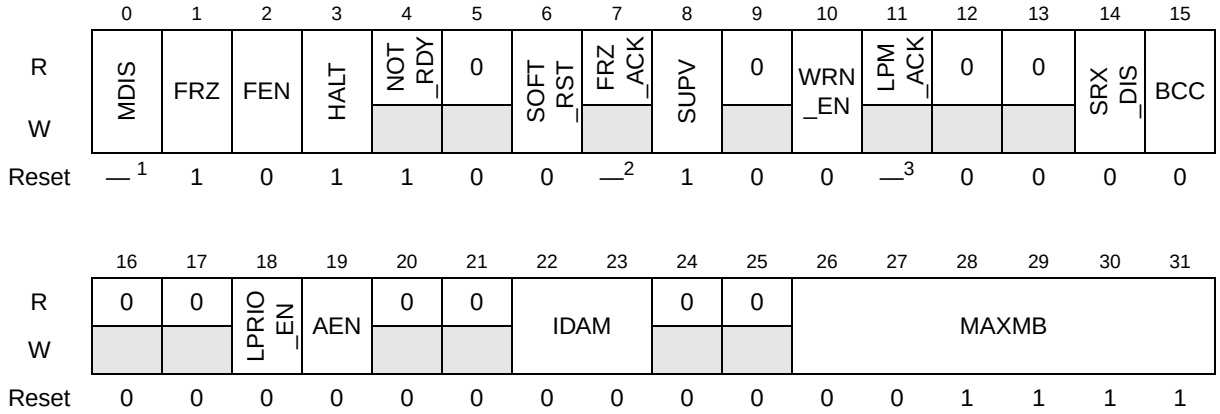


Figure 18-5. Module Configuration Register (MCR)

1 Reset value of this bit is different on various platforms. Consult the specific MCU documentation to determine its value.

2 Different on various platforms, but it is always the opposite of the MDIS reset value.

3 Different on various platforms, but it is always the same as the MDIS reset value.

Table 18-8. MCR field descriptions

Field	Description
MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management submodules. This is the only bit in MCR not affected by soft reset. See Section 18.4.9.2, Module Disable mode, for more information. 0 Enable the FlexCAN module 1 Disable the FlexCAN module
FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR Register is set or when the MCU is stopped by a debugger. When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
FEN	FIFO Enable This bit controls whether the FIFO feature is enabled or not. When FEN is set, MBs 0 to 7 cannot be used for normal reception and transmission because the corresponding memory region (0x80–0xFF) is used by the FIFO engine. See Section 18.3.3, Rx FIFO Structure, and Section 18.4.7, Rx FIFO, for more information. 0 FIFO not enabled 1 FIFO enabled

Table 18-8. MCR field descriptions (continued)

Field	Description
HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. While in Freeze mode, the CPU has write access to the Error Counter Register, which is otherwise read-only. Freeze mode cannot be entered while FlexCAN is in any of the low-power modes. See Section 18.4.9.1, Freeze mode for more information. 0 No Freeze mode request. 1 Enters Freeze mode if the FRZ bit is asserted.
NOT_RDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable mode or Freeze mode. It is negated once FlexCAN has exited these modes. 0 FlexCAN module is in Normal mode, Listen-Only mode, or Loopback mode 1 FlexCAN module is in either Disable mode or Freeze mode
SOFT_RST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory-mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR, IMRL, IMRH, IFRL, and IFRH. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: • CTRL • RXIMR0–RXIMR63 • RXGMASK, RX14MASK, RX15MASK • all Message Buffers The SOFT_RST bit can be asserted directly by the CPU when it writes to the MCR register, but it is also asserted when global soft reset is requested at MCU level. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low-power modes. The module should be first removed from low-power mode, and then soft reset can be applied. 0 No reset request 1 Resets the registers marked as “affected by soft reset” in Table 18-2.
FRZ_ACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze mode request is negated, then this bit is negated once the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in any of the low-power modes, then the FRZ_ACK bit will only be set when the low-power mode is exited. See Section 18.4.9.1, Freeze mode for more information. 0 FlexCAN not in Freeze mode, prescaler running 1 FlexCAN in Freeze mode, prescaler stopped
SUPV	Supervisor Mode This bit configures some of the FlexCAN registers to be either in Supervisor or Unrestricted memory space. The registers affected by this bit are marked as S/U in the Access Type column of Table 18-2. Reset value of this bit is 1, so the affected registers start with Supervisor access restrictions. 0 Affected registers are in Unrestricted memory space 1 Affected registers are in Supervisor memory space—any access without supervisor permission behaves as though the access was done to an unimplemented register location

Table 18-8. MCR field descriptions (continued)

Field	Description
WRN_EN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. 0 TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters 1 TWRN_INT and RWRN_INT bits are set when the respective error counter transition from < 96 to ≥ 96
LPM_ACK	Low-Power Mode Acknowledge This read-only bit indicates that FlexCAN is in Disable mode. This mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPM_ACK bit to know when FlexCAN has actually entered low-power mode. See Section 18.4.9.2, Module Disable mode, for more information. 0 FlexCAN not in any of the low-power modes 1 FlexCAN is either in Disable mode
SRX_DIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. 0 Self reception enabled 1 Self reception disabled
BCC	Backwards Compatibility Configuration This bit is provided to support Backwards Compatibility with previous FlexCAN versions. When this bit is negated, the following configuration is applied: • For MCUs supporting individual Rx ID masking, this feature is disabled. Instead of individual ID masking per MB, FlexCAN uses its previous masking scheme with RXGMASK, RX14MASK, and RX15MASK. • The reception queue feature is disabled. Upon receiving a message, if the first MB with a matching ID that is found is still occupied by a previous unread message, FlexCAN will not look for another matching MB. It will override this MB with the new message and set the CODE field to 0110 (overrun). Upon reset this bit is negated, allowing legacy software to work without modification. 0 Individual Rx masking and queue feature are disabled 1 Individual Rx masking and queue feature are enabled
LPRIO_EN	Local Priority Enable This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to extend the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID is still 11-bit for standard frames and 29-bit for extended frames. 0 Local Priority disabled 1 Local Priority enabled
AEN	Abort Enable This bit is supplied for backwards compatibility reasons. When asserted, it enables the Tx abort feature. This feature guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. 0 Abort disabled 1 Abort enabled

Table 18-8. MCR field descriptions (continued)

Field	Description
IDAM	ID Acceptance Mode This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown in Table 18-9. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section 18.3.3, Rx FIFO Structure.
MAXMB	Maximum Number of Message Buffers This 6-bit field defines the maximum number of message buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field should be changed only while the module is in Freeze mode. **Maximum MBs in use = MAXMB + 1** **Note:** MAXMB has to be programmed with a value smaller than or equal to the number of available Message Buffers; otherwise FlexCAN will not transmit or receive frames. **Note:** When the Rx FIFO is enabled, it uses 8 MBs. These should be included in the MAXMB total. Thus, for example, if the Rx FIFO and 4 other MBs are enabled, MAXMB = 11.

Table 18-9. IDAM coding

IDAM	Format	Explanation
00	A	One full ID (standard or extended) per filter element.
01	B	Two full standard IDs or two partial 14-bit extended IDs per filter element.
10	C	Four partial 8-bit IDs (standard or extended) per filter element.
11	D	All frames rejected.

18.3.4.2 Control Register (CTRL)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loopback mode, Listen-Only mode, Bus Off recovery behavior, and interrupt enabling (Bus-Off, Error, Warning). It also determines the division factor for the clock prescaler. Most of the fields in this register should only be changed while the module is in Disable mode or in Freeze mode. Exceptions are the BOFF_MSK, ERR_MSK, TWRN_MSK, RWRN_MSK, and BOFF_REC bits, which can be accessed at any time.

Address: Base + 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRESDIV								RJW		PSEG1		PSEG2			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BOFF_MSK	ERR_MSK	CLK_SRC	LPB	TWRN_MSK	RWRN_MSK	0	0	SMP	BOFF_REC	TSYN	LBUF	LOM	PROPSEG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 18-6. Control Register (CTRL)

Table 18-10. Control Register (CTRL) field descriptions

Field	Description
0–7 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the CPI clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the CPI clock frequency. The maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the CPI clock frequency divided by 256. For more information refer to Section 18.4.8.4, Protocol timing. **Sclock frequency = CPI clock frequency / (PRESDIV + 1)**
8–9 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta[1] that a bit time can be changed by one resynchronization. The valid programmable values are 0–3. **Resync Jump Width = RJW + 1**
10–12 PSEG1	PSEG1—Phase Segment 1 This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. **Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta**
13–15 PSEG2	PSEG2—Phase Segment 2 This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. **Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta**
16 BOFF_MSK	Bus Off Mask This bit provides a mask for the Bus Off Interrupt. 0 Bus Off interrupt disabled 1= Bus Off interrupt enabled
17 ERR_MSK	Error Mask This bit provides a mask for the Error Interrupt. 0 Error interrupt disabled 1 Error interrupt enabled
18 CLK_SRC	CAN Engine Clock Source This bit selects the clock source to the CAN Protocol Interface (CPI) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock). In order to guarantee reliable operation, this bit should only be changed while the module is in Disable mode. See Section 18.4.8.4, Protocol timing for more information. 0 The CAN engine clock source is the oscillator clock 1 The CAN engine clock source is the bus clock **Note:** This clock selection feature may not be available in all MCUs. A particular MCU may not have a PLL, in which case it would have only the oscillator clock, or it may use only the PLL clock feeding the FlexCAN module. In these cases, this bit has no effect on the module operation.
19 TWRN_MSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated. 0 Tx Warning Interrupt disabled 1 Tx Warning Interrupt enabled

Table 18-10. Control Register (CTRL) field descriptions (continued)

Field	Description
20 RWRN_MSK	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning Interrupt associated with the RWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated. 0 Rx Warning Interrupt disabled 1 Rx Warning Interrupt enabled
21 LPB	Loopback This bit configures FlexCAN to operate in Loopback mode. In this mode, FlexCAN performs an internal loopback that can be used for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loopback disabled 1 Loopback enabled
24 SMP	Sampling Mode This bit defines the sampling mode of CAN bits at the Rx input. 0 Just one sample is used to determine the bit value 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used
25 BOFF_REC	Bus Off Recovery Mode This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits have occurred, then FlexCAN will resynchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B 1 Automatic recovering from Bus Off state disabled
26 TSYN	Timer Sync Mode This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special SYNC message (such as global network time). If the FEN bit in MCR is set (FIFO enabled), MB8 is used for timer synchronization instead of MB0. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
27 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIO_EN bit does not affect the priority arbitration. 0 Buffer with highest priority is transmitted first 1 Lowest number buffer is transmitted first

Table 18-10. Control Register (CTRL) field descriptions (continued)

Field	Description
28 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. 0 Listen-Only mode is deactivated 1 FlexCAN module operates in Listen-Only mode
29-31 PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

1 One time quantum is equal to the Sclock period.

18.3.4.3 Free Running Timer (TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0000 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During Freeze mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

Address: Base + 0x0008 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIMER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-7. Free Running Timer (TIMER)

18.3.4.4 Rx Global Mask (RXGMASK)

This register is provided for legacy support and for low-cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RXGMASK Register to have no effect on the module operation. For MCUs not supporting individual masks per MB, this register is always effective.

RXGMASK is used as acceptance mask for all Rx MBs, excluding MBs 14–15, which have individual mask registers. When the FEN bit in MCR is set (FIFO enabled), the RXGMASK also applies to all elements of the ID filter table, except elements 6-7, which have individual masks.

The contents of this register must be programmed while the module is in Freeze mode, and must not be modified when the module is transmitting or receiving frames.

When used with the FEN bit set, there is a misalignment between the position of the ID field in the Rx MB and in RXIDA, RXIDB, and RXIDC fields of the ID tables. In fact, the RXIDA filter in the ID tables is shifted one bit to the left from the Rx MBs ID position as shown below:

- Rx MB ID = bits 3–31 of ID word corresponding to message ID bits 0–28
- RXIDA = bits 2–30 of ID Table corresponding to message ID bits 0–28

Note that for the mask bits, one-to-one correspondence occurs with the filter bits, not with the incoming message ID bits.

This leads the RXGMASK to affect Rx MB and Rx FIFO filtering in different ways.

For example, if the user intends to mask out bit 24 of the ID filter of Message Buffers, then the RXGMASK will be configured as 0xFFFF_FFEF. As result, bit 24 of the ID field of the incoming message will be ignored during the filtering process for Message Buffers. This very same configuration of RXGMASK would lead bit 24 of RXIDA to be a don’t-care bit, and thus bit 25 of the ID field of the incoming message would be ignored during the filtering process for the Rx FIFO.

Similarly, both RXIDB and RXIDC filters have multiple misalignments with regards to position of the ID field in Rx MBs, which can lead to erroneous masking during filtering process for either Rx FIFO or MBs.

Address: Base + 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 18-8. Rx Global Mask Register (RXGMASK)

Table 18-11. Rx Global Mask Register (RXGMASK) field descriptions

Field	Description
0-31 MI31–MI0	Mask Bits For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR). 0 the corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked against the one received

18.3.4.5 Rx 14 Mask (RX14MASK)

This register is provided for legacy support and for low-cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RX14MASK Register to have no effect on the module operation.

RX14MASK is used as acceptance mask for the identifier in Message Buffer 14. When the FEN bit in MCR is set (FIFO enabled), the RXG14MASK also applies to element 6 of the ID filter table. This register has the same structure as the Rx Global Mask Register. It must be programmed while the module is in Freeze mode, and must not be modified when the module is transmitting or receiving frames.

- Address offset: 0x14
- Reset value: 0xFFFF_FFFF

See Section 18.3.4.4, Rx Global Mask (RXGMASK) for more information on mask alignment.

18.3.4.6 Rx 15 Mask (RX15MASK)

This register is provided for legacy support and for low-cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RX15MASK Register to have no effect on the module operation.

When the BCC bit is negated, RX15MASK is used as acceptance mask for the Identifier in Message Buffer 15. When the FEN bit in MCR is set (FIFO enabled), the RXG14MASK also applies to element 7 of the ID filter table. This register has the same structure as the Rx Global Mask Register. It must be programmed while the module is in Freeze mode, and must not be modified when the module is transmitting or receiving frames.

- Address offset: 0x18
- Reset value: 0xFFFF_FFFF

See Section 18.3.4.4, Rx Global Mask (RXGMASK) for more information on mask alignment.

18.3.4.7 Error Counter Register (ECR)

This register has two 8-bit fields reflecting the value of two FlexCAN error counters: Transmit Error Counter (Tx_Err_Counter field) and Receive Error Counter (Rx_Err_Counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read-only, except in Freeze mode where they can be written by the CPU.

Writing to the Error Counter Register while in Freeze mode is an indirect operation. The data is first written to an auxiliary register. Then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

FlexCAN responds to any bus state as described in the protocol, for example, transmit the Error Active or Error Passive flag, delay its transmission start time (Error Passive), and avoid any influence on the bus when in Bus Off state. The following are the basic rules for FlexCAN bus state transitions.

- If the value of Tx_Err_Counter or Rx_Err_Counter increases to be greater than or equal to 128, the FLT_CONF field in the Error and Status Register is updated to reflect the Error Passive state.
- If the FlexCAN state is Error Passive, and either Tx_Err_Counter or Rx_Err_Counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the Error and Status Register is updated to reflect the Error Active state.
- If the value of Tx_Err_Counter increases to be greater than 255, the FLT_CONF field in the Error and Status Register is updated to reflect the Bus Off state, and an interrupt may be issued. The value of Tx_Err_Counter is then reset to zero.
- If FlexCAN is in the Bus Off state, then Tx_Err_Counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Err_Counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, the FLT_CONF field in the Error and Status Register is updated to be Error Active, and both error counters are reset to zero. At any instance of a dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the Tx_Err_Counter value.
- If during system startup only one node is operating, then its Tx_Err_Counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the Error and Status Register). After the transition to the Error Passive state, the Tx_Err_Counter does not increment anymore by acknowledge errors. Therefore, the device never goes to the Bus Off state.
- If the Rx_Err_Counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume the Error Active state.

Address: Base + 0x001C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Rx_Err_Counter								Tx_Err_Counter							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-9. Error Counter Register (ECR)

18.3.4.8 Error and Status Register (ESR)

This register reflects various conditions and some general status of the device, and it is the source of four interrupts to the CPU. The reported error conditions (bits 16–21) are those that occurred since the last time the CPU read this register. The CPU read action clears bits 16–21. Bits 22–28 are status bits.

Most bits in this register are read only, except TWRN_INT, RWRN_INT, BOFF_INT, and ERR_INT, which are interrupt flags that can be cleared by writing 1 to them (writing 0 has no effect). See Section 18.4.10, Interrupts, for more details.

Address: Base + 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TWRN_INT	RWRN_INT
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1_ERR	BIT0_ERR	ACK_ERR	CRC_ERR	FRM_ERR	STF_ERR	TX_WRN	RX_WRN	IDLE	TXRX	FLT_CONF		0	BOFF_INT	ERR_INT	WAK_INT
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-10. Error and Status Register (ESR)

Table 18-12. Error and Status Register (ESR) field descriptions

Field	Description
TWRN_INT	Tx Warning Interrupt Flag If the WRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transitions from 0 to 1, meaning that the Tx error counter has reached 96. If the corresponding mask bit in the Control Register (TWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing 1 to it. Writing 0 has no effect. 0 No such occurrence 1 The Tx error counter transitioned from < 96 to ≥ 96
RWRN_INT	Rx Warning Interrupt Flag If the WRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing 1 to it. Writing 0 has no effect. 0 No such occurrence 1 The Rx error counter transitioned from < 96 to ≥ 96

Table 18-12. Error and Status Register (ESR) field descriptions (continued)

Field	Description
BIT1_ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence 1 At least one bit sent as recessive is received as dominant **Note:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.
BIT0_ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence 1 At least one bit sent as dominant is received as recessive
ACK_ERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node; that is, a dominant bit has not been detected during the ACK SLOT. 0 No such occurrence 1 An ACK error occurred since last read of this register
CRC_ERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node; that is, the calculated CRC is different from the received. 0 No such occurrence 1 A CRC error occurred since last read of this register
FRM_ERR	Form Error This bit indicates that a Form Error has been detected by the receiver node; that is, a fixed-form bit field contains at least one illegal bit. 0 No such occurrence 1 A Form Error occurred since last read of this register
STF_ERR	Stuffing Error This bit indicates that a Stuffing Error has been detected. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register
TX_WRN	TX Error Counter This bit indicates when repetitive errors are occurring during message transmission. 0 No such occurrence 1 TX_Err_Counter $\geq$ 96
RX_WRN	Rx Error Counter This bit indicates when repetitive errors are occurring during message reception. 0 No such occurrence 1 Rx_Err_Counter $\geq$ 96
IDLE	CAN bus Idle state This bit indicates when CAN bus is in Idle state. 0 No such occurrence 1 CAN bus is now idle

Table 18-12. Error and Status Register (ESR) field descriptions (continued)

Field	Description
TXRX	Current FlexCAN status (transmitting/receiving) This bit indicates if FlexCAN is transmitting or receiving a message when the CAN bus is not in Idle state. This bit has no meaning when IDLE is asserted. 0 FlexCAN is receiving a message (IDLE=0) 1 FlexCAN is transmitting a message (IDLE=0)
FLT_CONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module, as shown in Table 18-13. If the LOM bit in the Control Register is asserted, the FLT_CONF field will indicate the Error Passive state. Since the Control Register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted.
BOFF_INT	Bus Off Interrupt This bit is set when FlexCAN enters 'Bus Off' state. If the corresponding mask bit in the Control Register (BOFF_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing 1 to it. Writing 0 has no effect. 0 No such occurrence 1 FlexCAN module entered the Bus Off state
ERR_INT	Error Interrupt This bit indicates that at least one of the Error Bits (bits 16-21) is set. If the corresponding mask bit in the Control Register (ERR_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing 1 to it. Writing 0 has no effect. 0 No such occurrence 1 Indicates setting of any Error Bit in the Error and Status Register

Table 18-13. Fault confinement state

Value	Meaning
00	Error Active
01	Error Passive
1X	Bus Off

18.3.4.9 Interrupt Mask Register High (IMRH)

This register allows enabling or disabling any number of a range of 32 Message Buffer interrupts. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (for example, when the corresponding IFRH bit is set).

Base + 0x0024

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUF 63M	BUF 62M	BUF 61M	BUF 60M	BUF 59M	BUF 58M	BUF 57M	BUF 56M	BUF 55M	BUF 54M	BUF 53M	BUF 52M	BUF 51M	BUF 50M	BUF 49M	BUF 48M
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF 47M	BUF 46M	BUF 45M	BUF 44M	BUF 43M	BUF 42M	BUF 41M	BUF 40M	BUF 39M	BUF 38M	BUF 37M	BUF 36M	BUF 35M	BUF 34M	BUF 33M	BUF 32M
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-11. Interrupt Mask Register High (IMRH)

Table 18-14. IMRH field descriptions

Field	Description
BUF63M – BUF32M	**Buffer MB_i Mask** Each bit enables or disables the respective FlexCAN Message Buffer (MB32 to MB63) interrupt. 0 The corresponding buffer interrupt is disabled 1 The corresponding buffer interrupt is enabled **Note:** Setting or clearing a bit in the IMRH register can assert or negate an interrupt request, if the corresponding IFRH bit is set.

18.3.4.10 Interrupt Mask Register Low (IMRL)

This register allows enabling or disabling any number of a range of 32 Message Buffer interrupts. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (for example, when the corresponding IFRL bit is set).

Address: Base + 0x0028 Access: User read/write

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

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

Figure 18-12. Interrupt Mask Register Low (IMRL)

Table 18-15. IMRL field descriptions

Field	Description
BUF31M – BUF0M	BUF31M–BUF0M — Buffer MB_i Mask Each bit enables or disables the respective FlexCAN Message Buffer (MB0 to MB31) interrupt. 0 The corresponding buffer interrupt is disabled 1 The corresponding buffer interrupt is enabled **Note:** Setting or clearing a bit in the IMRL register can assert or negate an interrupt request, if the corresponding IFRL bit is set.

18.3.4.11 Interrupt Flag Register High (IFRH)

This register defines the flags for 32 Message Buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFRH bit. If the corresponding IMRH bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing a 1 to it. Writing 0 has no effect.

When the AEN bit in the MCR is set (Abort enabled), while the IFRH bit is set for a MB configured as Tx, then the writing access done by the CPU into the corresponding MB will be blocked.

Address: Base + 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUF 63I	BUF 62I	BUF 61I	BUF 60I	BUF 59I	BUF 58I	BUF 57I	BUF 56I	BUF 55I	BUF 54I	BUF 53I	BUF 52I	BUF 51I	BUF 50I	BUF 49I	BUF 48I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF 47I	BUF 46I	BUF 45I	BUF 44I	BUF 43I	BUF 42I	BUF 41I	BUF 40I	BUF 39I	BUF 38I	BUF 37I	BUF 36I	BUF 35I	BUF 34I	BUF 33I	BUF 32I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-13. Interrupt Flag Register High (IFRH)

Table 18-16. IFRH field descriptions

Field	Description
BUF32I – BUF63I	Buffer MB_i Interrupt Each bit flags the respective FlexCAN Message Buffer (MB32 to MB63) interrupt. 0 No such occurrence 1 The corresponding buffer has successfully completed transmission or reception

18.3.4.12 Interrupt Flag Register Low (IFRL)

This register defines the flags for 32 Message Buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFRL bit. If the corresponding IMRL bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing a 1 to it. Writing 0 has no effect.

When the AEN bit in the MCR is set (Abort enabled), while the IFRL bit is set for an MB configured as Tx, the writing access done by the CPU into the corresponding MB will be blocked.

When the FEN bit in the MCR is set (FIFO enabled), the function of the eight least significant interrupt flags (BUF7I–BUF0I) is changed to support the FIFO operation. BUF7I, BUF6I, and BUF5I indicate operating conditions of the FIFO, while BUF4I to BUF0I are not used.

Address: Base + 0x0030 Access: User read/write

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

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

Figure 18-14. Interrupt Flag Register Low (IFRL)

Table 18-17. IFRL field descriptions

Field	Description
BUF31I–BUF8I	Buffer MB_i Interrupt Each bit flags the respective FlexCAN Message Buffer (MB8 to MB31) interrupt. 0 No such occurrence 1 The corresponding MB has successfully completed transmission or reception
BUF7I	Buffer MB7 Interrupt or FIFO Overflow If the FIFO is not enabled, this bit flags the interrupt for MB7. If the FIFO is enabled, this flag indicates an overflow condition in the FIFO (frame lost because FIFO is full). 0 No such occurrence 1 MB7 completed transmission/reception or FIFO overflow
BUF6I	Buffer MB6 Interrupt or FIFO Warning If the FIFO is not enabled, this bit flags the interrupt for MB6. If the FIFO is enabled, this flag indicates that four out of six buffers of the FIFO are already occupied (FIFO almost full). 0 No such occurrence 1 MB6 completed transmission/reception or FIFO almost full
BUF5I	Buffer MB5 Interrupt or Frames available in FIFO If the FIFO is not enabled, this bit flags the interrupt for MB5. If the FIFO is enabled, this flag indicates that at least one frame is available to be read from the FIFO. 0 No such occurrence 1 MB5 completed transmission/reception or frames available in the FIFO
BUF4I–BUF0I	Buffer MB_i Interrupt or Reserved If the FIFO is not enabled, these bits flag the interrupts for MB0 to MB4. If the FIFO is enabled, these flags are not used and must be considered as reserved locations. 0 No such occurrence 1 Corresponding MB completed transmission/reception

18.3.4.13 Rx Individual Mask Registers (RXIMR0–RXIMR63)

These registers are used as acceptance masks for ID filtering in Rx MBs and the FIFO. If the FIFO is not enabled, one mask register is provided for each available Message Buffer, providing ID masking capability on a per Message Buffer basis. When the FIFO is enabled (FEN bit in MCR is set), the first eight Mask Registers apply to the eight elements of the FIFO filter table (on a one-to-one correspondence), while the rest of the registers apply to the regular MBs, starting from MB8.

The Individual Rx Mask Registers are implemented in RAM, so they are not affected by reset and must be explicitly initialized prior to any reception. Furthermore, they can only be accessed by the CPU while the module is in Freeze mode. Out of Freeze mode, write accesses are blocked and read accesses will return all zeroes. Furthermore, if the BCC bit in the MCR Register is negated, any read or write operation to these registers results in an access error.

Address: See Table 18-19 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-15. Rx Individual Mask Registers (RXIMR0–RXIMR63)

Table 18-18. Rx Individual Mask Registers (RXIMR0–RXIMR63) field descriptions

Field	Description
0-31 MI31–MI0	Mask Bits For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR). 0 The corresponding bit in the filter is a don't-care bit 1 The corresponding bit in the filter is checked against the one received

Table 18-19. RXIMR0–RXIMR31 addresses

Address	Register	Address	Register
Base + 0x0880	RXIMR0	Base + 0x08C0	RXIMR16
Base + 0x0884	RXIMR1	Base + 0x08C4	RXIMR17
Base + 0x0888	RXIMR2	Base + 0x08C8	RXIMR18
Base + 0x088C	RXIMR3	Base + 0x08CC	RXIMR19
Base + 0x0890	RXIMR4	Base + 0x08D0	RXIMR20
Base + 0x0894	RXIMR5	Base + 0x08D4	RXIMR21

Table 18-19. RXIMR0–RXIMR31 addresses (continued)

Address	Register	Address	Register
Base + 0x0898	RXIMR6	Base + 0x08D8	RXIMR22
Base + 0x089C	RXIMR7	Base + 0x08DC	RXIMR23
Base + 0x08A0	RXIMR8	Base + 0x08E0	RXIMR24
Base + 0x08A4	RXIMR9	Base + 0x08E4	RXIMR25
Base + 0x08A8	RXIMR10	Base + 0x08E8	RXIMR26
Base + 0x08AC	RXIMR11	Base + 0x08EC	RXIMR27
Base + 0x08B0	RXIMR12	Base + 0x08F0	RXIMR28
Base + 0x08B4	RXIMR13	Base + 0x08F4	RXIMR29
Base + 0x08B8	RXIMR14	Base + 0x08F8	RXIMR30
Base + 0x08BC	RXIMR15	Base + 0x08FC	RXIMR31

18.4 Functional description

18.4.1 Overview

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed of a set of up to 64 Message Buffers (MB) that store configuration and control data, time stamp, message ID, and data (see Section 18.3.2, Message Buffer structure). The memory corresponding to the first eight MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to eight extended IDs or 16 standard IDs or 32 8-bit ID slices), each one with its own individual mask register. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by three local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be active at a given time if it can participate in the matching and arbitration algorithms that are happening at that time. An Rx MB with a 0000 code is inactive (refer to Table 18-5). Similarly, a Tx MB with a 1000 or 1001 code is also inactive (refer to Table 18-6). An MB not programmed with 0000, 1000, or 1001 will be temporarily deactivated (will not participate in the current arbitration or matching run) when the CPU writes to the Control and Status field of that MB (see Section 18.4.6.2, Message Buffer deactivation).

18.4.2 Transmit process

In order to transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. If the MB is active (transmission pending), write an ABORT code (1001) to the Code field of the Control and Status word to request an abortion of the transmission, then read back the Code field and the IFRL/IFRH registers to check if the transmission was aborted (see Section 18.4.6.1, Transmission abort mechanism). If backwards compatibility is desired (AEN in MCR negated), just write 1000 to the Code field to inactivate the MB, but be aware that in this case the pending frame may be transmitted without notification (see Section 18.4.6.2, Message Buffer deactivation).
2. Write the ID word.
3. Write the data bytes.
4. Write the Length, Control, and Code fields of the Control and Status word to activate the MB.

Once the MB is activated in the fourth step, it will participate in the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the Code field in the Control and Status word is updated, a status flag is set in the Interrupt Flag Register, and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new Code field after transmission depends on the code that was used to activate the MB in step four (see Table 18-5 and Table 18-6 in Section 18.3.2, Message Buffer structure). When the Abort feature is enabled (AEN in MCR is asserted), after the Interrupt Flag is asserted for a MB configured as transmit buffer, the MB is blocked—therefore the CPU is not able to update it until the Interrupt Flag is negated by the CPU. It means that the CPU must clear the corresponding interrupt flag bit before starting to prepare this MB for a new transmission or reception.

18.4.3 Arbitration process

The arbitration process is an algorithm executed by the MBM that scans the whole MB memory looking for the highest priority message to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID[1] or the lowest MB number or the highest priority, depending on the LBUF and LPRIO_EN bits on the Control Register. The arbitration process is triggered in the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame
- During Intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the CPU wrote to the Control and Status word of any MB after the previous arbitration finished
- When MBM is in Idle or Bus Off state and the CPU writes to the Control and Status word of any MB
- Upon leaving Freeze mode

When LBUF is asserted, the LPRIO_EN bit has no effect and the lowest number buffer is transmitted first. When LBUF and LPRIO_EN are both negated, the MB with the lowest ID is transmitted first. However,

1. Actually, if LBUF is negated, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

if LBUF is negated and LPRIO_EN is asserted, then the PRIO bits augment the ID used during the arbitration process. With this extended ID concept, arbitration is done based on the full 32-bit ID, and the PRIO bits define which MB should be transmitted first—therefore MBs with PRIO = 000 have higher priority. If two or more MBs have the same priority, the regular ID will determine the priority of transmission. If two or more MBs have the same priority (three extra bits) and the same regular ID, the lowest MB will be transmitted first.

Once the highest priority MB is selected, it is transferred to a temporary storage space called Serial Message Buffer (SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out, and after it is done write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Halt or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (Data Length Code) value is bigger.

18.4.4 Receive process

To be able to receive CAN frames in the mailbox MBs, the CPU must prepare one or more Message Buffers for reception by executing the following steps:

1. If the MB has a pending transmission, write an ABORT code (1001) to the Code field of the Control and Status word to request an abortion of the transmission, then read back the Code field and the IFRL/IFRH registers to check if the transmission was aborted (see Section 18.4.6.1, Transmission abort mechanism). If backwards compatibility is desired (AEN in MCR negated), just write 1000 to the Code field to inactivate the MB, but be aware that in this case the pending frame may be transmitted without notification (see Section 18.4.6.2, Message Buffer deactivation). If the MB is already programmed as a receiver, just write 0000 to the Code field of the Control and Status word to keep the MB inactive.
2. Write the ID word.
3. Write 0100 to the Code field of the Control and Status word to activate the MB.

Once the MB is activated in the third step, it will be able to receive frames that match the programmed ID. At the end of a successful reception, the MB is updated by the MBM as follows:

1. The value of the Free Running Timer is written into the Time Stamp field.
2. The received ID, Data (8 bytes at most), and Length fields are stored.
3. The Code field in the Control and Status word is updated (see Table 18-5 and Table 18-6 in Section 18.3.2, Message Buffer structure).
4. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

Upon receiving the MB interrupt, the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (mandatory – activates an internal lock for this buffer).
2. Read the ID field (optional – needed only if a mask was used).
3. Read the Data field.
4. Read the Free Running Timer (optional – releases the internal lock).

Upon reading the Control and Status word, if the BUSY bit is set in the Code field, then the CPU should defer the access to the MB until this bit is negated. Reading the Free Running Timer is not mandatory. If not executed the MB remains locked, unless the CPU reads the Control and Status word of another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is the one on the Control and Status word to assure data coherency (see Section 18.4.6, Data coherence).

The CPU should synchronize to frame reception by the status flag bit for the specific MB in one of the interrupt flag registers (IFRL, IFRH) and not by the Code field of that MB. Polling the Code field does not work because once a frame has been received and the CPU services the MB (by reading the Control and Status word followed by unlocking the MB), the Code field will not return to EMPTY. It will remain FULL, as explained in Table 18-5. If the CPU tries to work around this behavior by writing to the Control and Status word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost.

NOTE

Never do polling by reading directly the Control and Status word of the MBs. Instead, read the interrupt flag registers (IFRL, IFRH).

Note that the received ID field is always stored in the matching MB; thus the contents of the ID field in an MB may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists an Rx matching MB, provided the SRX_DIS bit in the MCR is not asserted. If SRX_DIS is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the FIFO, the CPU must enable and configure the FIFO during Freeze mode (see Section 18.4.7, Rx FIFO). Upon receiving the frames-available interrupt from FIFO, the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits).
2. Read the ID field (optional – needed only if a mask was used).
3. Read the Data field.
4. Clear the frames available interrupt (mandatory – release the buffer and allow the CPU to read the next FIFO entry).

18.4.5 Matching process

The matching process is an algorithm executed by the MBM that scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the

8-entry ID table from FIFO is scanned first and then, if a match is not found within the FIFO table, the other MBs are scanned. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

When the frame is received, it is temporarily stored in a hidden auxiliary MB called Serial Message Buffer (SMB). The matching process takes place during the CRC field of the received frame. If a matching ID is found in the FIFO table or in one of the regular MBs, the contents of the SMB will be transferred to the FIFO or to the matched MB during the 6th bit of the End-of-Frame field of the CAN protocol. This operation is called move-in. If any protocol error (CRC, ACK, etc.) is detected, than the move-in operation does not happen.

For the regular mailbox MBs, an MB is said to be free to receive a new frame if the following conditions are satisfied:

- The MB is not locked (see Section 18.4.6.3, Message Buffer lock mechanism).
- The Code field is either EMPTY or else it is FULL or OVERRUN but the CPU has already serviced the MB (has read the Control and Status word and then unlocked the MB).

If the first MB with a matching ID is not free to receive the new frame, then the matching algorithm keeps looking for another free MB until it finds one. If it cannot find one that is free, then it will overwrite the last matching MB (unless it is locked) and set the Code field to OVERRUN (refer to Table 18-5 and Table 18-6). If the last matching MB is locked, then the new message remains in the SMB, waiting for the MB to be unlocked (see Section 18.4.6.3, Message Buffer lock mechanism).

Suppose, for example, that the FIFO is disabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not free to receive, so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are free to receive, so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the Code field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

The matching algorithm described above can be changed to be the same one used in previous versions of the FlexCAN module. When the BCC bit in MCR is negated, the matching algorithm stops at the first MB with a matching ID that it founds, whether this MB is free or not. As a result, the message queueing feature does not work if the BCC bit is negated.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB (see Section 18.3.4.13, Rx Individual Mask Registers (RXIMR0–RXIMR63)). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a don’t-care bit. Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed if the BCC bit is asserted and while the module is in Freeze mode.

FlexCAN also supports an alternate masking scheme with only three mask registers (RGXMASK, RX14MASK, and RX15MASK) for backwards compatibility. This alternate masking scheme is enabled when the BCC bit in the MCR Register is negated.

18.4.6 Data coherence

In order to maintain data coherency and proper FlexCAN operation, the CPU must obey the rules described in Transmit process and Section 18.4.4, Receive process. Any form of CPU accessing an MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

18.4.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead. In order to maintain backwards compatibility, the abort mechanism must be explicitly enabled by asserting the AEN bit in the MCR.

In order to abort a transmission, the CPU must write a specific abort code (1001) to the Code field of the Control and Status word. When the abort mechanism is enabled, the active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to an MB that is currently being transmitted, or to an MB that was already loaded into the SMB for transmission, the write operation is blocked and the MB is not deactivated, but the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration.
- There is an error during the transmission.
- The module is put into Freeze mode.

If none of these conditions is reached, then the MB is transmitted correctly, the interrupt flag is set in the interrupt flag registers (IFRL, IFRH), and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted—therefore the abort code is written into the Code field, the interrupt flag is set in the interrupt flag registers (IFRL, IFRH), and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked—therefore the MB is updated and no interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was deactivated. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is deactivated and not aborted, because the transmission did not start yet. One MB is only aborted when the abort request is captured and kept pending until one of the previous conditions is satisfied.

The abort procedure can be summarized as follows:

- CPU writes 1001 into the code field of the Control and Status word.
- CPU reads the CODE field and compares it to the value that was written.
- If the CODE field that was read is different from the value that was written, the CPU must read the corresponding interrupt flag to check if the frame was transmitted or is currently being transmitted.

If the corresponding interrupt flag bit is set, the frame was transmitted. If the corresponding interrupt flag bit is reset, the CPU must wait for it to be set, and then the CPU must read the CODE field to check if the MB was aborted (CODE = 1001) or was transmitted (CODE = 1000).

18.4.6.2 Message Buffer deactivation

Deactivation is a mechanism provided to maintain data coherence when the CPU writes to the Control and Status word of active MBs out of Freeze mode. Any CPU write access to the Control and Status word of an MB causes that MB to be excluded from the transmit or receive processes during the current matching or arbitration round. The deactivation is temporary, affecting only the current match/arbitration round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the CPU updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent—therefore deactivation of that MB is done.

Even with the coherence mechanism described above, writing to the Control and Status word of active MBs when not in Freeze mode may produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If MBs are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If an Rx MB with a matching ID is deactivated during the matching process after it was scanned, then this MB is marked as invalid to receive the frame, and FlexCAN will keep looking for another matching MB within the ones it has not scanned yet. If it cannot find one, then the message will be lost. Suppose, for example, that two MBs have a matching ID to a received frame, and the user deactivated the first matching MB after FlexCAN has scanned the second. The received frame will be lost even if the second matching MB was free to receive.
- If a Tx MB containing the lowest ID is deactivated after FlexCAN has scanned it, then FlexCAN will look for another winner within the MBs that it has not scanned yet. Therefore, it may transmit an MB with ID that may not be the lowest, at the time because a lower ID might be present in one of the MBs that it had already scanned before the deactivation.
- There is a point in time until which the deactivation of a Tx MB causes it not to be transmitted (end of move-out). After this point, it is transmitted but no interrupt is issued and the Code field is not updated. In order to avoid this situation, the abort procedures described in Section 18.4.6.1, Transmission abort mechanism should be used.

18.4.6.3 Message Buffer lock mechanism

Besides MB deactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an “active not empty” Rx MB, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

NOTE

The locking mechanism only applies to Rx MBs which have a code different than INACTIVE (0000) or EMPTY[1] (0100). Also, Tx MBs cannot be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no free to receive MBs, so it decides to override MB number 5. However, this MB is locked, so the new message cannot be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the Code field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the Code field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

NOTE

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Deactivation takes precedence over locking. If the CPU deactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB.

18.4.7 Rx FIFO

The receive-only FIFO is enabled by asserting the FEN bit in the MCR. The reset value of this bit is zero to maintain software backwards compatibility with previous versions of the module that did not have the FIFO feature. When the FIFO is enabled, the memory region normally occupied by the first eight MBs (0x80–0xFF) is now reserved for use of the FIFO engine (see Section 18.3.3, Rx FIFO Structure). Management of read and write pointers is done internally by the FIFO engine. The CPU can read the received frames sequentially, in the order they were received, by repeatedly accessing a Message Buffer structure at the beginning of the memory.

The FIFO can store up to six frames pending service by the CPU. An interrupt is sent to the CPU when new frames are available in the FIFO. Upon receiving the interrupt, the CPU must read the frame (accessing an MB in the 0x80 address) and then clear the interrupt. The act of clearing the interrupt triggers the FIFO engine to replace the MB in 0x80 with the next frame in the queue, and then issue another interrupt to the CPU. If the FIFO is full and more frames continue to be received, an OVERFLOW interrupt is issued to the CPU and subsequent frames are not accepted until the CPU creates space in the

1. In previous FlexCAN versions, reading the Control and Status word locked the MB even if it was EMPTY. This behavior will be honored when the BCC bit is negated.

FIFO by reading one or more frames. A warning interrupt is also generated when four frames are accumulated in the FIFO.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of eight 32-bit registers that can be configured to one of the following formats (see also Section 18.3.3, Rx FIFO Structure):

- Format A: 8 extended or standard IDs (including IDE and RTR)
- Format B: 16 standard IDs or 16 extended 14-bit ID slices (including IDE and RTR)
- Format C: 32 standard or extended 8-bit ID slices

NOTE

A chosen format is applied to all eight registers of the filter table. It is not possible to mix formats within the table.

The eight elements of the filter table are individually affected by the first eight Individual Mask Registers (RXIMR0–RXIMR7), allowing very powerful filtering criteria to be defined. The rest of the RXIMR, starting from RXIM8, continue to affect the regular MBs, starting from MB8. If the BCC bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO filter table is affected by the legacy mask registers as follows: element 6 is affected by RX14MASK, element 7 is affected by RX15MASK and the other elements (0 to 5) are affected by RXGMASK.

18.4.8 CAN protocol related features

18.4.8.1 Remote frames

A remote frame is a special kind of frame. The user can program an MB to be a Request Remote frame type by writing the MB as Transmit with the RTR bit set to 1. After the Remote Request frame is transmitted successfully, the MB becomes a Receive Message Buffer, with the same ID as before.

When a Remote Request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with the Code field 1010. If there is a matching ID, then this MB frame will be transmitted. Note that if the matching MB has the RTR bit set, then FlexCAN will transmit a Remote frame as a response.

A received Remote Request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match.

In the case that a Remote Request frame was received and matched an MB, this message buffer immediately enters the internal arbitration process, but is considered as a normal Tx MB, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

If the Rx FIFO is enabled (bit FEN set in MCR), FlexCAN will not generate an automatic response for Remote Request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is

possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).

18.4.8.2 Overload frames

FlexCAN transmits overload frames due to detection of the following conditions on the CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End-of-Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame delimiter or Overload Frame delimiter

18.4.8.3 Time stamp

The value of the free-running timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of “move-in” in the TIME STAMP field, providing network behavior with respect to time.

Note that the free-running timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in Section 18.3.4.2, Control Register (CTRL).

18.4.8.4 Protocol timing

Figure 18-16 shows the structure of the clock generation circuitry that feeds the CAN Protocol Interface (CPI) submodule. The clock source bit (CLK_SRC) in the CTRL Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock (generally from a PLL). In order to guarantee reliable operation, the clock source should be selected while the module is in Disable mode (bit MDIS set in the Module Configuration Register).

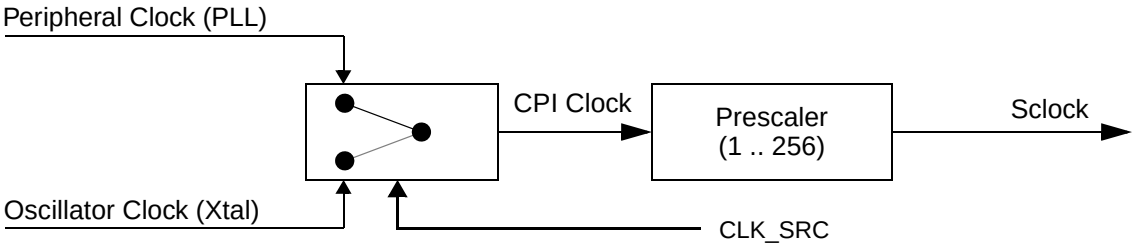


Figure 18-16. CAN engine clocking scheme

The crystal oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks.

NOTE

This clock selection feature may not be available in all MCUs. A particular MCU may not have a PLL, in which case it would have only the oscillator clock, or it may use only the PLL clock feeding the FlexCAN module. In these cases, the CLK_SRC bit in the CTRL Register has no effect on the module operation.

The FlexCAN module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW. See Section 18.3.4.2, Control Register (CTRL).

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the time quantum used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler Value})}$$

A bit time is subdivided into three segments[1] (reference Figure 18-17 and Table 18-20):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 4 to 16 time quanta.
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long.

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

1. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3. See also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

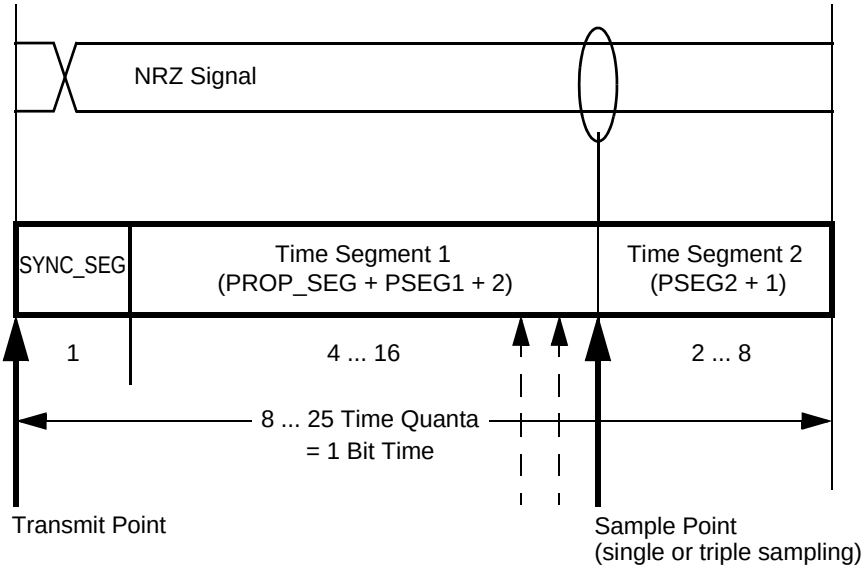


Figure 18-17. Segments within the bit time

Table 18-20. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 18-21 gives an overview of the CAN compliant segment settings and the related parameter values.

Table 18-21. CAN standard compliant bit time segment settings

Time Segment 1	Time Segment 2	Resynchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

NOTE

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

18.4.8.5 Arbitration and matching timing

During normal transmission or reception of frames, the arbitration, matching, move-in, and move-out processes are executed during certain time windows inside the CAN frame, as shown in Figure 18-18.

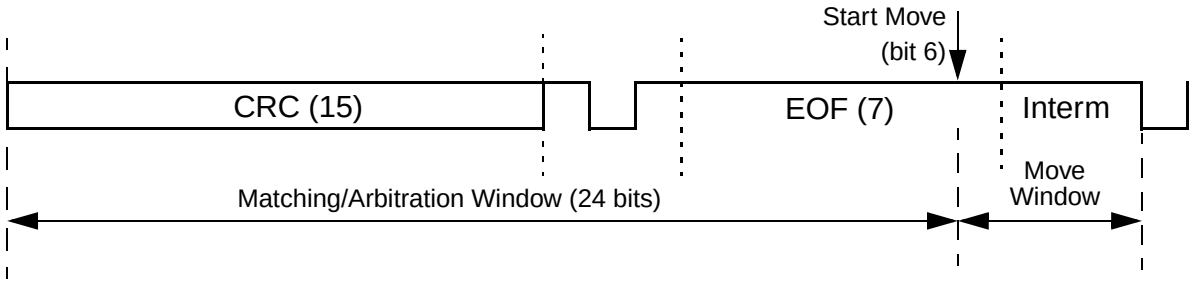


Figure 18-18. Arbitration, match, and move time windows

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 18-21.
- The peripheral clock frequency cannot be smaller than the oscillator clock frequency; that is, the PLL cannot be programmed to divide down the oscillator clock.
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in Table 18-22.

Table 18-22. Minimum ratio between peripheral clock frequency and CAN bit rate

Number of message buffers	Minimum ratio
16	8
32	8
64	16

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be eight, so the oscillator clock frequency should be at least eight times the CAN bit rate. The minimum frequency ratio specified in Table 18-22 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have eight time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least two. For a prescaler factor

equal to one and CAN bit timing with eight time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least two.

18.4.9 Modes of operation: details

18.4.9.1 Freeze mode

This mode is entered by asserting the HALT bit in the MCR Register or when the MCU is stopped by a debugger. In both cases it is also necessary that the FRZ bit be asserted in the MCR Register and the module is not in any of the low-power modes (Disable or Stop). When Freeze mode is requested during transmission or reception, the FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off, or Idle state
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action—otherwise the FlexCAN may operate in an unpredictable way. In Freeze mode, all memory-mapped registers are accessible.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR Register.
- The MCU is started by the debugger and/or the HALT bit is negated.

Once out of Freeze mode, FlexCAN tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

18.4.9.2 Module Disable mode

This low-power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze mode, it shuts down the clocks to the CPI and MBM submodules, sets the LPM_ACK bit, and negates the FRZ_ACK bit. If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks to see if it is recessive
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the CPI and MBM submodules
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory-mapped registers, except the Free Running Timer, the Error Counter Register, and the Message Buffers, which cannot be accessed

when the module is in Disable mode. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate the LPM_ACK bit.

18.4.10 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to ORed interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning, and Wake Up). The number of actual sources depends on the configured number of Message Buffers.

On this device, the individual MB interrupts are grouped as follows:

- Groups of four interrupts (up to MB 16)
- MB16_31
- MB32_63

These are then used as the interrupt source.

Each one of the message buffers can be an interrupt source, if its corresponding mask bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the IFRL or IFRH registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the CPU writes a 1 to it (unless another interrupt is generated at the same time).

NOTE

It must be guaranteed that the CPU only clears the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (bit FEN on MCR set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFRL becomes the FIFO Overflow flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the Frames Available in FIFO flag. and bits 4–0 are unused. See Section 18.3.4.12, Interrupt Flag Register Low (IFRL), for more information.

A combined interrupt for all MBs is also generated by an OR of all the interrupt sources from MBs. This interrupt gets generated when any of the MBs generates an interrupt. In this case the CPU must read the IFRL and IFRH registers to determine which MB caused the interrupt.

The other five interrupt sources (Bus Off, Error, Tx Warning, Rx Warning, and Wake Up) generate interrupts like the MB ones, and can be read from the Error and Status Register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the Control Register, and the Wake-Up interrupt mask bit is located in the MCR.

18.4.11 Bus interface

CPU access to FlexCAN registers is subject to the following rules:

- Read and write access to supervisor registers in User mode results in access error.

- Read and write access to unimplemented or reserved address space also results in access error. Any access to unimplemented MB or Rx Individual Mask Register locations results in access error. Any access to the Rx Individual Mask Register space when the BCC bit in MCR is negated results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that the Rx Individual Mask Registers can only be accessed in Freeze mode, and this is still true for unused space within this memory. Note also that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 64 MBs and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The MB memory starts at 0x0060, but the space from 0x0060 to 0x007F is reserved (for SMB usage), and the space from 0x0080 to 0x008F is used by the one MB. This leaves us with the available space from 0x0090 to 0x047F. The available memory in the Mask Registers space would be from 0x0884 to 0x097F.

NOTE

Unused MB space must not be used as general purpose RAM while FlexCAN is transmitting and receiving CAN frames.

18.5 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

18.5.1 FlexCAN initialization sequence

The FlexCAN module may be reset in three ways:

- MCU level hard reset, which resets all memory-mapped registers asynchronously
- MCU level soft reset, which resets some of the memory-mapped registers synchronously (refer to Table 18-2 to see what registers are affected by soft reset)
- SOFT_RST bit in MCR, which has the same effect as the MCU level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset cannot be applied while clocks are shut down in any of the low-power modes. The low-power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLK_SRC bit) should be selected while the module is in Disable mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled, and the FRZ_ACK and NOT_RDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization, the FlexCAN must be put into Freeze mode (see Section 18.4.9.1, Freeze mode). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register.
  - Enable the individual filtering per MB and reception queue features by setting the BCC bit.
  - Enable the warning interrupts by setting the WRN_EN bit.
  - If required, disable frame self-reception by setting the SRX_DIS bit.
  - Enable the FIFO by setting the FEN bit.
  - Enable the abort mechanism by setting the AEN bit.
  - Enable the local priority feature by setting the LPRIO_EN bit.
- Initialize the Control Register.
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, and RJW.
  - Determine the bit rate by programming the PRESDIV field.
  - Determine the internal arbitration mode (LBUF bit).
- Initialize the Message Buffers.
  - The Control and Status word of all Message Buffers must be initialized.
  - If FIFO was enabled, the 8-entry ID table must be initialized.
  - Other entries in each Message Buffer should be initialized as required.
- Initialize the Rx Individual Mask Registers.
- Set required interrupt mask bits in the mask registers (for all MB interrupts), in CTRL Register (for Bus Off and Error interrupts) and in MCR Register for Wake-Up interrupt.
- Negate the HALT bit in MCR.

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

18.5.2 FlexCAN Addressing and RAM size configurations

There are three RAM configurations that can be implemented within the FlexCAN module. The possible configurations are:

- For 16 MBs: 288 bytes for MB memory and 64 bytes for Individual Mask Registers
- For 32 MBs: 544 bytes for MB memory and 128 bytes for Individual Mask Registers
- For 64 MBs: 1056 bytes for MB memory and 256 bytes for Individual Mask Registers

In each configuration the user can program the maximum number of MBs that will take part in the matching and arbitration processes using the MAXMB field in the MCR Register. For 16 MB configuration, MAXMB can be any number between 0–15. For 32 MB configuration, MAXMB can be any number between 0–31. For 64 MB configuration, MAXMB can be any number between 0–63.

Chapter 19
IEEE 1149.1 Test Access Port Controller (JTAGC)

19.1 Introduction

The JTAG port of the device consists of three inputs and one output. These pins include test data input (TDI), test data output (TDO), test mode select (TMS), and test clock input (TCK). TDI, TDO, TMS, and TCK are compliant with the IEEE 1149.1-2001 standard and are shared with the NDI through the test access port (TAP) interface.

IEEE 1149.7 (cJTAG) is not supported on this device.

19.2 Block diagram

Figure 19-1 is a block diagram of the JTAG Controller (JTAGC).

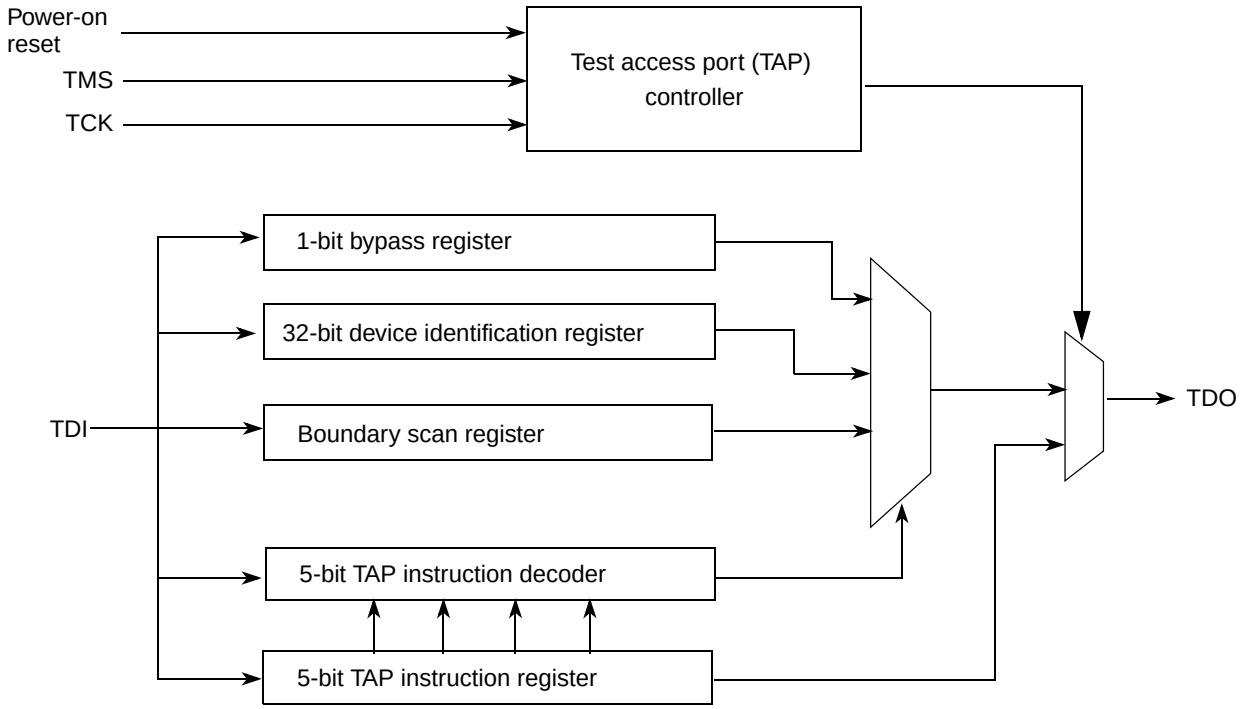


Figure 19-1. JTAG controller block diagram

19.3 Overview

The JTAGC provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. In addition, instructions can be executed that allow the Test Access Port (TAP) to be shared with other modules on the MCU. All data input to and output from the JTAGC is communicated in serial format.

19.4 Features

The JTAGC is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
- 4 pins (see Section 19.6, External signal description)
  - TDI
  - TMS
  - TCK
  - TDO
- A 5-bit instruction register that supports several IEEE 1149.1-2001 defined instructions, as well as several public and private MCU specific instructions
- Three test data registers, a bypass register, and a device identification register
- A TAP controller state machine that controls the operation of the data registers, instruction register, and associated circuitry

19.5 Modes of operation

The JTAGC uses a power-on reset indication as its primary reset signal. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

19.5.1 Reset

The JTAGC is placed in reset when the TAP controller state machine is in the Test-Logic-Reset state. The Test-Logic-Reset state is entered upon the assertion of the power-on reset signal, or through TAP controller state machine transitions controlled by TMS. Asserting power-on reset results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered.
- The instruction register is loaded with the IDCODE instruction.

In addition, execution of certain instructions can result in assertion of the internal system reset. These instructions include EXTEST.

19.5.2 IEEE 1149.1-2001 defined test modes

The JTAGC supports several IEEE 1149.1-2001 defined test modes. The test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, SAMPLE, and SAMPLE/PRELOAD. Each instruction defines the set of data registers that can operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is external to JTAGC but can be accessed by JTAGC TAP through the EXTEST, SAMPLE, and SAMPLE/PRELOAD instructions. The functionality of each test mode is explained in more detail in Section 19.8.4, JTAGC instructions.

19.5.2.1 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

19.5.2.2 TAP sharing mode

There are three selectable auxiliary TAP controllers that share the TAP with the JTAGC. Selectable TAP controllers include the Nexus port controller (NPC) and PLATFORM. The instructions required to grant ownership of the TAP to the auxiliary TAP controllers are ACCESS_AUX_TAP_NPC, ACCESS_AUX_TAP_ONCE, and ACCESS_AUX_TAP_TCU. Instruction opcodes for each instruction are shown in Table 19-3.

When the access instruction for an auxiliary TAP is loaded, control of the JTAG pins is transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the Update-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

For more information on the TAP controllers refer to Chapter 26, Nexus Development Interface (NDI).

19.6 External signal description

The JTAGC consists of four signals that connect to off-chip development tools and allow access to test support functions. The JTAGC signals are outlined in Table 19-1.

Table 19-1. JTAG signal properties

Name	I/O	Function	Reset State
TCK	I	Test clock	Pullup
TDI	I	Test data in	Pullup
TDO	O	Test data out	High Z
TMS	I	Test mode select	Pullup

All four JTAG pins (TCK/TMS/TDI/TDO) are shared with GPIO pins, so that the software may configure these pins as input/output by programming the appropriate registers.

To ensure the proper working of JTAG, these registers have a reset value such that these pins behave as JTAG pins when the POR is lifted:

- TDI: input/pullup
- TCK: input/pullup
- TMS: input/pullup
- TDO: high-impedance/pull disabled

On entry to Standby mode the TDO pin goes to the high-Z/pull-disabled state. Some external debugger connections may expect the TDO to be in a known state during standby, so an external pullup or pulldown may be required for correct operation when debugging Standby.

NOTE

The JTAG Clock (TCK) typically operates at a frequency well below the system clock frequency, as specified in the *MPC5606S Microcontroller Data Sheet*. In some cases, however, the system clock frequency may be lowered significantly from the normal operating range. If the system clock frequency is reduced below the frequency of TCK, it will no longer be possible to communicate with the Nexus Port Controller Port Configuration Register (NPC_PCR).

19.7 Memory map and register description

This section provides a detailed description of the JTAGC registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

19.7.1 Instruction register

The JTAGC uses a 5-bit instruction register as shown in Figure 19-2. The instruction register allows instructions to be loaded into the module to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 0b10101, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

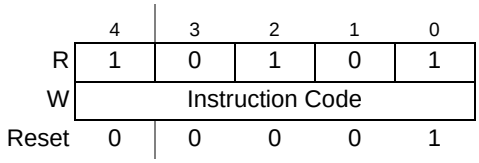


Figure 19-2. 5-Bit Instruction Register

19.7.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS or the reserve instructions are active. After entry into the capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

19.7.3 Device Identification register

The device identification register, shown in Figure 19-3, allows the part revision number, design center, part identification number, and manufacturer identity code to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

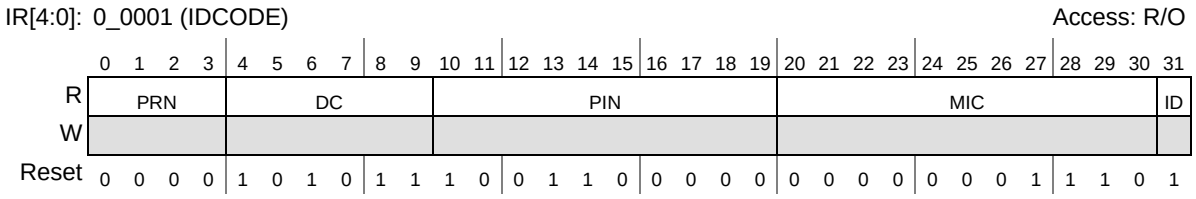


Figure 19-3. Device Identification Register

Table 19-2. Device Identification Register field descriptions

Field	Description
0–3 PRN	Part revision number. Contains the revision number of the device. This field changes with each revision of the device or module.
4–9 DC	Design center.
10–19 PIN	Part identification number. Contains the part number of the device.
20–30 MIC	Manufacturer identity code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale, 0xE
31 ID	IDCODE register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

19.7.4 Boundary Scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Section 19.8.5, Boundary scan. The size of the boundary scan register is 464 bits.

19.8 Functional description

19.8.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

19.8.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. For more detail on TAP sharing via JTAGC instructions, refer to Section 19.8.4.2, ACCESS_AUX_TAP_x instructions.

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in Figure 19-4. This applies for the instruction register, test data registers, and the bypass register.

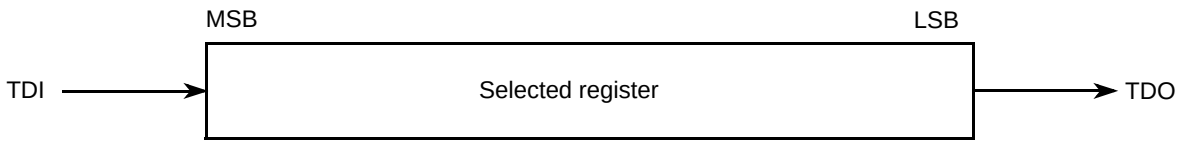
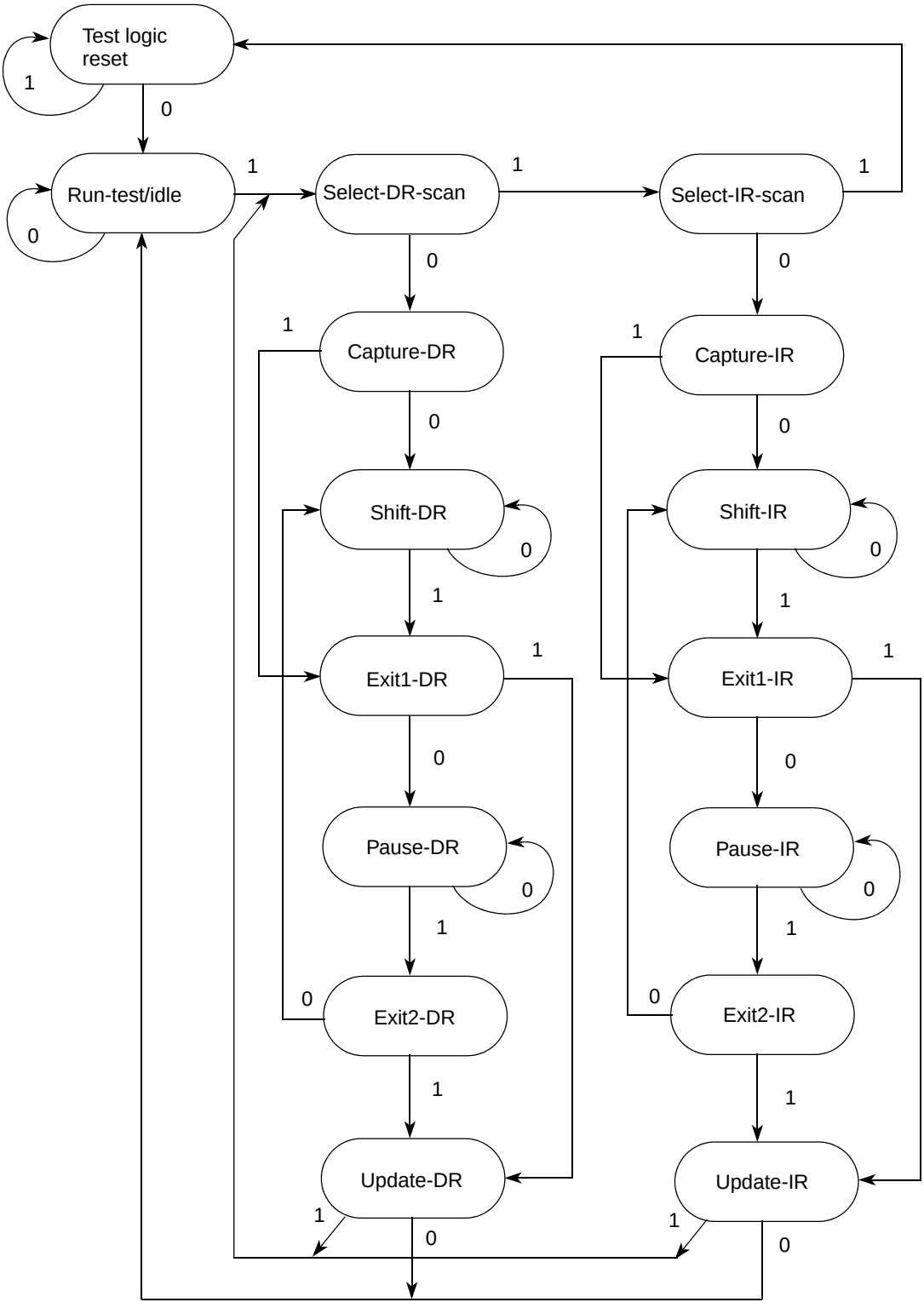


Figure 19-4. Shifting Data Through a Register

19.8.3 TAP Controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. Figure 19-5 shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal.

As Figure 19-5 shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 19-5. IEEE 1149.1-2001 TAP Controller finite state machine

19.8.3.1 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is done by loading the instruction register with any of the JTAGC instructions while the JTAGC is enabled. Instructions are shifted in via the select-IR-scan path and loaded in the Update-IR state. At this point, all data register access is performed via the select-DR-scan path.

The select-DR-scan path is used to read or write the register data by shifting in the data (LSB first) during the shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting can be terminated after fetching the required number of bits.

19.8.4 JTAGC instructions

This section gives an overview of each instruction. Refer to the IEEE 1149.1-2001 standard for more details.

The JTAGC implements the IEEE 1149.1-2001 defined instructions listed in Table 19-3.

Table 19-3. JTAG instructions

Instruction	Code[4:0]	Instruction summary
IDCODE	00001	Selects device identification register for shift
SAMPLE/PRELOAD	00010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	00011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	00100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
ACCESS_AUX_TAP_TCU	11011	Grants the TCU ownership of the TAP
ACCESS_AUX_TAP_ONCE	10001	Grants the PLATFROM ownership of the TAP
ACCESS_AUX_TAP_NPC	10000	Grants the Nexus port controller (NPC) ownership of the TAP
BYPASS	11111	Selects bypass register for data operations
Factory Debug Reserved[1]	00101 00110 01010	Intended for factory debug only
Reserved[2]	All Other Codes	Decoded to select bypass register

1 Intended for factory debug, and not customer use

2 Freescale reserves the right to change the decoding of reserved instruction codes

Table 19-4 shows the implementation for silicon cut1. By mistake, the Access to Nexus Port Controller is not using the standard PowerPC instruction.

For silicon cut2, the instruction coding will be changed to be 100% compatible with existing PowerPC.

Table 19-4. JTAG instructions for silicon cut1

Instruction	Code[4:0]	Instruction summary
IDCODE	00001	Selects device identification register for shift
SAMPLE/PRELOAD	00010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	00011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	00100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
ACCESS_AUX_TAP_TCU	10000	Grants the TCU ownership of the TAP
ACCESS_AUX_TAP_ONCE	10001	Grants the PLATFORM ownership of the TAP
ACCESS_AUX_TAP_NPC	10010	Grants the Nexus port controller (NPC) ownership of the TAP
BYPASS	11111	Selects bypass register for data operations
Factory Debug Reserved[1]	00101 00110 01010	Intended for factory debug only
Reserved[2]	All Other Codes	Decoded to select bypass register

1 Intended for factory debug, and not customer use

2 Freescale reserves the right to change the decoding of reserved instruction codes

19.8.4.1 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

19.8.4.2 ACCESS_AUX_TAP_*x* instructions

The ACCESS_AUX_TAP_*x* instructions allow the Nexus modules on the MCU to take control of the TAP. When this instruction is loaded, control of the TAP pins is transferred to the selected auxiliary TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the Update-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

19.8.4.3 EXTEST — External Test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the

internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

19.8.4.4 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC is reset.

19.8.4.5 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

19.8.4.6 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE part of the instruction samples the system data and control signals on the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising-edge of TCK in the capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD part of the instruction initializes the boundary scan register cells before selecting the EXTEST instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST instruction. System operation is not affected.

19.8.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

19.9 e200z0 OnCE controller

The e200z0 core OnCE controller supports a complete set of Nexus 1 debug features, as well as providing access to the Nexus2+ configuration registers. A complete discussion of the e200z0 OnCE debug features is available in the *e200z0 Reference Manual*.

19.9.1 e200z0 OnCE controller block diagram

Figure 19-6 is a block diagram of the e200z0 OnCE block.

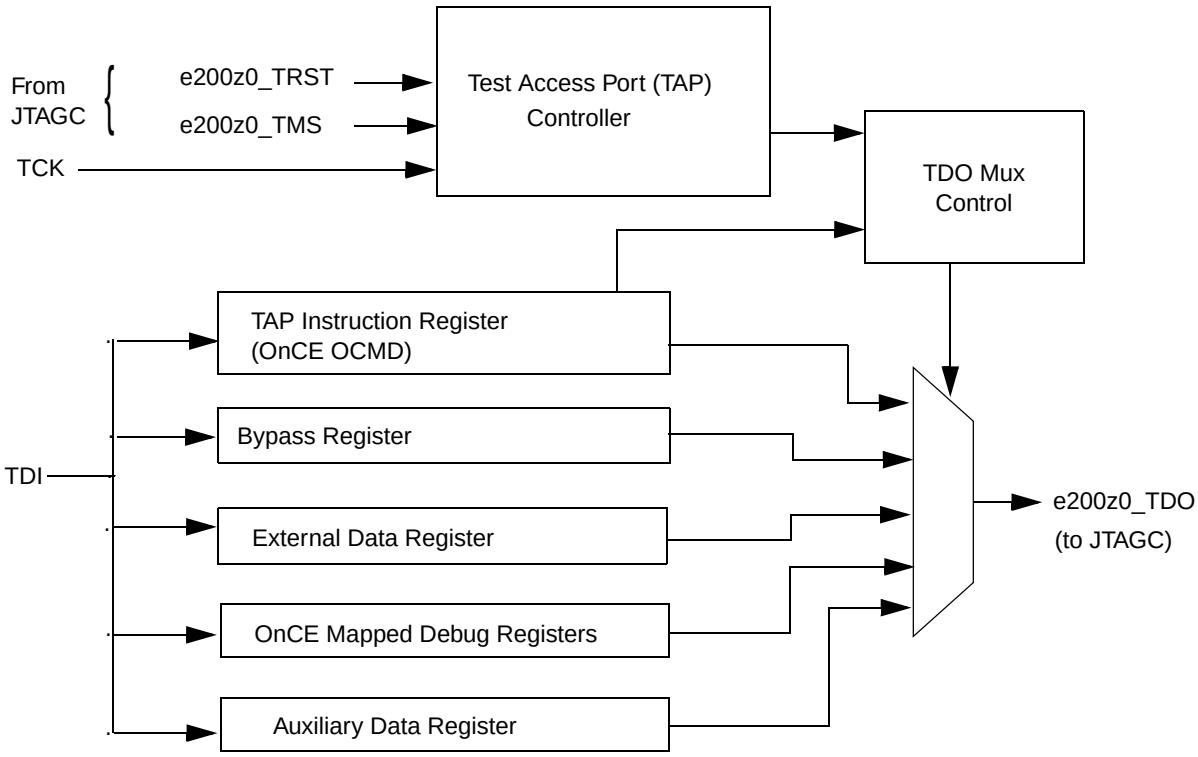


Figure 19-6. e200z0 OnCE block diagram

19.9.2 e200z0 OnCE controller functional description

The functional description for the e200z0 OnCE controller is the same as for the JTAGC, with the differences described below.

19.9.2.1 Enabling the TAP controller

To access the e200z0 OnCE controller, the proper JTAGC instruction needs to be loaded in the JTAGC instruction register, as discussed in Section 19.5.2.2, TAP sharing mode.

19.9.3 e200z0 OnCE controller register description

Most e200z0 OnCE debug registers are fully documented in the *e200z0 Reference Manual*.

19.9.3.1 OnCE Command Register (OCMD)

The OnCE command register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the e200z0 OnCE Decoder. The OCMD is shown in Table 19-5. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

	0	1	2	3	4	5	6	7	8	9
R W	R/W	GO	EX	RS[0:6]						
Reset:	0	0	0	0	0	1	1	0	1	1

Table 19-5. OnCE Command Register (OCMD)

Table 19-6. e200z0 OnCE register addressing

RS[0:6]	Register Selected
000 0000–000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1)

Table 19-6. e200z0 OnCE register addressing (continued)

RS[0:6]	Register Selected
010 0111	Data Value Compare 2 (DVC2)
010 1000–010 1111	Reserved
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100–101 1111	Reserved (do not access)
110 1111	Reserved (do not access)
111 0000–111 1001	General Purpose Register Selects [0:9]
111 1010–111 1011	Reserved
111 1100	Nexus2+ Access
111 1101	LSRL Select (factory test use only)
111 1110	Enable_OnCE
111 1111	Bypass

19.10 Initialization/application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC module and enable access to registers, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS.
2. Load the appropriate instruction for the test or action to be performed.

Chapter 20
Inter-Integrated Circuit Bus Controller Module (I^2C)

20.1 Introduction

20.1.1 Overview

The Inter-Integrated Circuit (I^2C™ or IIC) bus is a two wire bidirectional serial bus that provides a simple and efficient method of data exchange between devices. It minimizes the number of external connections to devices and does not require an external address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface is designed to operate up to 100 kbit/s with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of module clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

20.1.2 Features

The I^2C module has the following key features:

- Compatible with I^2C Bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection

Features currently not supported:

- No support for general call address
- Not compliant to ten-bit addressing

20.1.3 Block diagram

The block diagram of the I^2C module is shown in Figure 20-1.

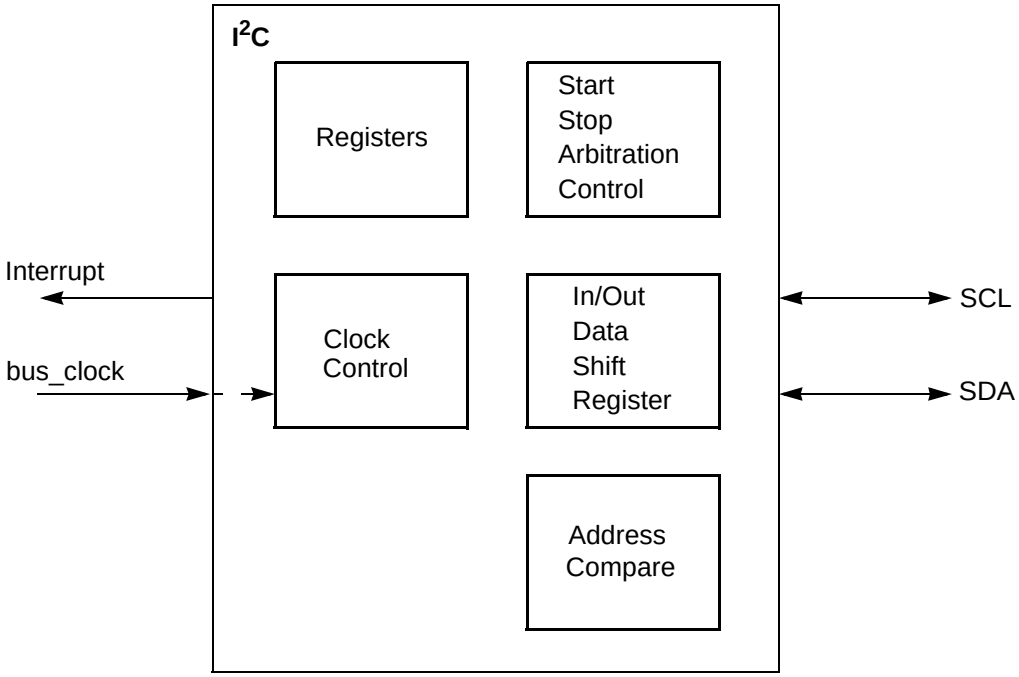


Figure 20-1. I^2C block diagram

20.2 Modes of operation

The I^2C module has the following modes of operation:

- Run mode: This is the basic mode of operation.
- Stop mode: This is the lowest power saving mode and allows the system to turn of all the clocks to the I^2C module. This state can only be entered when there are no active transfers on the bus.

20.3 External signal description

20.3.1 Overview

The Inter-Integrated Circuit (I^2C) module has 2 external pins.

20.3.2 Detailed signal descriptions

20.3.2.1 SCL

This is the bidirectional Serial Clock Line (SCL) of the module, compatible with the I^2C-Bus specification.

20.3.2.2 SDA

This is the bidirectional Serial Data line (SDA) of the module, compatible with the I^2C-Bus specification.

20.4 Memory map and register description

20.4.1 Overview

This section provides a detailed description of all memory-mapped registers in the I^2C module.

20.4.2 Module memory map

The memory map for the I^2C module is given below in Table 20-1. The total address for each register is the sum of the base address for the I^2C module and the address offset for each register.

Table 20-1. Module memory map

Address	Register	Size (bits)	Access	Mode[1]	Location
Base + 0x00	I^2C Bus Address Register (IBAD)	8	R/W	A	on page 732
Base + 0x01	I^2C Bus Frequency Divider Register (IBFD)	8	R/W	A	on page 732
Base + 0x02	I^2C Bus Control Register (IBCR)	8	R/W	A	on page 738
Base + 0x03	I^2C Bus Status Register (IBSR)	8	R/W	A	on page 739
Base + 0x04	I^2C Bus Data I/O Register (IBDR)	8	R/W	A	on page 740
Base + 0x05	I^2C Bus Interrupt Configuration Register (IBIC)	8	R/W	A	on page 741
Base + 0x06	Unused	8	R	A	—
Base + 0x07	Unused	8	R	A	—
Base + 0x08 – Base + 0x3FFF	Reserved		See Note [2]		—

1 **U** = User mode, **S** = Supervisor mode, **A** = All (No restrictions)

2 If enabled at the SoC level, reads or writes to these registers will cause bus aborts. Refer to the System Services Module documentation for more details.

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the IBDF register for the frequency divider is accessible by a 16-bit READ/WRITE to address Base + 0x000, but performing a 16-bit access to Base + 0x001 is illegal.

20.4.3 Register description

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

20.4.3.1 I^2C Bus Address Register

This register contains the address the I^2C Bus will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

Offset 0x00000 Access: User read/write

	0	1	2	3	4	5	6	7
R	ADR							0
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-2. I^2C Bus Address Register (IBAD)

Table 20-2. IBAD field descriptions

Field	Description
ADR	Slave Address. Specific slave address to be used by the I^2C Bus module. **Note:** The default mode of I^2C Bus is Slave mode for an address match on the bus.

20.4.3.2 I^2C Bus Frequency Divider Register

Offset 0x0001 Access: User read/write

	0	1	2	3	4	5	6	7
R	IBC							
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-3. I^2C Bus Frequency Divider Register (IBFD)

Table 20-3. IBFD field descriptions

Field	Description
IBC	I-Bus Clock Rate. This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider. The IBC bits are decoded to give the Tap and Prescale values as follows: 0–1 select the prescaled shift register (see Table 20-4) 2–4 select the prescaler divider (see Table 20-5) 5–7 select the shift register tap point (see Table 20-6)

Table 20-4. I-Bus Multiplier Factor

IBC[0:1]	MUL
00	01
01	02
10	04
11	RESERVED

Table 20-5. I-Bus Prescaler Divider Values

IBC[2:4]	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 20-6. I-Bus Tap and Prescale Values

IBC[5:7]	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 20-5. All subsequent tap points are separated by $2^{IBC[2:4]}$ as shown in the tap2tap column in Table 20-5. The SCL Tap is used to generate the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to the change of state of SDA; that is, the SDA hold time.

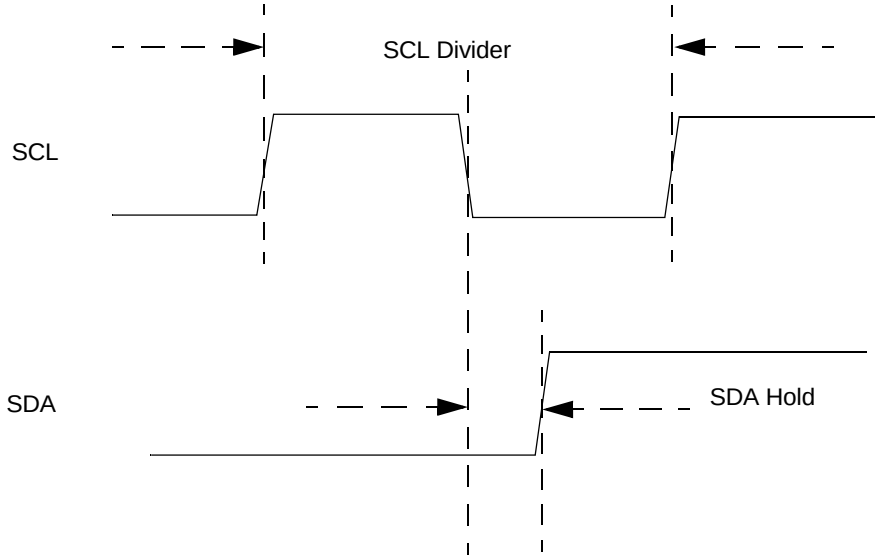


Figure 20-4. SDA Hold Time

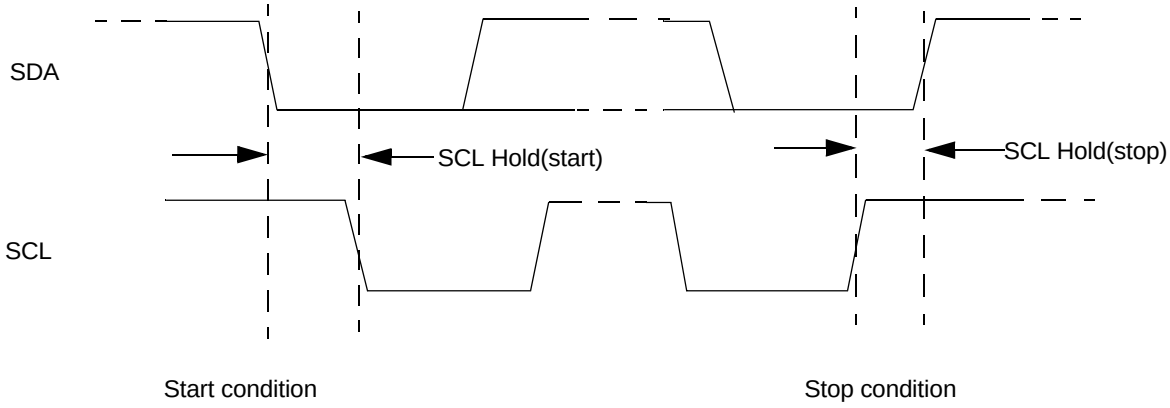


Figure 20-5. SCL Divider and SDA Hold

The equation used to generate the divider values from the IBFD bits is:

$$\text{SCL Divider} = \text{MUL} \times \{2 \times (\text{scl2tap} + [(\text{SCL\_Tap} - 1) \times \text{tap2tap}] + 2)\} \qquad \textit{Eqn. 20-1}$$

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 20-7. The equation used to generate the SDA Hold value from the IBFD bits is:

$$\text{SDA Hold} = \text{MUL} \times \{\text{scl2tap} + [(\text{SDA\_Tap} - 1) \times \text{tap2tap}] + 3\} \qquad \textit{Eqn. 20-2}$$

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

$$\text{SCL Hold(start)} = \text{MUL} \times [\text{scl2start} + (\text{SCL\_Tap} - 1) \times \text{tap2tap}] \qquad \textit{Eqn. 20-3}$$

$$\text{SCL Hold(stop)} = \text{MUL} \times [\text{scl2stop} + (\text{SCL\_Tap} - 1) \times \text{tap2tap}] \qquad \textit{Eqn. 20-4}$$

Table 20-7. I^2C Divider and Hold Values

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129
	24	288	49	142	145
	25	320	49	158	161
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481
	30	640	65	318	321
	31	768	65	382	385
	32	896	129	446	449
	33	1024	129	510	513
	34	1152	193	574	577
	35	1280	193	638	641
	36	1536	257	766	769
	37	1920	257	958	961
	38	1280	129	638	641
	39	1536	129	766	769
	3A	1792	257	894	897
	3B	2048	257	1022	1025
	3C	2304	385	1150	1153
	3D	2560	385	1278	1281
	3E	3072	513	1534	1537
	3F	3840	513	1918	1921

Table 20-7. I^2C Divider and Hold Values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34
	4A	72	18	28	38
	4B	80	18	32	42
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	28	156	162
	61	384	28	188	194
	62	448	32	220	226
	63	512	32	252	258
	64	576	36	284	290
	65	640	36	316	322
	66	768	40	380	386
	67	960	40	476	482
	68	640	28	316	322
	69	768	28	380	386
	6A	896	36	444	450
	6B	1024	36	508	514
	6C	1152	44	572	578
	6D	1280	44	636	642
	6E	1536	52	764	770
	6F	1920	52	956	962
	70	1280	36	636	642
	71	1536	36	764	770
	72	1792	52	892	898
	73	2048	52	1020	1026
	74	2304	68	1148	1154
	75	2560	68	1276	1282
	76	3072	84	1532	1538
	77	3840	84	1916	1922
	78	2560	36	1276	1282
	79	3072	36	1532	1538
	7A	3584	68	1788	1794
	7B	4096	68	2044	2050
	7C	4608	100	2300	2306
	7D	5120	100	2556	2562
	7E	6144	132	3068	3074
	7F	7680	132	3836	3842

Table 20-7. I^2C Divider and Hold Values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132
	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212
	97	512	84	232	260
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
	30	2560	260	1272	1284
	B1	3072	260	1528	1540
	B2	3584	516	1784	1796
	B3	4096	516	2040	2052
	B4	4608	772	2296	2308
	B5	5120	772	2552	2564
	B6	6144	1028	3064	3076
	B7	7680	1028	3832	3844
	B8	5120	516	2552	2564
	B9	6144	516	3064	3076
	BA	7168	1028	3576	3588
	BB	8192	1028	4088	4100
	BC	9216	1540	4600	4612
	BD	10240	1540	5112	5124
	BE	12288	2052	6136	6148
	BF	15360	2052	7672	7684

20.4.3.3 I^2C Bus Control Register

Offset 0x0002 Access: User read/write

	0	1	2	3	4	5	6	7
R	MDIS	IBIE	MS/$\overline{SL}$	Tx/$\overline{Rx}$	NOACK	0	DMAEN	0
W						RSTA		
Reset	1	0	0	0	0	0	0	0

Figure 20-6. I^2C Bus Control Register (IBCR)

Table 20-8. IBCR field descriptions

Field	Description
MDIS	Module disable. This bit controls the software reset of the entire I^2C Bus module. 0 The I^2C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed **Note:** If the I^2C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: Slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I^2C Bus module losing arbitration, after which, bus operation would return to normal.
IBIE	I-Bus Interrupt Enable. 0 Interrupts from the I^2C Bus module are disabled. Note that this does not clear any currently pending interrupt condition 1 Interrupts from the I^2C Bus module are enabled. An I^2C Bus interrupt occurs provided the IBIF bit in the status register is also set.
MS/$\overline{SL}$	Master/Slave mode select. Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus and the Master mode is selected. When this bit is changed from 1 to 0, a stop signal is generated and the operation mode changes from master to slave. A stop signal should be generated only if the IBIF flag is set. MS/$\overline{SL}$ is cleared without generating a stop signal when the master loses arbitration. 0 Slave mode 1 Master mode
Tx/$\overline{Rx}$	Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In Master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit
NOACK	Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I^2C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. Note that values written to this bit are used only when the I^2C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)

Table 20-8. IBCR field descriptions (continued)

Field	Description
RSTA	Repeat Start. Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 0 No effect 1 Generate repeat start cycle
DMAEN	DMA Enable. When this bit is set, the DMA TX and RX lines will be asserted when the I^2C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I^2C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See tSection 15.5, Initialization/application information, for more details. 0 Disable the DMA TX/RX request signals 1 Enable the DMA TX/RX request signals

20.4.3.4 I^2C Bus Status Register

Offset 0x0003 Access: Read-only any time[1]

	0	1	2	3	4	5	6	7
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

Figure 20-7. I^2C Bus Status Register (IBSR)

[1] With the exception of IBIF and IBAL, which are software clearable.

Table 20-9. IBSR field descriptions

Field	Description
TCF	Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the I^2C module or from the I^2C module. 1 Transfer complete 0 Transfer in progress
IAAS	Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set its Tx/$\overline{Rx}$ mode accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
IBB	Bus busy. This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a stop signal is detected, IBB is cleared and the bus enters idle state. 0 Bus is Idle 1 Bus is busy

Table 20-9. IBSR field descriptions (continued)

Field	Description
IBAL	Arbitration Lost. The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: • SDA is sampled low when the master drives a high during an address or data transmit cycle. • SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in Slave mode. • A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
SRW	Slave Read/Write. When IAAS is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in Slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By programming this bit, the CPU can select slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
IBIF	I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs: • Arbitration lost (IBAL bit set) • Byte transfer complete (TCF bit set) • Addressed as slave (IAAS bit set) • NoAck from Slave (MS and Tx bits set) • I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit.
RXAK	Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

20.4.3.5 I^2C Bus Data I/O Register

Offset 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7
R W	DATA							
Reset	0	0	0	0	0	0	0	0

Figure 20-8. I^2C Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In Slave mode, the same functions are available after an address match has occurred. Note that the Tx/$\overline{Rx}$ bit in the IBCR must correctly reflect the desired direction of transfer in master and Slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to the IBDR following assertion of MS/$\overline{SL}$ is used for the address transfer and should comprise the calling address (in position D7–D1) concatenated with the required R/$\overline{W}$ bit (in position D0).

20.4.3.6 I^2C Bus Interrupt Configuration Register

Offset 0x0005 Access: User read/write

	0	1	2	3	4	5	6	7
R	BIIE	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-9. I^2C Bus Interrupt Configuration Register (IBIC)

Table 20-10. IBIC field descriptions

Field	Description
BIIE	Bus Idle Interrupt Enable bit. This configuration bit can be used to enable the generation of an interrupt once the I^2C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a stop on the I^2C bus. 0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled

20.5 Functional description

20.5.1 General

This section provides a complete functional description of the Inter-Integrated Circuit (I^2C).

20.5.2 I-Bus Protocol

The I^2C Bus system uses a Serial Data line (SDA) and a Serial Clock Line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logical AND function is exercised on both lines with external pullup resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and stop signal. They are described briefly in the following sections and illustrated in Figure 20-10.

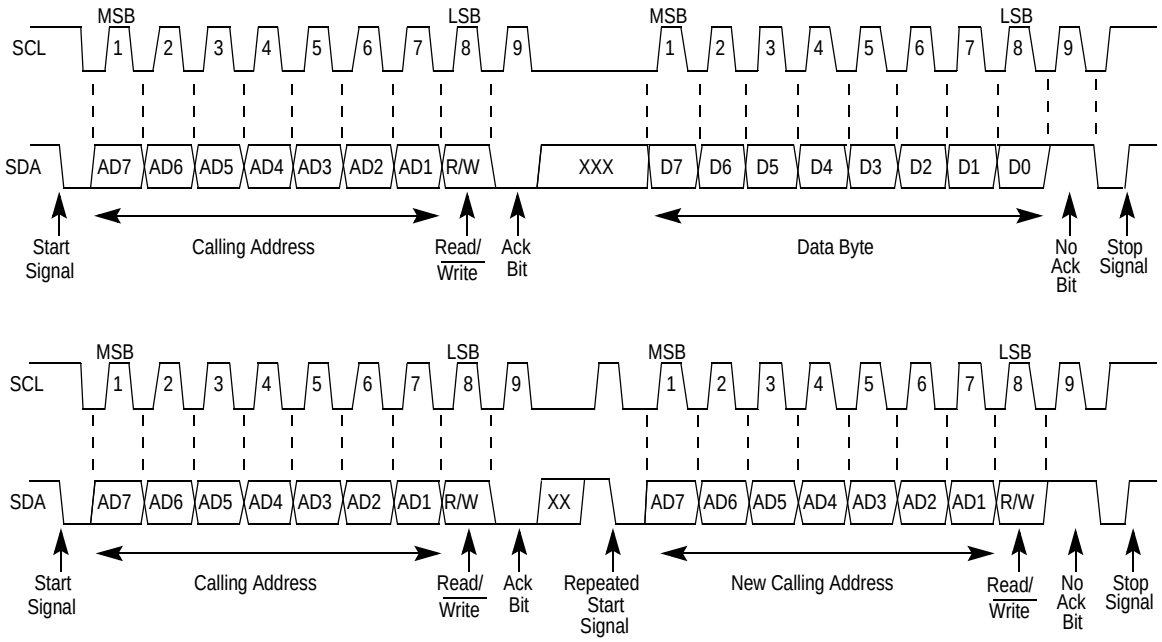


Figure 20-10. I^2C Bus Transmission Signals

20.5.2.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 20-10, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

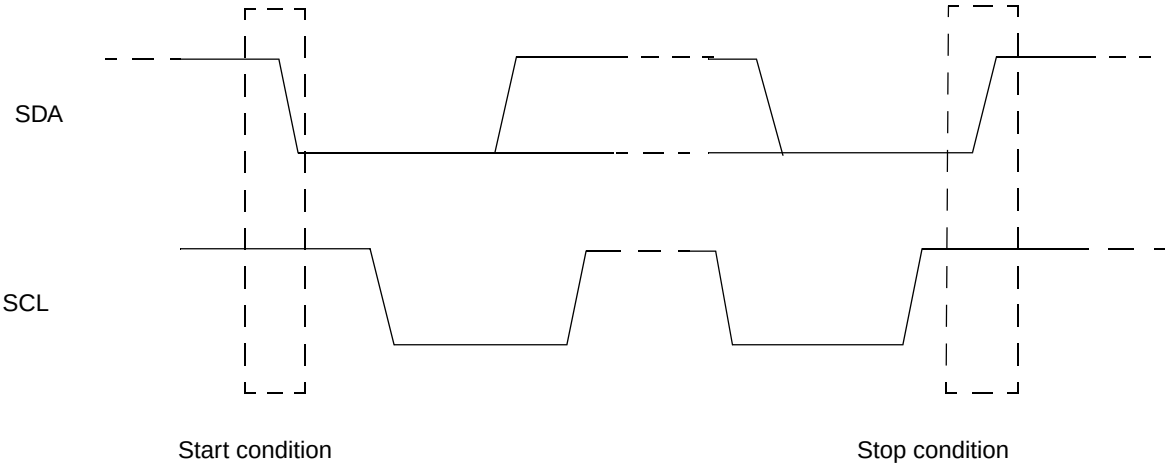


Figure 20-11. Start and stop conditions

20.5.2.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer—the slave transmits data to the master

0 = Write transfer—the master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 20-10).

No two slaves in the system may have the same address. If the I^2C Bus is master, it must not transmit an address that is equal to its own slave address. The I^2C Bus cannot be master and slave at the same time. However, if arbitration is lost during an address cycle the I^2C Bus will revert to Slave mode and operate correctly, even if it is being addressed by another master.

20.5.2.3 Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 20-10. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte must be followed by an acknowledge bit, which is signaled from the receiving device by pulling the SDA low at the ninth clock. Therefore, one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate a stop or START signal.

20.5.2.4 Stop Signal

The master can terminate the communication by generating a stop signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a stop signal first. This is called repeated START. A stop signal is defined as a low-to-high transition of SDA while SCL is at logical “1” (see Figure 20-10).

The master can generate a stop even if the slave has generated an acknowledge, at which point the slave must release the bus.

20.5.2.5 Repeated START Signal

As shown in Figure 20-10, a repeated START signal is a START signal generated without first generating a stop signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

20.5.2.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. The losing masters immediately switch over to slave receive mode and stop driving the SDA output. In this case, the transition from master to Slave mode does not generate a stop condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

20.5.2.7 Clock Synchronization

Since wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 20-12). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

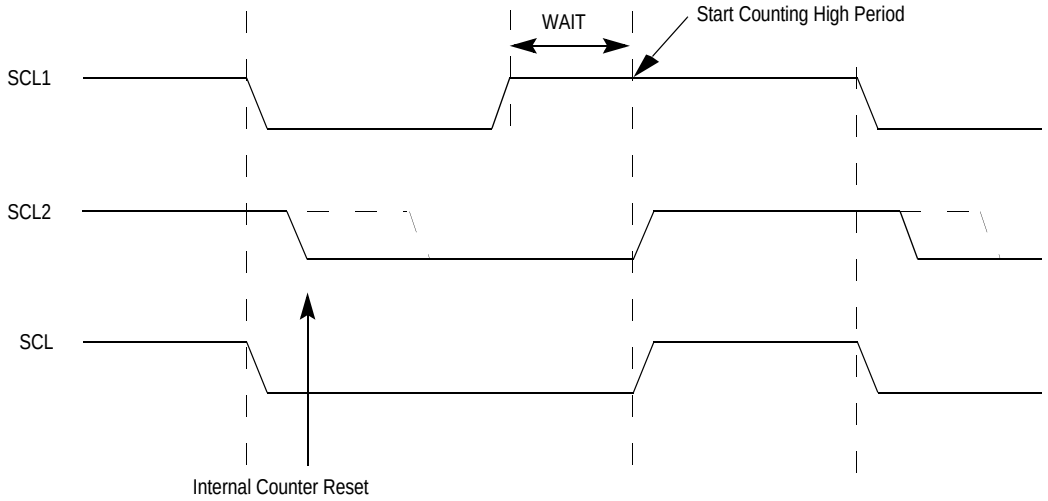


Figure 20-12. I^2C Bus Clock Synchronization

20.5.2.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

20.5.2.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

20.5.3 Interrupts

20.5.3.1 General

The I^2C module uses only one interrupt vector.

Table 20-11. Interrupt Summary

Interrupt	Offset	Vector	Priority	Source	Description
I^2C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the deassertion of IBB can also cause an interrupt, indicating that the bus is idle.

20.5.3.2 Interrupt Description

There are five types of internal interrupts in the I^2C. The interrupt service routine can determine the interrupt type by reading the Status Register.

I^2C Interrupt can be generated on

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I^2C interrupt is enabled by the IBIE bit in the I^2C Control Register. It must be cleared by writing 1 to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the BIIE bit in the IBIC register.

20.6 Initialization/application information

20.6.1 I^2C Programming Examples

20.6.1.1 Initialization Sequence

Reset will put the I^2C Bus Control Register to its default state. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the Frequency Divider Register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the I^2C Bus Address Register (IBAD) to define its slave address.
3. Clear the IBDIS bit of the I^2C Bus Control Register (IBCR) to enable the I^2C interface system.
4. Modify the bits of the I^2C Bus Control Register (IBCR) to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not. Optionally also modify the bits of the I^2C Bus Interrupt Config Register (IBIC) to further refine the interrupt behavior.

20.6.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I^2C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a stop condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (bit 5, IBSR ==1)// wait in loop for IBB flag to clear
bit4 and bit 5, IBCR = 1// set transmit and Master mode, i.e. generate start condition
IBDR = calling_address// send the calling address to the data register
while (bit 5, IBSR ==0)// wait in loop for IBB flag to be set
```

20.6.1.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I^2C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing 1 (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress by reading the IBDR data register in receive mode or writing the IBDR in transmit mode. The TCF bit should not be used as a data transfer

complete flag as the flag timing is dependent on a number of factors including the I^2C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I^2C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During Slave mode address cycles (IAAS=1) the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For Slave mode data cycles (IAAS=0) the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example software sequence for 'master transmitter' in the interrupt routine.

```
clear bit 1, IBSR// Clear the IBIF flag
if (bit 5, IBCR ==0)
slave_mode()// run Slave mode routine
if (bit 4, IBCR ==0))
receive_mode()// run receive_mode routine
if (bit 0, IBSR == 1)// if NO ACK
     end();// end transmission
else
IBDR = data_to_transmit// transmit next byte of data
```

20.6.1.4 Generation of stop

A data transfer ends with a stop signal generated by the 'master' device. A master transmitter can simply generate a stop signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
if (tx_count == 0) or// check to see if all data bytes have been transmitted
    (bit 0, IBSR == 1) {// or if no ACK generated
         clear bit 5, IBCR// generate stop condition
        }
else {
IBDR = data_to_transmit// write byte of data to DATA register
         tx_count --// decrement counter
      }// return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the second last byte of data. Before reading the last byte of data, a stop signal must first be generated. The following is an example showing how a stop signal is generated by a master receiver.

```
rx_count --// decrease the rx counter
if (rx_count ==1)// 2nd last byte to be read ?
   bit 3, IBCR = 1// disable ACK
if (rx_count == 0)// last byte to be read ?
  bit 1, IBCR = 0// generate stop signal
else
```

```
data_received = IBDR// read RX data and store
```

20.6.1.5 Generation of repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a stop signal. A program example is as shown.

```
bit 2, IBCR = 1// generate another start ( restart)
IBDR == calling_address// transmit the calling address
```

20.6.1.6 Slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a stop signal.

20.6.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a stop condition, generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

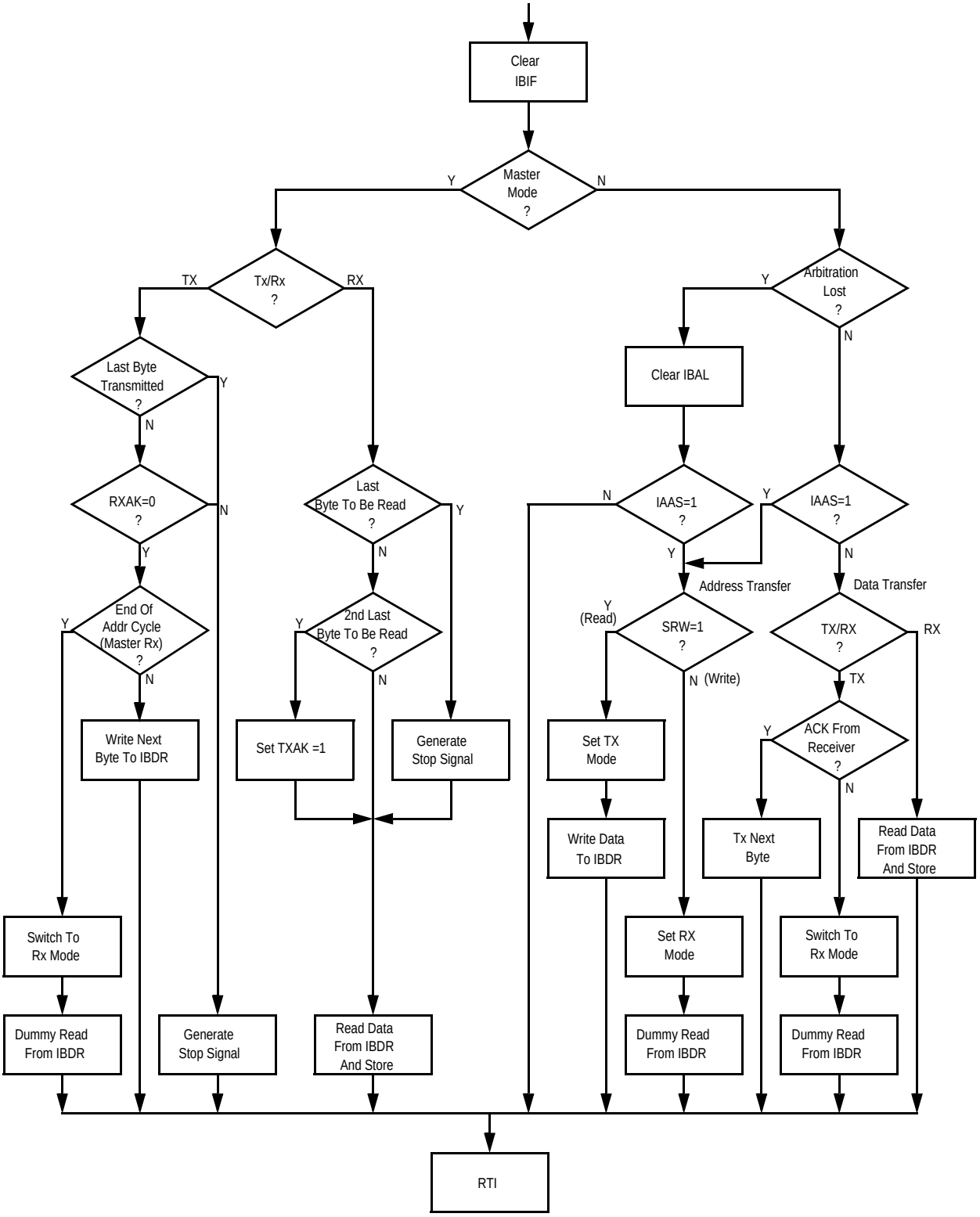


Figure 20-13. Flow-Chart of Typical I^2C Interrupt Routine

20.6.2 DMA application information

The DMA interface on the I2I^{2}CC is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the IBCR[DMAEN] bit is not set when the I^2C module is configured in Master mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C block.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The IBCR[DMAEN] bit works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, depending on the setting of the DMAEN bit. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

Chapter 21
Interrupt Controller (INTC)

21.1 Introduction

The INTC provides priority-based preemptive scheduling of interrupt service requests (ISRs). This scheduling scheme is suitable for statically scheduled hard real-time systems. The INTC supports 122 interrupt requests. It is targeted to work with Power Architecture technology and automotive applications where the ISRs nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, the time from the assertion of the peripheral's interrupt request to when the processor is performing useful work to service the interrupt request needs to be minimized. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 16 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Because each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the priority ceiling protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource cannot preempt each other.

Multiple processors can assert interrupt requests to each other through software configurable interrupt requests. These software configurable interrupt requests can also be used to separate the work involved in servicing an interrupt request into a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software configurable interrupt request to finish the servicing in a lower priority ISR. Therefore these software configurable interrupt requests can be used instead of the peripheral ISR scheduling a task through the RTOS.

21.2 Features

- Supports 114 peripheral and 8 software-configurable interrupt request sources
- Unique 9-bit vector per interrupt source
- Each interrupt source can be programmed to one of 16 priorities
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR at a higher priority preempts ISRs or tasks at lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the priority ceiling protocol for accessing shared resources.
- Low latency—three clocks from receipt of interrupt request from peripheral to interrupt request to processor

Table 21-1. Interrupt sources available

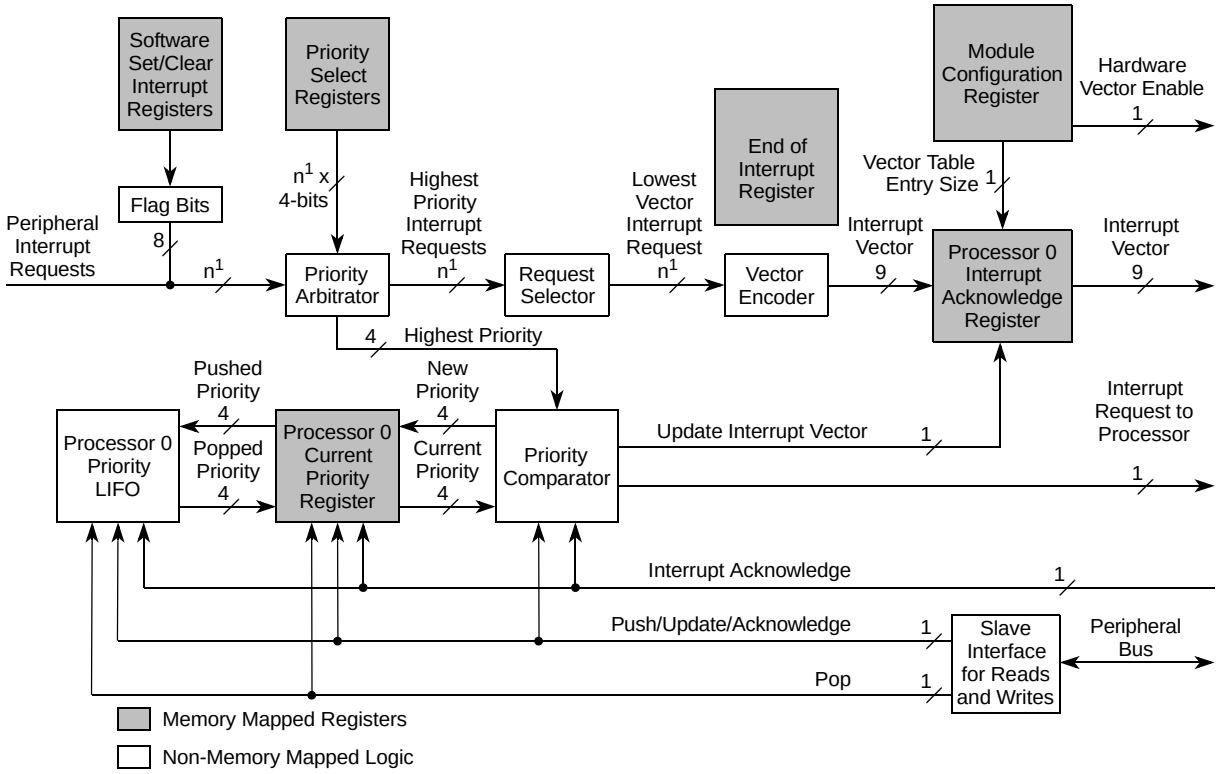
Interrupt sources	Number Available
Software	8
ECSM	3
eDMA	17
SWT	1
STM	4
Real Time Counter (RTC/API)	2
System Integration Unit Lite (SIUL)	2
WakeUp Unit (WKPU)	3
MC_ME	4
MC_RGM	1
FXOSC	1
PIT	4
Analog to Digital Converter 0 (ADC0)	3
FlexCAN 0 (CAN0)	9
FlexCAN 1 (CAN1)	9
DSPI 0	5
DSPI 1	5
LINFlex 0	3
LINFlex 1	3
Inter-IC Bus Interface Controller 0 (I2C0)	1
Inter-IC Bus Interface Controller 1 (I2C1)	1
Inter-IC Bus Interface Controller 2 (I2C2)	1
Inter-IC Bus Interface Controller 3 (I2C3)	1
Enhanced Modular I/O Subsystem 0 (eMIOS0)	8
Enhanced Modular I/O Subsystem 1 (eMIOS1)	4
Sound Generation Logic (SGL)	1
Display Control Unit (DCU)	4
Stepper Motor Driver (SMD0)	1
Stepper Stall Detect 0 (SSD0)	1
Stepper Stall Detect 1 (SSD1)	1
Stepper Stall Detect 2 (SSD2)	1
Stepper Stall Detect3 (SSD3)	1
Stepper Stall Detect 4 (SSD4)	1

Table 21-1. Interrupt sources available (continued)

Interrupt sources	Number Available
Stepper Stall Detect 5 (SSD5)	1
Liquid Crystal Display 0 (LCD0)	1
QuadSPI	6

21.3 Block diagram

Figure 21-1 is a block diagram of the interrupt controller (INTC).



1 The total number of interrupt sources is 122, which includes 16 reserved sources and 8 software sources.

Figure 21-1. INTC block diagram

21.4 Modes of operation

21.4.1 Normal mode

In normal mode, the INTC has two handshaking modes with the processor: software vector mode and hardware vector mode.

21.4.1.1 Software vector mode

In software vector mode, the interrupt exception handler software must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN bit in INTC_MCR is negated. The hardware vector enable signal to processor 0 or processor 1 is driven as negated when its associated HVEN bit is negated. The vector is read from INC_IACKR. Reading the INTC_IACKR negates the interrupt request to the associated processor. Even if a higher priority interrupt request arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in INTC_CPR onto the associated LIFO and updates PRI in the associated INTC_CPR with the new priority.

Furthermore, the interrupt vector to the processor is driven as all 0s. The interrupt acknowledge signal from the associated processor is ignored.

21.4.1.2 Hardware vector mode

In hardware vector mode, the hardware signals the interrupt vector from the INTC in conjunction with a processor that can use that vector. This hardware causes the first instruction to be executed in handling the interrupt request to the processor to be specific to that vector. Therefore, the interrupt exception handler is specific to a peripheral or software configurable interrupt request rather than being common to all of them. The INTC uses hardware vector mode for a given processor when the associated HVEN bit in the INTC_MCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted. When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software configurable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR field in the INTC_IACKR, depending on which processor was assigned to handle a given interrupt source.

The processor negates the interrupt request to the processor driven by the INTC by asserting the interrupt acknowledge signal for one clock. Even if a higher priority interrupt request arrived while waiting for the interrupt acknowledge, the interrupt request to the processor will negate for at least one clock.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR does not occur when the associated interrupt acknowledge signal asserts and INTC_SSCIR0_3–INTC_SSCIR4_7 is written at a time such that the PRI value in the associated INTC_CPR register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR is updated with the new priority, and the associated LIFO is neither pushed or popped.

21.4.1.3 Debug mode

The INTC operation in debug mode is identical to its operation in normal mode.

21.4.1.4 Stop mode

The INTC supports stop mode. The INTC can have its clock input disabled at any time by the clock driver on the device. While its clocks are disabled, the INTC registers are not accessible.

The INTC requires clocking in order for a peripheral interrupt request to generate an interrupt request to the processor. Since the INTC is not clocked in stop mode, peripheral interrupt requests cannot be used as a wakeup source, unless the clock, reset, and power module (CRP) supports that interrupt request as a wakeup source.

21.5 Memory map and register description

21.5.1 Module memory map

Table 21-2 shows the INTC memory map.

Table 21-2. INTC memory map

Offset from INTC_BASE_ADDR[1]	Register	Access	Reset Value	Location
0x0000	INTC_MCR—INTC module configuration register	R/W	0x0000_0000	on page 756
0x0004	Reserved			
0x0008	INTC_CPR—INTC current priority register	R/W	0x0000_000F	on page 756
0x00C	Reserved			
0x0010	INTC_IACKR—INTC interrupt acknowledge register	R[2]/W	0x0000_0000	on page 758
0x0014	Reserved			
0x0018	INTC_EOIR—INTC end of interrupt register	W	0x0000_0000	on page 759
0x001C	Reserved			
0x0020–0x0027	INTC_SSCIR[0:7]—INTC software set/clear interrupt register [0:7]	R/W	0x0000_0000	on page 759
0x0028–0x003C	Reserved			
0x0040–0x010C	INTC_PSRn -INTC priority select register [0:206][3]	R/W	0x0000_0000	on page 761

1 INTC_BASE_ADDR = 0xFFF4_8000

2 When the HVEN bit in the INTC module configuration register (INTC_MCR) is asserted, a read of the INTC_IACKR has no side effects.

3 The PRI fields are “reserved” for peripheral interrupt requests whose vectors are labeled as Reserved in Figure 21-3

21.5.2 Register description

With exception of the INTC_SSCI*n* and INTC_PSR*n*, all registers are 32 bits in width. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not

cross a register boundary. These supported accesses include types and sizes of eight bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCI*n* and INTC_PSR*n* are 8 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

In software vector mode, the side effects of a read of INTC_IACKR are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to either INTC_SSCIR0_3–INTC_SSCIR4_7 or INTC_EOIR does not affect the operation of the write.

21.5.2.1 INTC Module Configuration Register (INTC_MCR)

The module configuration register is used to configure options of the INTC.

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	VTES	0	0	0	0	HVEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-2. INTC Module Configuration Register (INTC_MCR)

Table 21-3. INTC_MCR field descriptions

Field	Description
26 VTES	Vector table entry size. Controls the number of 0s to the right of INTVEC in Section 21.5.2.3, INTC Interrupt Acknowledge Register (INTC_IACKR)). If the contents of INTC_IACKR are used as an address of an entry in a vector table as in software vector mode, then the number of rightmost 0s will determine the size of each vector table entry. VTES impacts software vector mode operation but also affects INTC_IACKR[INTVEC] position in both hardware vector mode and software vector mode. 0 4 bytes. 1 8 bytes.
31 HVEN	Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. Refer to Section 21.4, Modes of operation, for the details of the handshaking with the processor in each mode. 0 Software vector mode. 1 Hardware vector mode.

21.5.2.2 INTC Current Priority Register for Processor (INTC_CPR)

Offset: 0x0008 Access: User read/write

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

Figure 21-3. INTC Current Priority Register (INTC_CPR)

Table 21-4. INTC_CPR field descriptions

Field	Description
28–31 PRI[0:3]	Priority. PRI is the priority of the currently executing ISR according to the field values defined in Table 21-5.

The INTC_CPR masks any peripheral or software settable interrupt request set at the same or lower priority as the current value of the INTC_CPR[PRI] field from generating an interrupt request to the processor. When the INTC interrupt acknowledge register (INTC_IACKR) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR) is written, the LIFO is popped into the INTC_CPR's PRI field.

The masking priority can be raised or lowered by writing to the PRI field, supporting the PCP. Refer to Section 21.7.5, Priority Ceiling Protocol.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource can result in a non-coherent access to the resource. Refer to Section 21.7.5.2, Ensuring Coherency, for example code to ensure coherency.

Table 21-5. PRI Values

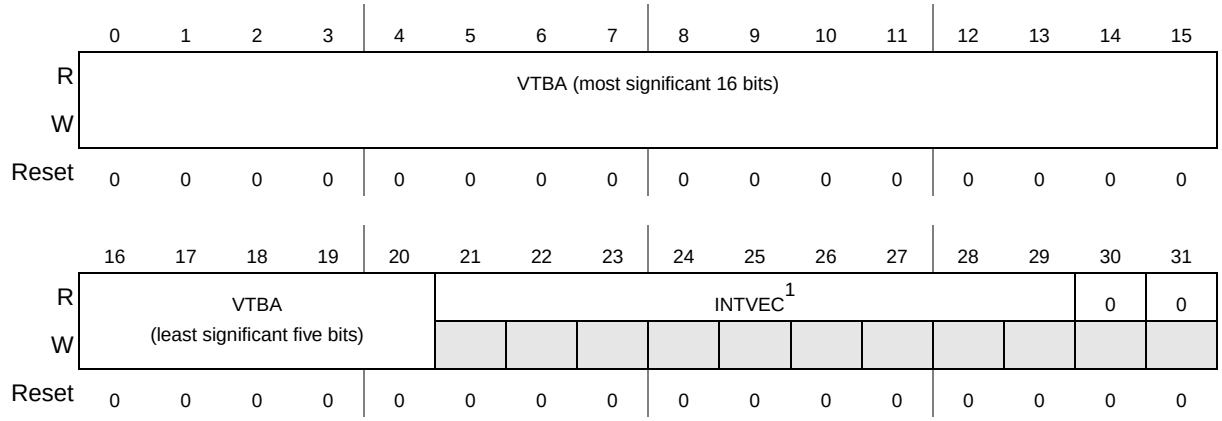
PRI	Meaning
1111	Priority 15—highest priority
1110	Priority 14
1101	Priority 13
1100	Priority 12
1011	Priority 11
1010	Priority 10
1001	Priority 9
1000	Priority 8
0111	Priority 7

Table 21-5. PRI Values (continued)

PRI	Meaning
0110	Priority 6
0101	Priority 5
0100	Priority 4
0011	Priority 3
0010	Priority 2
0001	Priority 1
0000	Priority 0—lowest priority

21.5.2.3 INTC Interrupt Acknowledge Register (INTC_IACKR)

Offset: 0x0010 Access: User read/write



1 When the VTES bit in INTC_MCR is asserted, INTVEC is shifted to the left one bit. Bit 29 is read as a 0. VTBA is narrowed to 20 bits in width.

Figure 21-4. INTC Interrupt Acknowledge Register (INTC_IACKR)

Table 21-6. INTC_IACKR field descriptions

Field	Description
0–20 or 0–19 VTBA	Vector Table Base Address. Can be the base address of a vector table of addresses of ISRs. The VTBA only uses the leftmost 20 bits when the VTES bit in INTC_MCR is asserted.
21–29 or 20–28 INTVEC	Interrupt Vector.It is the vector of the peripheral or software configurable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode. **Note:** If INTC_MCR[VTES] = 1, then the INTVEC field is shifted left one position to bits 20–28. VTBA is then shortened by one bit to bits 0–19.

The interrupt acknowledge register provides a value that can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

In software vector mode, the INTC_IACKR has side effects from reads. Therefore, it must not be speculatively read while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR does not have side effects in hardware vector mode.

21.5.2.4 INTC End-of-Interrupt Register (INTC_EOIR)

Offset 0x0018 Access: write only

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

Figure 21-5. INTC End-of-Interrupt Register (INTC_EOIR)

Writing to the end-of-interrupt register signals the end of the servicing of the interrupt request. When the INTC_EOIR is written, the priority last pushed on the LIFO is popped into INTC_CPR. An exception to this behavior is described in Section 21.4.1.2, Hardware vector mode. The values and size of data written to the INTC_EOIR are ignored. The values and sizes written to this register neither update the INTC_EOIR contents or affect whether the LIFO pops. For possible future compatibility, write four bytes of all 0s to the INTC_EOIR.

Reading the INTC_EOIR has no effect on the LIFO.

21.5.2.5 INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR4_7)

Offset: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	CLR0	0	0	0	0	0	0	0	CLR1
W							SET0								SET1	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	CLR2	0	0	0	0	0	0	0	CLR3
W							SET2								SET3	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-6. INTC Software Set/Clear Interrupt Register 0–3 (INTC_SSCIR[0:3])

Offset: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	CLR4	0	0	0	0	0	0	0	CLR5
W							SET4								SET5	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	CLR6	0	0	0	0	0	0	0	CLR7
W							SET6								SET7	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-7. INTC Software Set/Clear Interrupt Register 4–7 (INTC_SSCIR[4:7])

Table 21-7. INTC_SSCIR[0:7] field descriptions

Field	Description
6, 14, 22, 30 SET[0:7]	Set Flag Bits. Writing a 1 sets the corresponding CLR*x* bit. Writing a 0 has no effect. Each SET*x* always will be read as a 0.
7, 15, 23, 31 CLR[0:7]	Clear Flag Bits. CLR*x* is the flag bit. Writing a 1 to CLR*x* clears it provided that a 1 is not written simultaneously to its corresponding SET*x* bit. Writing a 0 to CLR*x* has no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

The software set/clear interrupt registers support the setting or clearing of software configurable interrupt request. These registers contain eight independent sets of bits to set and clear a corresponding flag bit by software. Excepting being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC like a peripheral interrupt request. Writing a 1 to SET*x* will leave SET*x* unchanged at 0 but sets CLR*x*. Writing a 0 to SET*x* has no effect. CLR*x* is the flag bit. Writing a 1 to CLR*x* clears it. Writing a 0 to CLR*x* has no effect. If a 1 is written simultaneously to a pair of SET*x* and CLR*x* bits, CLR*x* will be asserted, regardless of whether CLR*x* was asserted before the write.

21.5.2.6 INTC Priority Select Registers (INTC_PSR0_3–INTC_PSR204_207)

Offset: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PRI0				0	0	0	0	PRI1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PRI2				0	0	0	0	PRI3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-8. INTC Priority Select Register 0–3 (INTC_PSR[0:3])

Offset: 0x00D0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PRI204				0	0	0	0	PRI205			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PRI206				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-9. INTC Priority Select Register 204-206 (INTC_PSR[204:206])

Table 21-8. INTC_PSR0_3–INTC_PSR204-206 field descriptions

Field	Description
4–7, 12–15, 20–23, 28–31 PRI[0:3]–PRI204:206	Priority Select. PRI*x* selects the priority for interrupt requests. Refer to Section 21.6, Functional description.

Table 21-9. INTC Priority Select Register Address Offsets

INTC_PSR*x*_*x*	Offset Address
INTC_PSR0_3	0x0040
INTC_PSR4_7	0x0044
INTC_PSR8_11	0x0048

INTC_PSR*x*_*x*	Offset Address
INTC_PSR104_107	0x00A8
INTC_PSR108_111	0x00AC
INTC_PSR112_115	0x00B0

Table 21-9. INTC Priority Select Register Address Offsets (continued)

INTC_PSR*x*_*x*	Offset Address
INTC_PSR12_15	0x004C
INTC_PSR16_19	0x0050
INTC_PSR20_23	0x0054
INTC_PSR24_27	0x0058
INTC_PSR28_31	0x005C
INTC_PSR32_35	0x0060
INTC_PSR36_39	0x0064
INTC_PSR40_43	0x0068
INTC_PSR44_47	0x006C
INTC_PSR48_51	0x0070
INTC_PSR52_55	0x0074
INTC_PSR56_59	0x0078
INTC_PSR60_63	0x007C
INTC_PSR64_67	0x0080
INTC_PSR68_71	0x0084
INTC_PSR72_75	0x0088
INTC_PSR76_79	0x008C
INTC_PSR80_83	0x0090
INTC_PSR84_87	0x0094
INTC_PSR88_91	0x0098
INTC_PSR92_95	0x009C
INTC_PSR96_99	0x00A0
INTC_PSR100_103	0x00A4

INTC_PSR*x*_*x*	Offset Address
INTC_PSR116_119	0x00B4
INTC_PSR120_123	0x00B8
INTC_PSR124_127	0x00BC
INTC_PSR128_131	0x00C0
INTC_PSR132_135	0x00C4
INTC_PSR136_139	0x00C8
INTC_PSR140_143	0x00CC
INTC_PSR144_147	0x00D0
INTC_PSR148_151	0x00D4
INTC_PSR152_155	0x00D8
INTC_PSR156_159	0x00DC
INTC_PSR160_163	0x00E0
INTC_PSR164_167	0x00E4
INTC_PSR168_171	0x00E8
INTC_PSR172_175	0x00EC
INTC_PSR176_179	0x00F0
INTC_PSR180_183	0x00F4
INTC_PSR184_187	0x00F8
INTC_PSR188_191	0x00FC
INTC_PSR192_195	0x0100
INTC_PSR196_199	0x0104
INTC_PSR200_203	0x0108
INTC_PSR204_207	0x010C

21.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor.

NOTE

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software settable interrupt request, whose PRI*n* value in INTC_PSR0_3–INTC_PSR204_206 is higher than the PRI value in INTC_CPR, negates before the interrupt request to the processor for that peripheral or software settable interrupt request is acknowledged, the interrupt request to the processor still can assert or will remain asserted for that peripheral or software settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software settable interrupt request. Also, the PRI value in the INTC_CPR will be updated with the corresponding PRI*n* value in INTC_PSR*n*. Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral or, alternatively, setting its mask bit has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

Table 21-10. Interrupt vector table

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
Section A (Core Section)					
-	0x0000	16	X	Critical Input (INTC software vector mode)	Core
-	0x0010	16	X	Machine check / NMI	Core
-	0x0020	16	X	Data Storage	Core
-	0x0030	16	X	Instruction Storage	Core
-	0x0040	16	X	External Input (INTC software vector mode)	Core
-	0x0050	16	X	Alignment	Core
-	0x0060	16	X	Program	Core
-	0x0070	16	X	Reserved	Core
-	0x0080	16	X	System call	Core
-	0x0090	96	X	Unused	Core
-	0x00F0	16	X	Debug	Core
-	0x0100	1792	X	Unused	Core
Section B (On-Platform Peripherals)					

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
0	0x0800	4	X	software-settable flag 0	Software
1	0x0804	4	X	software-settable flag 1	Software
2	0x0808	4	X	software-settable flag 2	Software
3	0x080C	4	X	software-settable flag 3	Software
4	0x0810	4	X	software-settable flag 4	Software
5	0x0814	4	X	software-settable flag 5	Software
6	0x0818	4	X	software-settable flag 6	Software
7	0x081C	4	X	software-settable flag 7	Software
8	0x0820	4			Reserved
9	0x0824	4	X	Platform Flash Bank 0 Abort | Platform Flash Bank 0 Stall | Platform Flash Bank 1 Abort | Platform Flash Bank 1 Stall | Platform Flash Bank 2 Abort | Platform Flash Bank 2 Stall | Platform Flash Bank 3 Abort | Platform Flash Bank 3 Stall	ECSM
10	0x0828	4	X	Combined Error	DMA2x
11	0x082C	4	X	Channel 0	DMA2x
12	0x0830	4	X	Channel 1	DMA2x
13	0x0834	4	X	Channel 2	DMA2x
14	0x0838	4	X	Channel 3	DMA2x
15	0x083C	4	X	Channel 4	DMA2x
16	0x0840	4	X	Channel 5	DMA2x
17	0x0844	4	X	Channel 6	DMA2x
18	0x0848	4	X	Channel 7	DMA2x
19	0x084C	4	X	Channel 8	DMA2x
20	0x0850	4	X	Channel 9	DMA2x
21	0x0854	4	X	Channel 10	DMA2x

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
22	0x0858	4	X	Channel 11	DMA2x
23	0x085C	4	X	Channel 12	DMA2x
24	0x0860	4	X	Channel 13	DMA2x
25	0x0864	4	X	Channel 14	DMA2x
26	0x0868	4	X	Channel 15	DMA2x
27	0x086C	4			Reserved
28	0x0870	4	X	Timeout	Software Watchdog (SWT)
29	0x0874	4			Reserved
30	0x0878	4	X	Match on channel 0	STM
31	0x087C	4	X	Match on channel 1	STM
32	0x0880	4	X	Match on channel 2	STM
33	0x0884	4	X	Match on channel 3	STM
34	0x0888	4			Reserved
35	0x088C	4	X	ECC_DBD_PlatformFlash | ECC_DBD_PlatformRAM	ECSM
36	0x0890	4	X	ECC_SBC_PlatformFlash | ECC_SBC_PlatformRAM	ECSM
37	0x0894	4			Reserved
Section C					
38	0x0898	4	X	RTC	Real Time Counter (RTC/API)
39	0x089C	4	X	API	Real Time Counter (RTC/API)
40	0x08A0	4			Reserved
41	0x08A4	4	X	SIU External IRQ_0	System Integration Unit Lite (SIUL)
42	0x08A8	4	X	SIU External IRQ_1	System Integration Unit Lite (SIUL)
43	0x08AC	4			Reserved
44	0x08B0	4			Reserved
45	0x08B4	4			Reserved

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
46	0x08B8	4	X	WakeUp_IRQ_0	WakeUp Unit (WKPU)
47	0x08BC	4	X	WakeUp_IRQ_1	WakeUp Unit (WKPU)
48	0x08C0	4	X	WakeUp_IRQ_2	WakeUp Unit (WKPU)
50	0x08C4	4			Reserved
50	0x08C8	4			Reserved
51	0x08CC	4	X	Safe Mode Interrupt	MC_ME
52	0x08D0	4	X	Mode Transition Interrupt	MC_ME
53	0x08D4	4	X	Invalid Mode Interrupt	MC_ME
54	0x08D8	4	X	Invalid Mode Config	MC_ME
55	0x08DC	4	O		Reserved
56	0x08E0	4	X	Functional and destructive reset alternate event interrupt (ipi_int)	MC_RGM
57	0x08E4	4	X	FXOSC counter expired (ipi_int_osc)	FXOSC
58	0x08E8	4	O		Reserved
59	0x08EC	4	X	PITimer Channel 0	Periodic Interrupt Timer (PIT)
60	0x08F0	4	X	PITimer Channel 1	Periodic Interrupt Timer (PIT)
61	0x08F4	4	X	PITimer Channel 2	Periodic Interrupt Timer (PIT)
62	0x08F8	4	X	ADC_EOC	Analog to Digital Converter 0 (ADC0)
63	0x08FC	4	X	ADC_ER	Analog to Digital Converter 0 (ADC0)
64	0x0900	4	X	ADC_WD	Analog to Digital Converter 0 (ADC0)
65	0x0904	4	X	FLEXCAN_ESR[ERR_INT]	FlexCAN 0 (CAN0)
66	0x0908	4	X	FLEXCAN_ESR_BOFF | FLEXCAN_Transmit_Warning | FLEXCAN_Receive_Warning	FlexCAN 0 (CAN0)
67	0x090C	4	O		Reserved

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
68	0x0910	4	X	FLEXCAN_BUF_00_03	FlexCAN 0 (CAN0)
69	0x0914	4	X	FLEXCAN_BUF_04_07	FlexCAN 0 (CAN0)
70	0x0918	4	X	FLEXCAN_BUF_08_11	FlexCAN 0 (CAN0)
71	0x091C	4	X	FLEXCAN_BUF_12_15	FlexCAN 0 (CAN0)
72	0x0920	4	X	FLEXCAN_BUF_16_31	FlexCAN 0 (CAN0)
73	0x0924	4	X	FLEXCAN_BUF_32_63	FlexCAN 0 (CAN0)
74	0x0928	4	X	DSPI_SR[TFUF] DSPI_SR[RFOF]	DSPI 0
75	0x092C	4	X	DSPI_SR[EOQF]	DSPI 0
76	0x0930	4	X	DSPI_SR[TFFF]	DSPI 0
77	0x0934	4	X	DSPI_SR[TCF]	DSPI 0
78	0x0938	4	X	DSPI_SR[RFDF]	DSPI 0
79	0x093C	4	X	LINFlex_RXI	LINFlex 0
80	0x0940	4	X	LINFlex_TXI	LINFlex 0
81	0x0944	4	X	LINFlex_ERR	LINFlex 0
82	0x0948	4	O		Reserved
83	0x094C	4	O		Reserved
84	0x0950	4	O		Reserved
85	0x0954	4	X	FLEXCAN_ESR[ERR_INT]	FlexCAN 1 (CAN1)
86	0x0958	4	X	FLEXCAN_ESR_BOFF | FLEXCAN_Transmit_Warning | FLEXCAN_Receive_Warning	FlexCAN 1 (CAN1)
87	0x095C	4	O		Reserved
88	0x0960	4	X	FLEXCAN_BUF_00_03	FlexCAN 1 (CAN1)
89	0x0964	4	X	FLEXCAN_BUF_04_07	FlexCAN 1 (CAN1)
90	0x0968	4	X	FLEXCAN_BUF_08_11	FlexCAN 1 (CAN1)
91	0x096C	4	X	FLEXCAN_BUF_12_15	FlexCAN 1 (CAN1)

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
92	0x0970	4	X	FLEXCAN_BUF_16_31	FlexCAN 1 (CAN1)
93	0x0974	4	X	FLEXCAN_BUF_32_63	FlexCAN 1 (CAN1)
94	0x0978	4	X	DSPI_SR[TFUF] DSPI_SR[RFOF]	DSPI 1
95	0x097C	4	X	DSPI_SR[EOQF]	DSPI 1
96	0x0980	4	X	DSPI_SR[TFFF]	DSPI 1
97	0x0984	4	X	DSPI_SR[TCF]	DSPI 1
98	0x0988	4	X	DSPI_SR[RFDF]	DSPI 1
99	0x098C	4	X	LINFlex_RXI	LINFlex 1
100	0x0990	4	X	LINFlex_TXI	LINFlex 1
101	0x0994	4	X	LINFlex_ERR	LINFlex 1
102	0x0998	4	O		Reserved
103	0x099C	4	O		Reserved
104	0x09A0	4	O		Reserved
105	0x09A4	4	O		Reserved
106	0x09A8	4	O		Reserved
107	0x09AC	4	O		Reserved
108	0x09B0	4	O		Reserved
109	0x09B4	4	O		Reserved
110	0x09B8	4	O		Reserved
111	0x09BC	4	O		Reserved
112	0x09C0	4	O		Reserved
113	0x09C4	4	O		Reserved
114	0x09C8	4	O		Reserved
115	0x09CC	4	O		Reserved
116	0x09D0	4	O		Reserved
117	0x09D4	4	O		Reserved

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
118	0x09D8	4	O		Reserved
119	0x09DC	4	O		Reserved
120	0x09E0	4	O		Reserved
121	0x09E4	4	O		Reserved
122	0x09E8	4	O		Reserved
123	0x09EC	4	O		Reserved
124	0x09F0	4	O		Reserved
125	0x09F4	4	X	IBIF	Inter-IC Bus Interface Controller 0 (I2C0)
126	0x09F8	4	X	IBIF	Inter-IC bus interface Controller 1 (I2C1)
127	0x09FC	4	X	PITimer Channel 3	Periodic Interrupt Timer (PIT)
128	0x0A00	4	O		Reserved
129	0x0A04	4	O		Reserved
130	0x0A08	4	O		Reserved
131	0x0A0C	4	O		Reserved
132	0x0A10	4	O		Reserved
133	0x0A14	4	O		Reserved
134	0x0A18	4	O		Reserved
135	0x0A1C	4	O		Reserved
136	0x0A20	4	O		Reserved
137	0x0A24	4	O		Reserved
138	0x0A28	4	O		Reserved
139	0x0A2C	4	O		Reserved
140	0x0A30	4	O		Reserved
141	0x0A34	4	X	EMIOS_GFR[F8,F9]	Enhanced Modular I/O Subsystem 0 (eMIOS0)

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
142	0x0A38	4	X	EMIOS_GFR[F10,F11]	Enhanced Modular I/O Subsystem 0 (eMIOS0)
143	0x0A3C	4	X	EMIOS_GFR[F12,F13]	Enhanced Modular I/O Subsystem 0 (eMIOS0)
144	0x0A40	4	X	EMIOS_GFR[F14,F15]	Enhanced Modular I/O Subsystem 0 (eMIOS0)
145	0x0A44	4	X	EMIOS_GFR[F16,F7]	Enhanced Modular I/O Subsystem 0 (eMIOS0)
146	0x0A48	4	X	EMIOS_GFR[F18,F19]	Enhanced Modular I/O Subsystem 0 (eMIOS0)
147	0x0A4C	4	X	EMIOS_GFR[F20,F21]	Enhanced Modular I/O Subsystem 0 (eMIOS0)
148	0x0A50	4	X	EMIOS_GFR[F22,F23]	Enhanced Modular I/O Subsystem 0 (eMIOS0)
149	0x0A54	4	O		Reserved
150	0x0A58	4	O		Reserved
151	0x0A5C	4	O		Reserved
152	0x0A60	4	O		Reserved
153	0x0A64	4	O		Reserved
154	0x0A68	4	O		Reserved
155	0x0A6C	4	O		Reserved
156	0x0A70	4	O		Reserved
Section D (Device-Specific Section)					
157	0x0A74	4	X	EMIOS_GFR[F16,F17]	Enhanced Modular I/O Subsystem 1 (eMIOS1)
158	0x0A78	4	X	EMIOS_GFR[F18,F19]	Enhanced Modular I/O Subsystem 1 (eMIOS1)
159	0x0A7C	4	X	EMIOS_GFR[F20,F21]	Enhanced Modular I/O Subsystem 1 (eMIOS1)
160	0x0A80	4	X	EMIOS_GFR[F22,F23]	Enhanced Modular I/O Subsystem 1 (eMIOS1)

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
161	0x0A84	4	O		Reserved
162	0x0A88	4	O		Reserved
163	0x0A8C	4	O		Reserved
164	0x0A90	4	O		Reserved
165	0x0A94	4	O		Reserved
166	0x0A98	4	O		Reserved
167	0x0A9C	4	O		Reserved
168	0x0AAO	4	O		Reserved
169	0x0AA4	4	O		Reserved
170	0x0AA8	4	O		Reserved
171	0x0AAC	4	O		Reserved
172	0x0AB0	4	O		Reserved
173	0x0AB4	4	X	IBIF	Inter-IC bus interface Controller 2 (I2C2)
174	0x0AB8	4	X	IBIF	Inter-IC Bus Interface Controller 3 (I2C3)
175	0x0ABC	4	O		Reserved
176	0x0AC0	4	O		Reserved
177	0x0AC4	4	O		Reserved
178	0x0AC8	4	O		Reserved
179	0x0ACC	4	O		Reserved
180	0x0AD0	4	O		Reserved
181	0x0AD4	4	O		Reserved
182	0x0AD8	4	O		Reserved
183	0x0ADC	4	X	SDCI	Sound Generation Logic (SGL)
184	0x0AE0	4	X	VS_BLANK, LS_BF_VS, VSYNC	Display Control Unit (DCU0)
185	0x0AE4	4	X	UNDRUN	Display Control Unit (DCU0)

Table 21-10. Interrupt vector table (continued)

IRQ #	Offset	Size [Byte]	This Device	Resource	Module
186	0x0AE8	4	X	PARERR	Display Control Unit (DCU0)
187	0x0AEC	4	X	PDI	Display Control Unit (DCU0)
188	0x0AF0	4	O		Reserved
189	0x0AF4	4	O		Reserved
190	0x0AF8	4	O		Reserved
191	0x0AFC	4	O		Reserved
192	0x0B00	4	X	MCTOI, SCDetect[0:23]	Stepper Motor Driver (SMD0)
193	0x0B04	4	X	BLNIF, ITGIF, and ACOVIF	Stepper Stall Detect 0 (SSD0)
194	0x0B08	4	X	BLNIF, ITGIF, and ACOVIF	Stepper Stall Detect 1 (SSD1)
195	0x0B0C	4	X	BLNIF, ITGIF, and ACOVIF	Stepper Stall Detect 2 (SSD2)
196	0x0B10	4	X	BLNIF, ITGIF, and ACOVIF	Stepper Stall Detect 3 (SSD3)
197	0x0B14	4	X	BLNIF, ITGIF, and ACOVIF	Stepper Stall Detect 4 (SSD4)
198	0x0B18	4	X	BLNIF, ITGIF, and ACOVIF	Stepper Stall Detect 5 (SSD5)
199	0x0B1C	4	X	EOF	Liquid Crystal Display 0 (LCD0)
200	0x0B20	4	O		Reserved
201	0x0B24	4	X	TFUF, RFOF, TBUF, RBOF, ABOF	QuadSPI 0
202	0x0B28	4	X	EOQF	QuadSPI 0
203	0x0B2C	4	X	TFFF, TBFF	QuadSPI 0
204	0x0B30	4	X	TCF, TFF	QuadSPI 0
205	0x0B34	4	X	RFDF, RBDF	QuadSPI 0
206	0x0B38	4	X	IPAEF, IPIEF, ICEF	QuadSPI 0

21.6.1 Interrupt Request Sources

The INTC has two types of interrupt requests, peripheral and software configurable. These interrupt requests can assert on any clock cycle.

21.6.1.1 Peripheral Interrupt Requests

An interrupt event in a peripheral's hardware sets a flag bit that resides in the peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC to the time that the INTC starts to drive the interrupt request to the processor is three clocks.

External interrupts are handled by the SIU (see Section 37.6.4, External interrupts).

21.6.1.2 Software configurable Interrupt Requests

An interrupt request is triggered by software by writing a 1 to a SET*x* bit in INTC_SSCIR0_3–INTC_SSCIR4_7. This write sets the corresponding flag bit, CLR*x*, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR*x* bit.

The time from the write to the SET*x* bit to the time that the INTC starts to drive the interrupt request to the processor is four clocks.

21.6.1.3 Unique Vector for Each Interrupt Request Source

Each peripheral and software configurable interrupt request is assigned a hardwired unique 9-bit vector. Software configurable interrupts 0–7 are assigned vectors 0–7 respectively. The peripheral interrupt requests are assigned vectors 8 to as high as needed to include all the peripheral interrupt requests. The peripheral interrupt request input ports at the boundary of the INTC block are assigned specific hardwired vectors within the INTC (see Table 21-1).

21.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*x* values set in INTC_PSR0_3–INTC_PSR204_206. The result is compared to PRI in the associated INTC_CPR. The results of those comparisons manage the priority of the ISR executed by the associated processor. The associated LIFO also assists in managing that priority.

21.6.2.1 Current Priority and Preemption

The priority arbitrator, selector, encoder, and comparator subblocks shown in Figure 21-1 compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software configurable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software settable interrupt request is generated for INTC interrupt acknowledge register (INTC_IACKR), and if in hardware vector mode, for the interrupt vector provided to the processor.

21.6.2.1.1 Priority Arbitrator Subblock

The priority arbitrator subblock for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software configurable. The output of the priority arbitrator subblock is the highest of those priorities assigned to a given processor. Also, any interrupt

requests which have this highest priority are output as asserted interrupt requests to the associated request selector subblock.

21.6.2.1.2 Request Selector Subblock

If only one interrupt request from the associated priority arbitrator subblock is asserted, then it is passed as asserted to the associated vector encoder subblock. If multiple interrupt requests from the associated priority arbitrator subblock are asserted, only the one with the lowest vector passes as asserted to the associated vector encoder subblock. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software configurable interrupt requests.

21.6.2.1.3 Vector Encoder Subblock

The vector encoder subblock generates the unique 9-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

21.6.2.1.4 Priority Comparator Subblock

The priority comparator submodule compares the highest priority output from the priority arbitrator submodule with PRI in INTC_CPR. If the priority comparator submodule detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in INTC_CPR or the PRI value in INTC_CPR is lowered below this highest priority. This highest priority then becomes the new priority which will be written to PRI in INTC_CPR when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in INTC_CPR.

21.6.2.2 Last-In First-Out (LIFO)

The LIFO stores the preempted PRI values from the INTC_CPR. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the INTC_CPR does not need to be loaded from the INTC_CPR and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the INTC_CPR.

The PRI value in the INTC_CPR is pushed onto the LIFO when the INTC_IACKR is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode. The priority is popped into PRI in the INTC_CPR whenever the INTC_EOIR is written.

Although the INTC supports 16 priorities, an ISR executing with PRI in the INTC_CPR equal to 15 will not be preempted. Therefore, the LIFO supports the stacking of 15 priorities. However, the LIFO is only 14 entries deep. An entry for a priority of 0 is not needed because of how pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 15 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop 0s if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

The LIFO is not memory-mapped.

21.6.3 Handshaking with processor

21.6.3.1 Software vector mode handshaking

This section describes handshaking in software vector mode.

21.6.3.1.1 Acknowledging interrupt request to processor

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode and the handshake near the end of the interrupt exception handler, is shown in Figure 21-10. The INTC examines the peripheral and software configurable interrupt requests. When it finds an asserted peripheral or software configurable interrupt request with a higher priority than PRI in the associated INTC_CPR, it asserts the interrupt request to the processor. The INTVEC field in the associated INTC_IACKR is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of handshaking process is described in Section 21.4.1.1, Software vector mode.

21.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC end-of-interrupt register (INTC_EOIR) must be written.When written, the associated LIFO is popped so the preempted priority is restored into PRI of the INTC_CPR. Before it is written, the peripheral or software configurable flag bit must be cleared so that the peripheral or software configurable interrupt request is negated.

NOTE

To ensure proper operation across all eSys MCUs, execute an MBAR or MSYNC instruction between the access to clear the flag bit and the write to the INTC_EOIR.

When returning from the preemption, the INTC does not search for the peripheral or software settable interrupt request whose ISR was preempted. Depending on how much the ISR progressed, that interrupt request may no longer even be asserted. When PRI in INTC_CPR is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it was to next execute before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

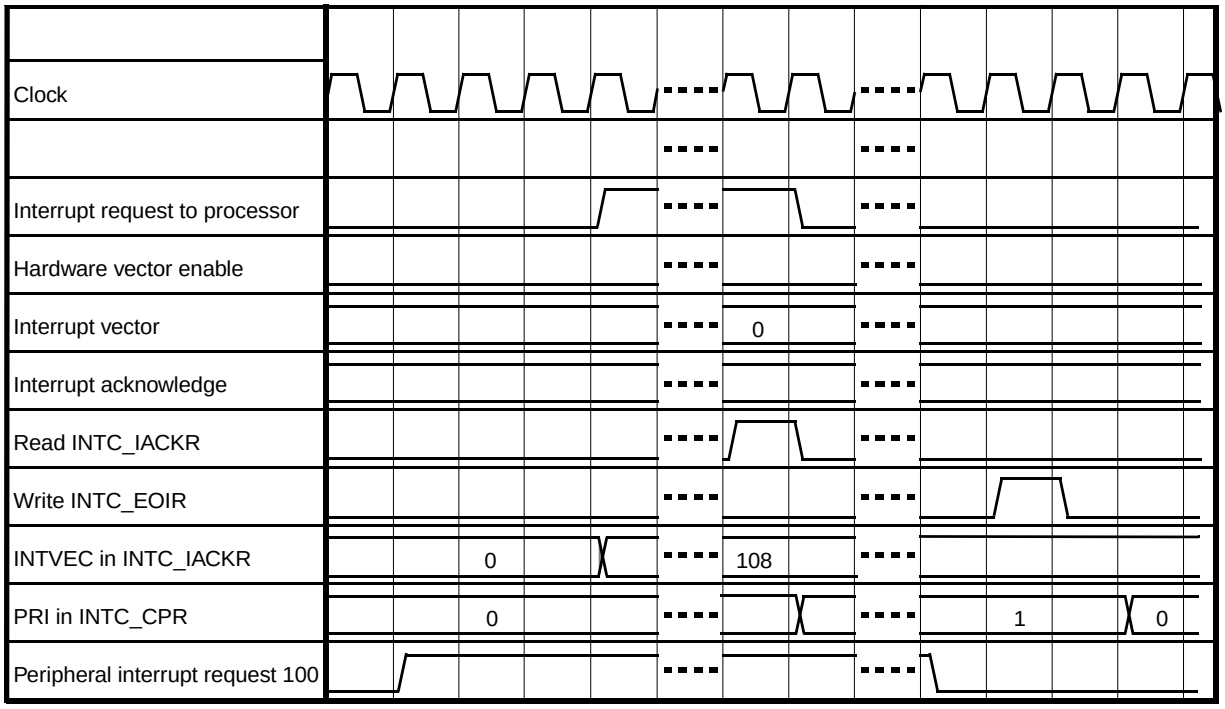


Figure 21-10. Software Vector mode handshaking timing diagram

21.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 21-11. As in software vector mode, the INTC examines the peripheral and software settable interrupt requests, and when it finds an asserted one with a higher priority than PRI in INTC_CPR, it asserts the interrupt request to the processor. The INTVEC field in the INTC_IACKR is updated with the preempting peripheral or software settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. In addition, the value of the interrupt vector to the processor matches the value of the INTVEC field in the INTC_IACKR. The rest of the handshaking is described in Section 21.4.1.2, Hardware vector mode.

The handshaking near the end of the interrupt exception handler, that is the writing to the INTC_EOIR, is the same as in software vector mode. Refer to Section 21.6.3.1.2, End of interrupt exception handler.

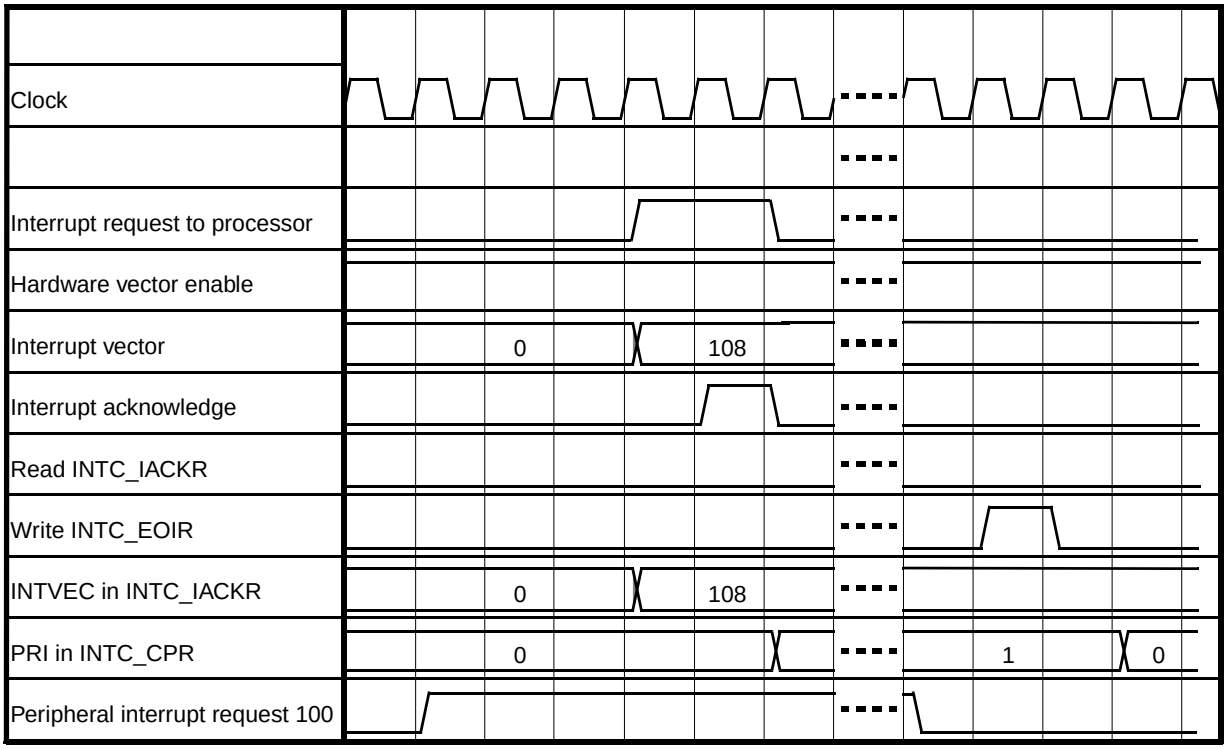


Figure 21-11. Hardware vector mode handshaking timing diagram

21.7 Initialization/application information

21.7.1 Initialization flow

After exiting reset, all of the PRI*n* fields in INTC priority select registers (INTC_PSR0_3–INTC_PSR_204_206) will be zero, and PRI in INTC current priority register (INTC_CPR) will be 15. These reset values will prevent the INTC from asserting the interrupt request to the processor. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software settable interrupt requests to cause an interrupt request to the processor is: interrupt_request_initialization:

```
interrupt_request_initialization:
configure VTES and HVEN in INTC_MCR
configure VTBA in INTC_IACKR
raise the PRIn fields in INTC_PSRn
set the enable bits or clear the mask bits for the peripheral interrupt requests
lower PRI in INTC_CPR to zero
enable processor recognition of interrupts
```

21.7.2 Interrupt Exception Handler

These example interrupt exception handlers use Power Architecture assembly code.

21.7.2.1 Software vector mode

```
interrupt_exception_handler:
code to create stack frame, save working register, and save SRR0 and SRR1
lis      r3,INTC_IACKR@ha           # form adjusted upper half of INTC_IACKR address
lwz      r3,INTC_IACKR@l(r3)        # load INTC_IACKR, which clears request to processor
lwz      r3,0x0(r3)                 # load address of ISR from vector table
wrteei 1                            # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

mtlr     r3                         # move INTC_IACKR contents into link register
blrl                                # branch to ISR; link register updated with epilog
                                    # address

epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                                # ensure store to clear flag bit has completed
lis      r3,INTC_EOIR@ha            # form adjusted upper half of INTC_EOIR address
li       r4,0x0                     # form 0 to write to INTC_EOIR
wrteei   0                          # disable processor recognition of interrupts
stw      r4,INTC_EOIR@l(r3)         # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi


vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
.
.
.
address of ISR for interrupt with vector 510
address of ISR for interrupt with vector 511

ISRx:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr # return to epilog
```

21.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations which support a hardware vector. This example assumes that each interrupt_exception_handler*x* only has space for four instructions, and therefore a branch to interrupt_exception_handler_continued*x* is needed.

```
interrupt_exception_handlerx:
b interrupt_exception_handler_continuedx# 4 instructions available, branch to continue
```

```
interrupt_exception_handler_continuedx:
code to create stack frame, save working register, and save SRR0 and SRR1

wrteei   1                # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

bl       ISRx             # branch to ISR for interrupt with vector x


epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                      # ensure store to clear flag bit has completed
lis      r3,INTC_EOIR@ha  # form adjusted upper half of INTC_EOIR address
li       r4,0x0           # form 0 to write to INTC_EOIR
wrteei   0                # disable processor recognition of interrupts
stw      r4,INTC_EOIR@l(r3) # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi

ISRx:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC
blr                       # branch to epilog
```

21.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC current priority register (INTC_CPR) having a value of 0. The RTOS will execute the tasks according to whatever priority scheme that it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR priority 0.

If a task shares a resource with an ISR and the PCP is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR while the shared resource is being accessed.

An ISR whose PRI*n* in INTC priority select registers (INTC_PSR0_3–INTC_PSR204_206) has a value of 0 will not cause an interrupt request to the processor, even if its peripheral or software settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

21.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software configurable interrupt requests. However, if multiple peripheral or software configurable interrupt requests are asserted, more than one has the highest priority, and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software configurable interrupt requests asserted.

The example in Table 21-11 shows the order of execution of both ISRs with different priorities and the same priority

Table 21-11. Order of ISR execution example

Step	Step description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108[1]	ISR208	ISR308	ISR408	Interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 is asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 is asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 is asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR.						X	0
12	RTOS continues execution.	X						0

[1] ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software configurable interrupt requests.

21.7.5 Priority Ceiling Protocol

21.7.5.1 Elevating Priority

The PRI field in INTC_CPR is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR to 3, the ceiling of all of the ISR priorities. After they release the resource, the PRI value in INTC_CPR can be lowered. If they do not raise their priority, ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

21.7.5.2 Ensuring Coherency

A scenario can cause non-coherent accesses to the shared resource. For example, ISR1 and ISR2 are both running on the same core and both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing and writes to the INTC_CPR. The instruction following this store is a store to a value in a shared coherent data block. Either immediately before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible that both stores will be executed. ISR2 thereby thinks that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent corruption of a coherent data block, modifications to PRI in INTC_CPR can be made by those system services with the code sequence:

```
disable processor recognition of interrupts
PRI modification
enable processor recognition of interrupts
```

21.7.6 Selecting Priorities According to Request Rates and Deadlines

The selection of the priorities for the ISRs can be made using rate monotonic scheduling (RMS) or a superset of it, deadline monotonic scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is

assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3; however, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC has 16 priorities, which may be less than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, etc. With this approach, a range of ISR request rates of 2^{16} could be included, regardless of the number of ISRs.

Reducing the number of priorities reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They do not need to use the PCP to access the shared resource.

21.7.7 Software configurable interrupt requests

The software configurable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and they may also be used by processors to interrupt other processors in a multiple processor system.

21.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI*x* value in INTC_PSR0_3 -INTC_PSR204_206, which becomes the PRI value in INTC_CPR with the interrupt acknowledge. The ISR, however, can have a portion that does not need to be executed at this higher priority. Therefore, executing the later portion that does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than what the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET*x* bit in INTC_SSCIR0_3–INTC_SSCIR4_7. Writing a 1 to SET*x* causes a software configurable interrupt request. This software configurable interrupt request will usually have a lower PRI*x* value in the INTC_PSR*x*_*x* and will not cause preemptive scheduling inefficiencies. After generating a software settable interrupt request, the higher priority ISR completes. The lower priority ISR is scheduled according to its priority. Execution of the higher priority ISR is not resumed after the completion of the lower priority ISR.

21.7.7.2 Scheduling an ISR on another processor

Because the SET*x* bits in the INTC_SSCIR*x_x* are memory-mapped, processors in multiple-processor systems can schedule ISRs on the other processors. One application is that one processor wants to command another processor to perform a piece of work and the initiating processor does not need to use the results of that work. If the initiating processor is concerned that the processor executing the software configurable ISR has not completed the work before asking it to again execute the ISR, it can check if the corresponding CLR*x* bit in INTC_SSCIR*x_x* is asserted before again writing a 1 to the SET*x* bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. To do this, the first processor writes a 1 to a SET*x* bit on the second processor. After accessing the block of data, the second processor clears the corresponding CLR*x* bit and then writes 1 to a SET*x* bit on the first processor, informing it that it can now access the block of data.

21.7.8 Lowering priority within an ISR

A common method for avoiding preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (see Section 21.7.7.1, Scheduling a lower priority portion of an ISR) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

NOTE

Lowering the PRI value in INTC_CPR within an ISR to below the ISR's corresponding PRI value in INTC_PSR0_3–INTC_PSR292_293 allows more preemptions than the LIFO depth can support.

Therefore, the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

21.7.9 Negating an Interrupt Request Outside of its ISR

21.7.9.1 Negating an Interrupt Request as a Side Effect of an ISR

Some peripherals have flag bits that can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits and their corresponding interrupt requests. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

21.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own. One reason that an ISR clears multiple flag bits is because it serviced those flag bits, and therefore the ISRs for these flag bits do not need to be executed.

21.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside its own ISR due to the side effect of an ISR execution or the intentional clearing a flag bit, the priorities of the peripheral or software configurable interrupt requests for these other flag bits must be selected properly. Their PRI*x* values in INTC_PSR0_3–INTC_PSR204_206 must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR0_3–INTC_SSCIR4_7 as the clearing of the flag bit that caused the present ISR to be executed (see Section 21.6.3.1.2, End of interrupt exception handler).

A flag bit whose enable bit or mask bit negates its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI*x* value in INTC_PSR*x*_*x*.

21.7.10 Examining LIFO contents

In normal mode, the user does not need to know the contents of the LIFO. He may not even know how deeply the LIFO is nested. However, if he wants to read the contents, such as in debug mode, they are not memory-mapped. The contents can be read by popping the LIFO and reading the PRI field in either INTC_CPR. The code sequence is:

```
pop_lifo:
store to INTC_EOIR
load INTC_CPR, examine PRI, and store onto stack
if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
load stacked PRI value and store to INTC_CPR
load INTC_IACKR
if stacked PRI values are not depleted, branch to push_lifo
```

Chapter 22
LCD Driver (LCD64F6B)

22.1 Information specific to this device

This section presents device-specific parameterization and customization information not specifically referenced in the remainder of this chapter.

22.1.1 Number of front and back planes

Table 22-1. Number of front and back planes

Parameter	Value[1]
Number of front planes	38 or 40
Number of back planes	4 or 6

1 Software-configurable

22.1.2 LCD clock selection

Table 22-2 shows the clocks selected by the LCDCR[LCDOCS] bit.

Table 22-2. LCD clock selection based on LCDCR[LCDOCS]

Value of LCDCR[LCDOCS]	Clock selected
1	32 KHz OSC
0	128 kHz OSC

22.1.3 Settings during Standby mode

To keep the LCD driver shut down in Standby mode, the following settings are needed:

- LCDCR[LCDRST] = 0
- LCDCR[LCDRCS] = 0

To keep the LCD driver on (functioning) in Standby mode, the following settings are needed:

- LCDCR[LCDRST] = 1
- LCDCR[LCDRCS] = 1
- LCDCR[LCDOCS] = 0 or 1
  - — If this field is 0, the LCD driver operates from SIRC.
  - — If this field is 1, the LCD driver operates from SXOSC.

22.2 Introduction

22.2.1 Overview

The LCD driver module has up to 64 frontplane drivers (*n*) and up to 6 backplane drivers (*m*) so that a maximum of 384 LCD segments are controllable. The actual implementation (*n*,*m*) depends on the device specification. Each segment is controlled by a corresponding bit in the LCD RAM. *m* multiplex modes (1/1, 1/2,...1/*m* duty), and three bias (1/1, 1/2, 1/3) methods are available. The V_0 voltage is the lowest level of the output waveform and V_3 becomes the highest level. All frontplane and backplane pins can be multiplexed with other Port functions.

The LCD driver system consists of five major submodules:

- Timing and control – consists of registers and control logic for frame clock generation, bias voltage level select, frame duty select, contrast adjustment, backplane select and frontplane select/enable, remapping of backplane drivers to produce the required frame frequency and voltage waveforms.
- LCD RAM – contains the data to be displayed on the LCD. Data can be read from or written to the display RAM at any time.
- Frontplane drivers – consists of *n* frontplane drivers.
- Backplane drivers – consists of *m* backplane drivers.
- Voltage generator – Based on reference voltage, i.e. applied to VLCD, it generates the voltage levels for the timing and control logic to produce the frontplane and backplane waveforms.

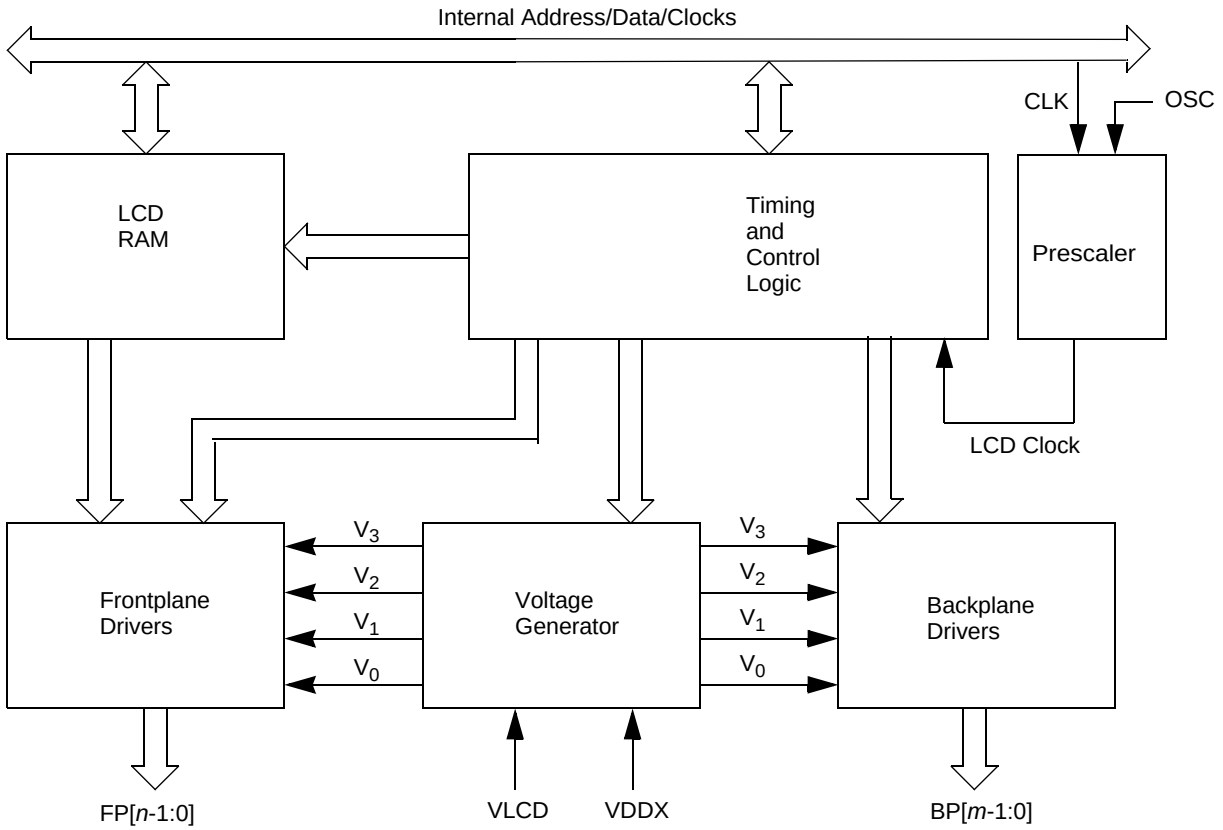


Figure 22-1. Block diagram

22.2.2 Features

The LCD64F6B includes these features:

- As many as 64 frontplane drivers
  - Each frontplane has an individual enable bit
- As many as six backplane drivers
- Remapping of backplane drivers
- Programmable frame clock generator
- Programmable bias voltage level selector
- Programmable output current
- Selectable output current boost during transitions.
- Selectable LCD frame frequency interrupt event
- On-chip generation of four different output voltage levels
- Two contrast adjustment options:
  - Using VLCD voltage (if available at pin)
  - Using contrast adjustment phases

- Selectable continuous drive of LCD while in power down mode

22.2.3 Modes of operation

The LCD64F6B module supports up to seven operation modes with different numbers of backplanes and different biasing levels.During power saving mode the LCD operation can be suspended under software control. Depending on the state of internal bits, the LCD can operate normally with source clock applied, or the LCD clock generation can be turned off and the LCD64F6B module enters a power conservation state.

This is a high-level description only, detailed descriptions of operating modes are contained in later sections.

22.3 External signal description

Table 22-3. Signal properties

Name	Port	Function	Reset	Pullup
m Backplane Waveforms	BP[*m* – 1:0]	Backplane waveform signals that connect directly to the pads	high impedance	
n Frontplane Waveforms	FP[*n* – 1:0]	Frontplane waveform signals that connect directly to the pads	high impedance	
LCD Voltage	VLCD	LCD reference supply voltage		
VDDX Voltage	VDDX	LCD supply voltage		
VSSX Voltage	VSSX	LCD ground voltage		

22.3.1 Detailed signal descriptions

Table 22-4. Interface A—detailed signal descriptions

Signal	Description
BP[*m* – 1:0]	This output signal vector represents the analog backplane waveforms of the LCD64F6B module and is connected directly to the corresponding pads.
FP[*n* – 1:0]	This output signal vector represents the analog frontplane waveforms of the LCD64F6B module and is connected directly to the corresponding pads.
VLCD	Positive reference supply voltage for the LCD waveform generation.
VDDX	Positive supply voltage for the LCD waveform generation.
VSSX	Ground supply voltage for the LCD waveform generation.

22.4 Memory map and register definition

22.4.1 Memory map

Table 22-5. Block memory map

Offset or Address	Register	Access	Reset Value	Location
General Registers				
0x00	LCD Control Register (LCDCR)	R/W	0x0000_0000	on page 790
0x04	LCD Prescaler Control Register (LCDPCR)	R/W	0x0000_0000	on page 793
0x08	LCD Contrast Control Register (LCDCCR)	R/W	0x0000_0000	on page 794
0x0C	Reserved			
0x10	LCD Frontplane Enable Register 0 (FPENR0)	R/W	0x0000_0000	on page 795
0x14	LCD Frontplane Enable Register 1 (FPENR1)	R/W	0x0000_0000	on page 796
0x18	Reserved			
0x1C	Reserved			
0x20	LCDRAM (Location 0)	R/W	0x0000_0000	on page 797
0x24	LCDRAM (Location 1)	R/W	0x0000_0000	on page 798
0x28	LCDRAM (Location 2)	R/W	0x0000_0000	on page 799
0x2C	LCDRAM (Location 3)	R/W	0x0000_0000	on page 800
0x30	LCDRAM (Location 4)	R/W	0x0000_0000	on page 801
0x34	LCDRAM (Location 5)	R/W	0x0000_0000	on page 802
0x38	LCDRAM (Location 6)	R/W	0x0000_0000	on page 803
0x3C	LCDRAM (Location 7)	R/W	0x0000_0000	on page 804
0x40	LCDRAM (Location 8)[1]	R/W	0x0000_0000	on page 805
0x44	LCDRAM (Location 9)[2]	R/W	0x0000_0000	on page 806
0x48	LCDRAM (Location 10)[3]	R/W	0x0000_0000	on page 807
0x4C	LCDRAM (Location 11)[4]	R/W	0x0000_0000	on page 808
0x50	LCDRAM (Location 12)[5]	R/W	0x0000_0000	on page 809
0x54	LCDRAM (Location 13)[6]	R/W	0x0000_0000	on page 810
0x58	LCDRAM (Location 14)[7]	R/W	0x0000_0000	on page 811
0x5C	LCDRAM (Location 15)[8]	R/W	0x0000_0000	on page 812

1 End of implemented RAM for 36 FPs
2 End of implemented RAM for 40 FPs
3 End of implemented RAM for 44 FPs
4 End of implemented RAM for 48 FPs
5 End of implemented RAM for 52 FPs

6 End of implemented RAM for 56 FPs
7 End of implemented RAM for 60 FPs
8 End of implemented RAM for 64 FPs

22.4.2 Register descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number.

22.4.2.1 LCD Control Register (LCDCR)

Address: Base + 0x00 Access: User read/write

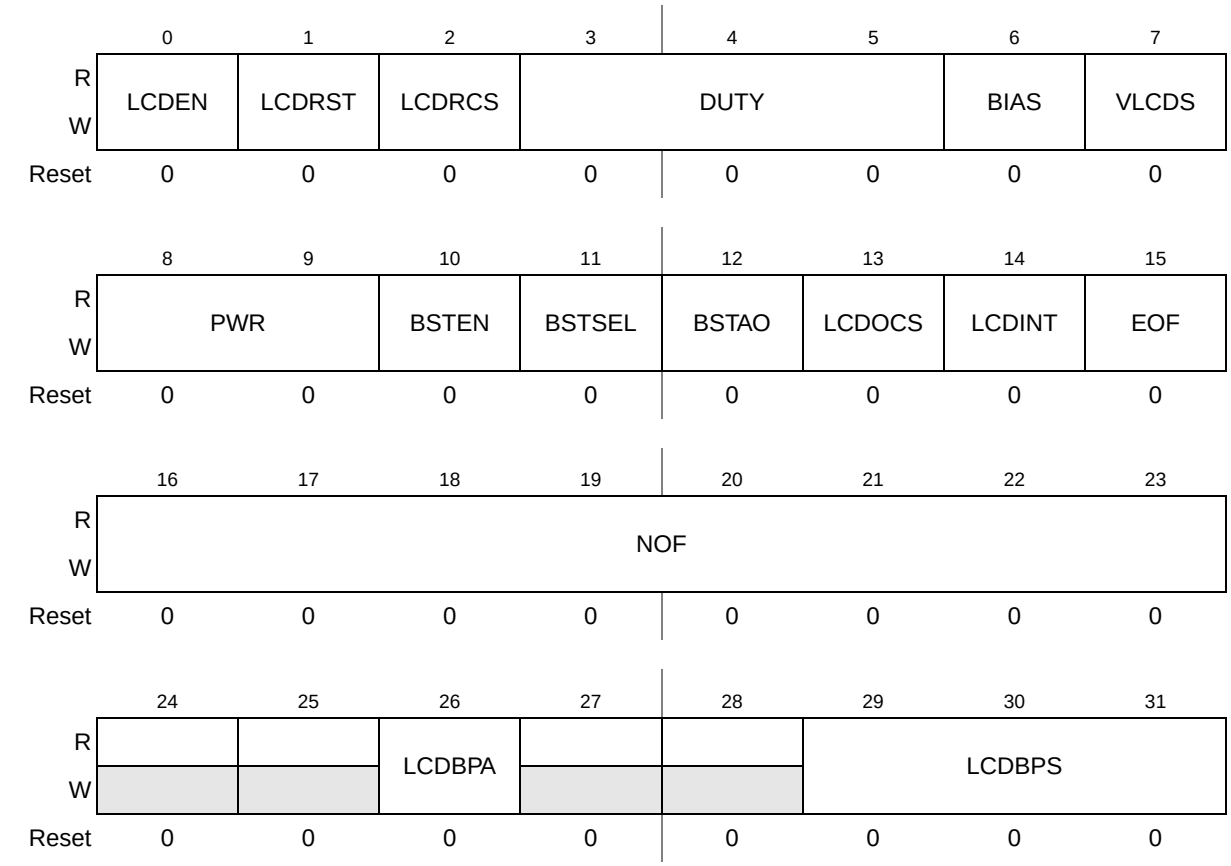


Figure 22-2. LCD Control Register (LCDCR)

Table 22-6. LCDCR field descriptions

Field	Description
0	LCDEN. LCD Driver System Enable. 0 All frontplane and backplane pins are disabled. In addition, the LCD Driver is disabled and all LCD waveform generation clocks are stopped. 1 LCD Driver System is enabled. All FP[*n*-1:0] pins with FP[*n*-1]EN set, will output an LCD driver waveform. The BP[*m*-1:0] pins will output an LCD Driver waveform based on the settings of DUTY.

Table 22-6. LCDCR field descriptions (continued)

Field	Description
1	LCDRST. Continue to drive LCD display while stop/standby requested. Can be written only when LCDEN is cleared. See Section 22.5.8, Operation in power saving modes, for details. 0 Stop LCD64F6B display driver system while stop/standby requested. 1 LCD display driver system operates normally while stop/standby requested.
2	LCDRCS. LCD Reference Clock Select. Can be written only when LCDEN is cleared. 0 Source clock for LCD Driver system (prescaler input) is system clock. 1 Source clock for LCD Driver system (prescaler input) is OSC clock. See device level documentation for detail on OSC clk.
3:5	DUTY. LCD Duty Select The DUTY bits select the duty (multiplex mode) of the LCD Driver system, as shown in Table 22-29. The multiplex mode should be changed only when LCD Driver is disabled, LCDEN is not set.
6	BIAS. BIAS Voltage Select This bit selects the bias voltage levels during various LCD operating modes, as shown in Table 22-29
7	VLCDS. LCD voltage reference select. The VCLDS bit selects which supply is be used as LCD waveform reference supply. Can be written only when LCDEN is cleared. 0 VDDX is used as reference supply. 1 VLCD is used as reference supply.
8:9	PWR. LCD Power mode The PWR bits select the output current and have a direct impact on the power consumption and drive capability of the LCD64F6B module. Table 22-30 lists the possible selections.
10	BSTEN. LCD output current boost enable Since the LCD appears like a capacitance to the driver it is sometimes useful to boost the output current during transitions to increase the slew rate of the driver. 0 No output current boost available. 1 Output current boost during transitions according to the BSTSEL bit.
11	BSTSEL. LCD output current boost select The BST bit sets the multiplier for the output current boost. If the BSTEN bit is set, the output current is boosted during transitions in the following way: 0 8 times boosting. 1 16 times boosting.

Table 22-6. LCDCR field descriptions (continued)

Field	Description
12	BSTAO. LCD Boost always on If set, the selected by BST and enabled by BSTEN boost current is always on and not only during the time frame generated by the state machine. 0 The Boost always on feature is disabled. 1 The Boost always on feature is enabled.
13	LCDOCS. LCD OSC Clock Select. Can be written only when LCDEN is cleared. See Device Level information which OSC clock is defined as first and second OSC clock. 0 First OSC clock is selected as clock for LCD Driver system (prescaler input) if LCDRCS is set. 1 Second OSC clock is selected as clock for LCD Driver system (prescaler input) if LCDRCS is set.
14	LCDINT. LCD interrupt enable. 0 Interrupt request is disabled. 1 Interrupt will be requested whenever EOF is set.
15	EOF. End of frame interrupt flag. End of frame interrupt flag bit is set every time when NOF frames have been executed while LCDEN is set. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LCDINT = 1), EOF causes an interrupt request. 0 Defined via NOF bits frames have not been executed yet. 1 Defined via NOF bits frames have been executed.
16:23	NOF: Number of frames. The number of frames bits determine how many frames are counted until an interrupt will be requested. The possible number of frames are: 0x00 An interrupt is requested at the end of every frame. 0x01 An interrupt is requested at the end of every second frame. . . 0xFF An interrupt is requested at the end of every 256th frame.
26	LCDBPA: Backplane Adding If set, the backplanes, which are not available in standard configuration, will be mapped on frontplanes as shown in Table 22-28. Hence, up to six backplanes can be used to drive LCD displays. 0 No additional backplanes will be mapped on frontplanes. 1 Additional backplanes will be mapped on frontplanes.
29:31	LCDBPS: Backplane Shifting Using these bits, backplanes will be swapped with frontplanes as shown in Table 22-28.

22.4.2.2 LCD Prescaler Control Register (LCDPCR)

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	LCLK			
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	C	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-3. LCD Prescaler Control Register (LCDPCR)

Table 22-7. LCDPCR field descriptions

Field	Description
4:7 LCLK	LCLK. LCD Clock Prescaler. The LCD Clock Prescaler bits determine the clock divider value to produce the LCD Clock Frequency. For detailed description of the correlation between LCD Clock Prescaler bits and the divider value please refer to Table 22-27. LCD Clock Prescaler bits should be changed only when LCD Driver is disabled, LCDEN is not set.

22.4.2.3 LCD Contrast Control Register (LCDCCR)

Address: Base + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7
R	CCEN	0	0	0	0	LCC		
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	LCC							
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-4. LCD Contrast Control Register (LCDCCR)

Table 22-8. LCDCCR field descriptions

Field	Description
0 CCEN	CCEN. LCD Contrast Control Enable. 0 The Contrast Control is disabled. 1 The Contrast Control is enabled. The length of the contrast phase depends on the bits LCC.
5:15 LCC	LCC. LCD Contrast Control. The Contrast Control bits determine the width of the contrast phase. 0x000 Contrast Phase has no duration what results in highest contrast. 0x7FF Contrast Phase lasts the whole duty cycle.

22.4.2.4 LCD Frontplane Enable Register 0 (FPENR0)

Address: Base + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7
R W	FP31EN	FP30EN	FP29EN	FP28EN	FP27EN	FP26EN	FP25EN	FP24EN
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R W	FP23EN	FP22EN	FP121EN	FP20EN	FP19EN	FP18EN	FP17EN	FP16EN
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R W	FP15EN	FP14EN	FP13EN	FP12EN	FP11EN	FP10EN	FP9EN	FP8EN
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R W	FP7EN	FP6EN	FP5EN	FP4EN	FP3EN	FP2EN	FP1EN	FP0EN
Reset	0	0	0	0	0	0	0	0

Figure 22-5. LCD Frontplane Enable Register 0 (FPENR0)

Table 22-9. FPENR0 field descriptions

Field	Description
0:31	FP[31:0]EN. Frontplane Output Enable. The FP[31:0]EN bits enable the frontplane driver outputs. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set FP[31:0]EN bits before LCDEN is set. 0 Frontplane driver output disabled on FP[31:0]. 1 Frontplane driver output enabled on FP[31:0].

22.4.2.5 LCD Frontplane Enable Register 1 (FPENR1)

Address: Base + 0x14 Access: User read/write

	0	1	2	3	4	5	6	7
R W	FP63EN	FP62EN	FP61EN	FP60EN	FP59EN	FP58EN	FP57EN	FP56EN
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R W	FP55EN	FP54EN	FP53EN	FP52EN	FP51EN	FP50EN	FP49EN	FP48EN
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R W	FP47EN	FP46EN	FP45EN	FP44EN	FP43EN	FP42EN	FP41EN	FP40EN
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R W	FP39EN	FP38EN	FP37EN	FP36EN	FP35EN	FP34EN	FP33EN	FP32EN
Reset	0	0	0	0	0	0	0	0

Figure 22-6. LCD Frontplane Enable Register 1 (FPENR1)

Table 22-10. FPENR1 field descriptions

Field	Description
0:31	FP[63:32]EN. Frontplane Output Enable. The FP[63:32]EN bits enable the frontplane driver outputs. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set FP[63:32]EN bits before LCDEN is set. 0 Frontplane driver output disabled on FP[63:32]. 1 Frontplane driver output enabled on FP[63:32]. **Note:** The implemented FP[*n*-1]EN bits depend on the number of implemented frontplanes.

22.4.2.6 LCDRAM (Location 0)

Address: Base + 0x20 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP0BP5	FP0BP4	FP0BP3	FP0BP2	FP0BP1	FP0BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP1BP5	FP1BP4	FP1BP3	FP1BP2	FP1BP1	FP1BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP2BP5	FP2BP4	FP2BP3	FP2BP2	FP2BP1	FP2BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP3BP5	FP3BP4	FP3BP3	FP3BP2	FP3BP1	FP3BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-7. LCDRAM (Location 0)

Table 22-11. LCDRAM (Location 0) field descriptions

Field	Description
0:31	FP[0:3]BP[5:0]. LCD segment ON. The FP[0:3]BP[5:0] bit displays (turns on) the LCD segment connected between FP[0:3] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.7 LCDRAM (Location 1)

Address: Base + 0x24 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP4BP5	FP4BP4	FP4BP3	FP4BP2	FP4BP1	FP4BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP5BP5	FP5BP4	FP5BP3	FP5BP2	FP5BP1	FP5BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP6BP5	FP6BP4	FP6BP3	FP6BP2	FP6BP1	FP6BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP7BP5	FP7BP4	FP7BP3	FP7BP2	FP7BP1	FP7BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-8. LCDRAM (Location 1)

Table 22-12. LCDRAM (Location 1) field descriptions

Field	Description
0:31	FP[4:7]BP[5:0]. LCD segment ON. The FP[4:7]BP[5:0] bit displays (turns on) the LCD segment connected between FP[4:7] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.8 LCDRAM (Location 2)

Address: Base + 0x28 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP8BP5	FP8BP4	FP8BP3	FP8BP2	FP8BP1	FP8BP0-
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP9BP5	FP9BP4	FP9BP3	FP9BP2	FP9BP1	FP9BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP10BP5	FP10BP4	FP10BP3	FP10BP2	FP10BP1	FP10BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP11BP5	FP11BP4	FP11BP3	FP11BP2	FP11BP1	FP11BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-9. LCDRAM (Location 2)

Table 22-13. LCDRAM (Location 2) field descriptions

Field	Description
0:31	FP[8:11]BP[5:0]. LCD segment ON. The FP[8:11]BP[5:0] bit displays (turns on) the LCD segment connected between FP[8:11] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.9 LCDRAM (Location 3)

Address: Base + 0x2C

Access: User read/write

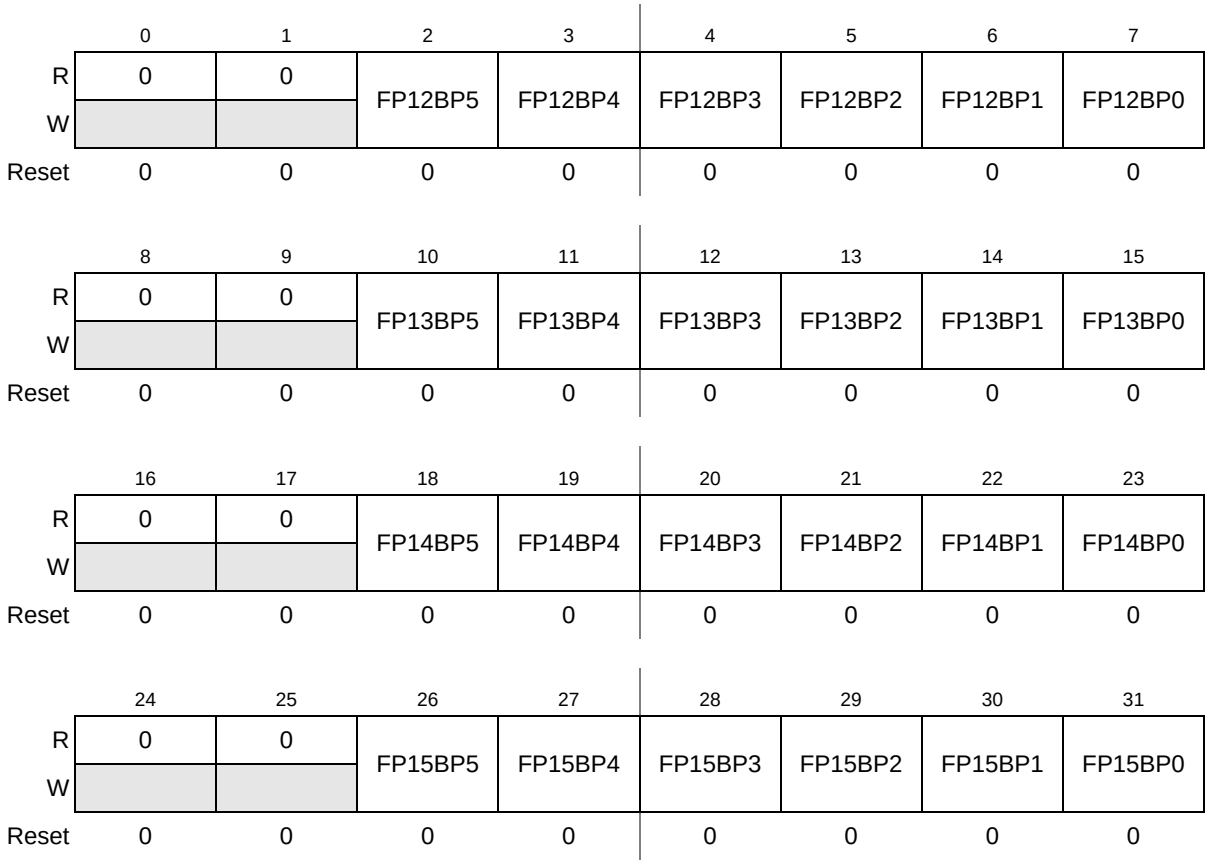


Figure 22-10. LCDRAM (Location 3)

Table 22-14. LCDRAM (Location 3) field descriptions

Field	Description
0:31	FP[12:15]BP[5:0]. LCD segment ON. The FP[12:15]BP[5:0] bit displays (turns on) the LCD segment connected between FP[12:15] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.10 LCDRAM (Location 4)

Address: Base + 0x30 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP16BP5	FP16BP4	FP16BP3	FP16BP2	FP16BP1	FP16BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP17BP5	FP17BP4	FP17BP3	FP17BP2	FP17BP1	FP17BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP18BP5	FP18BP4	FP18BP3	FP18BP2	FP18BP1	FP18BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP19BP5	FP19BP4	FP19BP3	FP19BP2	FP19BP1	FP19BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-11. LCDRAM (Location 4)

Table 22-15. LCDRAM (Location 4) field descriptions

Field	Description
0:31	FP[16:19]BP[5:0]. LCD segment ON. The FP[16:19]BP[5:0] bit displays (turns on) the LCD segment connected between FP[16:19] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.11 LCDRAM (Location 5)

Address: Base + 0x34 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP20BP5	FP20BP4	FP20BP3	FP20BP2	FP20BP1	FP20BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP21BP5	FP21BP4	FP21BP3	FP21BP2	FP21BP1	FP21BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP22BP5	FP22BP4	FP22BP3	FP22BP2	FP22BP1	FP22BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP23BP5	FP23BP4	FP23BP3	FP23BP2	FP23BP1	FP23BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-12. LCDRAM (Location 5)

Table 22-16. LCDRAM (Location 5) field descriptions

Field	Description
0:31	FP[20:23]BP[5:0]. LCD segment ON. The FP[20:23]BP[5:0] bit displays (turns on) the LCD segment connected between FP[20:23] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.12 LCDRAM (Location 6)

Address: Base + 0x38 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP24BP5	FP24BP4	FP24BP3	FP24BP2	FP24BP1	FP24BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP25BP5	FP25BP4	FP25BP3	FP25BP2	FP25BP1	FP25BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP26BP5	FP26BP4	FP26BP3	FP26BP2	FP26BP1	FP26BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP27BP5	FP27BP4	FP27BP3	FP27BP2	FP27BP1	FP27BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-13. LCDRAM (Location 6)

Table 22-17. LCDRAM (Location 6) field descriptions

Field	Description
0:31	FP[24:27]BP[5:0]. LCD segment ON. The FP[24:27]BP[5:0] bit displays (turns on) the LCD segment connected between FP[24:27] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.13 LCDRAM (Location 7)

Address: Base + 0x3C

Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP28BP5	FP28BP4	FP28BP3	FP28BP2	FP28BP1	FP28BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP29BP5	FP29BP4	FP29BP3	FP29BP2	FP29BP1	FP29BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP30BP5	FP30BP4	FP30BP3	FP30BP2	FP30BP1	FP30BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP31BP5	FP31BP4	FP31BP3	FP31BP2	FP31BP1	FP31BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-14. LCDRAM (Location 7)

Table 22-18. LCDRAM (Location 7) field descriptions

Field	Description
0:31	FP[28:31]BP[5:0]. LCD segment ON. The FP[28:31]BP[5:0] bit displays (turns on) the LCD segment connected between FP[28:31] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.14 LCDRAM (Location 8)

Address: Base + 0x40 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP32BP5	FP32BP4	FP32BP3	FP32BP2	FP32BP1	FP32BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP33BP5	FP33BP4	FP33BP3	FP33BP2	FP33BP1	FP33BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP34BP5	FP34BP4	FP34BP3	FP34BP2	FP34BP1	FP34BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP35BP5	FP35BP4	FP35BP3	FP35BP2	FP35BP1	FP35BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-15. LCDRAM (Location 8)

Table 22-19. LCDRAM (Location 8) field descriptions

Field	Description
0:31	FP[32:35]BP[5:0]. LCD segment ON. The FP[32:35]BP[5:0] bit displays (turns on) the LCD segment connected between FP[32:35] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.15 LCDRAM (Location 9)

Address: Base + 0x44 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP36BP5	FP36BP4	FP36BP3	FP36BP2	FP36BP1	FP36BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP37BP5	FP37BP4	FP37BP3	FP37BP2	FP37BP1	FP37BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP38BP5	FP38BP4	FP38BP3	FP38BP2	FP38BP1	FP38BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP39BP5	FP39BP4	FP39BP3	FP39BP2	FP39BP1	FP39BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-16. LCDRAM (Location 9)

Table 22-20. LCDRAM (Location 9) field descriptions

Field	Description
0:31	FP[36:39]BP[5:0]. LCD segment ON. The FP[36:39]BP[5:0] bit displays (turns on) the LCD segment connected between FP[36:39] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.16 LCDRAM (Location 10)

Address: Base + 0x48 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP40BP5	FP40BP4	FP40BP3	FP40BP2	FP40BP1	FP40BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP41BP5	FP41BP4	FP41BP3	FP41BP2	FP41BP1	FP41BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP42BP5	FP42BP4	FP42BP3	FP42BP2	FP42BP1	FP42BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP43BP5	FP43BP4	FP43BP3	FP43BP2	FP43BP1	FP43BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-17. LCDRAM (Location 10)

Table 22-21. LCDRAM (Location 10) field descriptions

Field	Description
0:31	FP[40:43]BP[5:0]. LCD segment ON. The FP[40:43]BP[5:0] bit displays (turns on) the LCD segment connected between FP[40:43] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.17 LCDRAM (Location 11)

Address: Base + 0x4C

Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP44BP5	FP44BP4	FP44BP3	FP44BP2	FP44BP1	FP44BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP45BP5	FP45BP4	FP45BP3	FP45BP2	FP45BP1	FP45BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP46BP5	FP46BP4	FP46BP3	FP46BP2	FP46BP1	FP46BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP47BP5	FP47BP4	FP47BP3	FP47BP2	FP47BP1	FP47BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-18. LCDRAM (Location 11)

Table 22-22. LCDRAM (Location 11) field descriptions

Field	Description
0:31	FP[44:47]BP[5:0]. LCD segment ON. The FP[44:47]BP[5:0] bit displays (turns on) the LCD segment connected between FP[44:47] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.18 LCDRAM (Location 12)

Address: Base + 0x50 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP48BP5	FP48BP4	FP48BP3	FP48BP2	FP48BP1	FP48BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP49BP5	FP49BP4	FP49BP3	FP49BP2	FP49BP1	FP49BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP50BP5	FP50BP4	FP50BP3	FP50BP2	FP50BP1	FP50BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP51BP5	FP51BP4	FP51BP3	FP51BP2	FP51BP1	FP51BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-19. LCDRAM (Location 12)

Table 22-23. LCDRAM (Location 12) field descriptions

Field	Description
0:31	FP[48:51]BP[5:0]. LCD segment ON. The FP[48:51]BP[5:0] bit displays (turns on) the LCD segment connected between FP[48:51] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.19 LCDRAM (Location 13)

Address: Base + 0x54 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP52BP5	FP52BP4	FP52BP3	FP52BP2	FP52BP1	FP52BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP53BP5	FP53BP4	FP53BP3	FP53BP2	FP53BP1	FP53BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP54BP5	FP54BP4	FP54BP3	FP54BP2	FP54BP1	FP54BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP55BP5	FP55BP4	FP55BP3	FP55BP2	FP55BP1	FP55BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-20. LCDRAM (Location 13)

Table 22-24. LCDRAM (Location 13) field descriptions

Field	Description
0:31	FP[52:55]BP[5:0]. LCD segment ON. The FP[52:55]BP[5:0] bit displays (turns on) the LCD segment connected between FP[52:55] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.20 LCDRAM (Location 14)

Address: Base + 0x58 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP56BP5	FP56BP4	FP56BP3	FP56BP2	FP56BP1	FP56BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP57BP5	FP57BP4	FP57BP3	FP57BP2	FP57BP1	FP57BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP58BP5	FP58BP4	FP58BP3	FP58BP2	FP58BP1	FP58BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP59BP5	FP59BP4	FP59BP3	FP59BP2	FP59BP1	FP59BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-21. LCDRAM (Location 14)

Table 22-25. LCDRAM (Location 14) field descriptions

Field	Description
0:31	FP[56:59]BP[5:0]. LCD segment ON. The FP[56:59]BP[5:0] bit displays (turns on) the LCD segment connected between FP[56:59] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON

22.4.2.21 LCDRAM (Location 15)

Address: Base + 0x5C Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	FP60BP5	FP60BP4	FP60BP3	FP60BP2	FP60BP1	FP60BP0
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	FP61BP5	FP61BP4	FP61BP3	FP61BP2	FP61BP1	FP61BP0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	FP62BP5	FP62BP4	FP62BP3	FP62BP2	FP62BP1	FP62BP0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	FP63BP5	FP63BP4	FP63BP3	FP63BP2	FP63BP1	FP63BP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-22. LCDRAM (Location 15)

Table 22-26. LCDRAM (Location 15) field descriptions

Field	Description
0:31	FP[60:63]BP[5:0]. LCD segment ON. The FP[60:63]BP[5:0] bit displays (turns on) the LCD segment connected between FP[60:63] and BP[5:0]. 0 LCD segment OFF 1 LCD segment ON0LCD segment OFF

22.5 Functional description

22.5.1 Frontplane, backplane, and LCD system during reset

During a reset the following conditions exist:

- All frontplane enable bits, FP[*n*-1:0]EN are cleared and the ON/OFF control for the display, the LCDEN bit is cleared, thereby forcing all frontplane and backplane driver outputs to the high impedance state. The pin state during reset is defined by the port control module.

- The LCD64F6B system is configured in the default mode, 1/1 duty and 1/1 bias, that means only BP0 is used, system clock as reference, VLCD pin not used as voltage reference.

22.5.2 LCD clock and frame frequency

The frequency of the clock and the clock divider determine the LCD Clock Frequency. The input clock for the prescaler can be selected by LCDRCS bit. The divider is set by the LCD Clock Prescaler bits, in the LCD Prescaler Control Register according to Table 22-27.

Table 22-27. Clock divider

LCLK	Divider
0000	480
0001	2 * 480
0010	2^2 * 480
0011	2^3 * 480
0100	2^4 * 480
0101	2^5 * 480
0110	2^6 * 480
0111	2^7 * 480
1000	2^8 * 480
1001	2^9 * 480
1010	2^{10} * 480
1011	2^{11} * 480
1100	2^{12} * 480
1101	2^{13} * 480
1110	2^{14} * 480
1111	2^{15} * 480

The following formula may be used to calculate the LCD frame frequency:

$$\text{LCD Frame Frequency (Hz)} = \left[\frac{\text{OSCCLK(Hz)}}{\text{Divider}}\right] \qquad \textit{Eqn. 22-1}$$

Example: Clock = 16 MHz, Prescaler = 1010;

$$\left[\frac{16 \bullet 10^6}{2^{10} \bullet 480}\right] \approx 33 \text{ Hz} \qquad \textit{Eqn. 22-2}$$

NOTE

A “Frame” is the full refresh cycle of the display. See Section 22.6, LCD waveform examples, for waveform illustrations.

22.5.3 Contrast adjustment

The LCD Driver module offers two different ways to adjust contrast:

22.5.3.1 Adjusting the supply voltage (VLCD)

The VLCD voltage can directly be used as reference and source to drive the LCD segments. The contrast could be easily adjusted by altering the voltage at the VLCD pin.

22.5.3.2 Adding contrast adjustment phases

Another way to adjust the contrast is to add another phase to the refresh cycle which keeps the voltage the same on all frontplane and backplane electrodes (V_{SS} in this case). This will contribute towards lowering the V_{ONRMS} and V_{OFFRMS} voltages over the segment. For this reason, the whole frame is divided into steps. The value from the Contrast Control register (LCC) determines how many of these steps are taken by the contrast phase. Whenever the length of the contrast adjustment phase is changed, the length of the other phases is automatically changed to ensure the frame length remains the same.

Example: A value of 256 in the Contrast Control (LCC) register will force the contrast phase to take approximately a 256/2047 of the length of the whole frame.

Figure 22-23 shows an example of waveform with 1/1 duty and 1/1 bias ratios with the additional contrast adjustment phases

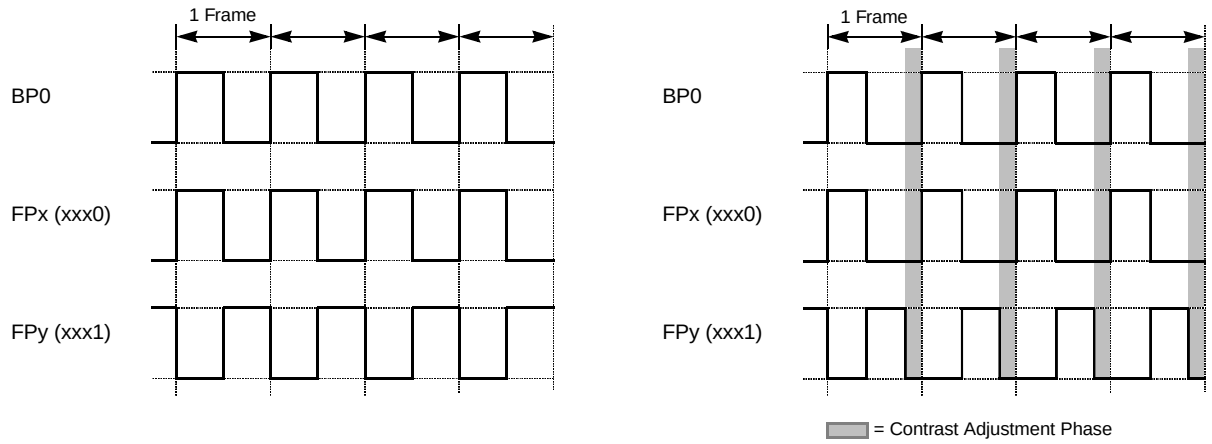


Figure 22-23. Contrast adjustment phases

NOTE

RMS stands for root mean square and is a statistical measure of the magnitude of a varying quantity. It is calculated with the formula:

Eqn. 22-3

$$V_{RMS} = \sqrt{\frac{\int_0^T V^2 dt}{T}}$$

22.5.4 LCD RAM

For a segment on the LCD to be displayed, data must be written to the LCD RAM which is shown in Section 22.4, Memory map and register definition. The bits in the LCD RAM correspond to the segments that are driven by the frontplane and backplane drivers. Writing a 1 to a given location will result in the corresponding display segment being driven with a differential RMS voltage, based on selected reference, necessary to turn the segment ON when the LCDEN bit is set and the corresponding FP[*n*-1:0]EN bit is set. Writing a 0 to a given location will result in the corresponding display segment being driven with a differential RMS voltage necessary to turn the segment OFF. The LCD RAM is a dual port RAM that interfaces with the internal address and data buses of the MCU. It is possible to read from LCD RAM locations for scrolling purposes. Writing or reading of the LCDEN bit does not change the contents of the LCD RAM. After a reset, the LCD RAM contents will be cleared.

22.5.5 LCD driver system enable and frontplane enable sequencing

If LCDEN = 0 (LCD Driver System disabled) and the frontplane enable bit, FP[*n*-1:0]EN, is set, the frontplane driver waveform will not appear on the output until LCDEN is set. If LCDEN = 1 (LCD Driver System enabled), the frontplane driver waveform will appear on the output as soon as the corresponding frontplane enable bit, FP[*n*-1:0]EN, in the registers FPENR0 and FPENR1 is set.

22.5.6 LCD driver backplane remapping

The backplane and frontplane pins can be remapped/swapped using LCDBPS and LCDBPA. While LCDBPA adds the non available backplane waveforms onto FP pins, LCDBPS does a swapping of backplanes waveforms with frontplanes waveforms on selected pins. The swapping of LCDBPS depends on the availability of Frontplane pins FP[n-1:0]. If the number of frontplanes n implemented is not sufficient no remapping will occur. See Table 22-28 for details.

Table 22-28. Backplane remapping

LCDBPS	Condition: FP[n-1:0]	Remapping LCDBPA=0 BP[m-1:0]	Remapping LCDBPA=1 BP[m-1:0]
000		no remapping	FP[5-m:0] <- BP[5:m] if m=6 no remapping
001		BP[m-1:0] <- FP[m-1+4:4] FP[m-1+4:4] <- BP[m-1:0]	BP[m-1:0] <- FP[m-1+4:4] FP[5+4:4] <- BP[5:0]
010		BP[m-1:0] <- FP[m-1+12:12] FP[m-1+12:12] <- BP[m-1:0]	BP[m-1:0] <- FP[m-1+12:12] FP[5+12:12] <- BP[5:0]
011		BP[m-1:0] <- FP[m-1+20:20] FP[m-1+20:20] <- BP[m-1:0]	BP[m-1:0] <- FP[m-1+20:20] FP[5+20:20] <- BP[5:0]
100	n=36 or n>36	BP[m-1:0] <- FP[m-1+28:28] FP[m-1+28:28] <- BP[m-1:0]	BP[m-1:0] <- FP[m-1+28:28] FP[5+28:28] <- BP[5:0]
101	n=44 or n>44	BP[m-1:0] <- FP[m-1+36:36] FP[m-1+36:36] <- BP[m-1:0]	BP[m-1:0] <- FP[m-1+36:36] FP[5+36:36] <- BP[5:0]
110	n=52 or n>52	BP[m-1:0] <- FP[m-1+44:44] FP[m-1+44:44] <- BP[m-1:0]	BP[m-1:0] <- FP[m-1+44:44] FP[5+44:44] <- BP[5:0]
111	n=60 or n>60	BP[m-1:0] <- FP[m-1+52:52] FP[m-1+52:52] <- BP[m-1:0]	BP[m-1:0] <- FP[m-1+52:52] FP[5+52:52] <- BP[5:0]

Examples:

1. 40 (n=40) Frontplanes and 4 (m=4) backplanes are implemented. LCDBPS is set to 000 and LCDBPA is set to 1

   Frontplanes FP[39:2] and BP[3:0] will stay the same, but FP[1:0] pins will controlled by BP[5:4] functionality.
2. 40 (n=40) Frontplanes and 4 (m=4) backplanes are implemented. LCDBPS is set to 010 and LCDBPA is set to 0

   Frontplanes FP[39:16] and FP[11:0] will stay the same, but FP[15:12] pins are swapped with BP[3:0].
3. 40 (n=40) Frontplanes and 4 (m=4) backplanes are implemented. LCDBPS is set to 010 and LCDBPA is set to 1

   Frontplanes FP[39:18] and FP[11:0] will stay the same. FP[15:12] pins are swapped with BP[3:0]. FP[17:16] is replaced by BP[5:4] functionality.
4. 40 (n=40) Frontplanes and 4 (m=4) backplanes are implemented. LCDBPS is set to 101 and LCDBPA is set to 1

   No remapping: Frontplanes FP[39:0] and BP[3:0] will stay the same, because condition n>=44 is not fulfilled.

22.5.7 LCD bias and modes of operation

The LCD64F6B driver has seven modes of operation:

- 1/1 Duty (1 backplane), 1/1 Bias (2 voltage levels)

- 1/2 Duty (2 backplanes), 1/2 Bias (3 voltage levels)
- 1/2 Duty (2 backplanes), 1/3 Bias (4 voltage levels)
- 1/3 Duty (3 backplanes), 1/3 Bias (4 voltage levels)
- 1/4 Duty (4 backplanes), 1/3 Bias (4 voltage levels)
- 1/5 Duty (5 backplanes), 1/3 Bias (4 voltage levels)
- 1/6 Duty (6 backplanes), 1/3 Bias (4 voltage levels)

The voltage levels required for the different operating modes are generated internally based on selected reference voltage. If VLCD as reference selected, changing VLCD alters the differential RMS voltage across the segments in the ON and OFF states, thereby setting the display contrast.

The backplane waveforms are continuous and repetitive every frame. They are fixed within each operating mode and are not affected by the data in the LCD RAM.

The frontplane waveforms generated are dependent on the state (ON or OFF) of the LCD segments as defined in the LCD RAM. The LCD driver hardware uses the data in the LCD RAM to construct the frontplane waveform to create a differential RMS voltage necessary to turn the segment ON or OFF.

The LCD duty is decided by the DUTY bits in the LCD Control Register (LCDCR). The number of bias voltage levels is determined by the BIAS bit in the LCDCR. Table 22-29 summarizes the Multiplex modes (duties) and the bias voltage levels that can be selected for each multiplex mode (duty). The backplane pins have their corresponding backplane waveform output BP[*m*-1:0] in high impedance state when in the OFF state as indicated in Table 22-29. In the OFF state the corresponding pins BP[*m*-1:0]can be used for other functionality.

Table 22-29. LCD duty and bias

<table>
<tr><td rowspan="2">Duty</td><td>LCDCR Register</td><td colspan="6">Backplanes</td><td colspan="2">Bias Level</td></tr>
<tr><td>DUTY</td><td>BP5</td><td>BP4</td><td>BP3</td><td>BP2</td><td>BP1</td><td>BP0</td><td>BIAS=0</td><td>BIAS=1</td></tr>
<tr><td>1/1</td><td>000</td><td colspan="5">OFF</td><td>ON</td><td colspan="2">1/1</td></tr>
<tr><td>1/2</td><td>001</td><td colspan="4">OFF</td><td colspan="2">ON</td><td>1/2</td><td>1/3</td></tr>
<tr><td>1/3</td><td>010</td><td colspan="3">OFF</td><td colspan="3">ON</td><td colspan="2">1/3</td></tr>
<tr><td>1/4</td><td>011</td><td colspan="2">OFF</td><td colspan="4">ON</td><td colspan="2">1/3</td></tr>
<tr><td>1/5</td><td>100</td><td>OFF</td><td colspan="5">ON</td><td colspan="2">1/3</td></tr>
<tr><td>1/6</td><td>101</td><td colspan="6">ON</td><td colspan="2">1/3</td></tr>
<tr><td>1/6</td><td>110</td><td colspan="6">ON</td><td colspan="2">1/3</td></tr>
<tr><td>1/6</td><td>111</td><td colspan="6">ON</td><td colspan="2">1/3</td></tr>
</table>

22.5.8 Operation in power saving modes

22.5.8.1 Operation in Stop mode

The LCD64F6B system operation during Stop mode is controlled by the LCDRST bit in the LCD Control Register (LCDCR).

If the LCD64F6B is requested to enter Stop mode and LCDRST is cleared while LCDEN is set the LCD waveform generation clocks are stopped and the LCD64F6B drivers pull down to ground those frontplane and backplane pins that were enabled before entering Stop mode.The contents of the LCD RAM and the LCD registers retain the values they had prior to entering stop mode.

If LCDRST is set while LCDEN is set and the LCD64F6B is requested to enter Stop mode the LCD waveform generation is continued.

22.5.8.2 Operation in Standby mode

See device guide for information if the LCD64F6B is powered down or stays powered in Standby mode. The LCD64F6B system operation during Standby mode is controlled by the LCDRST bit in the LCD Control Register (LCDCR).

If the LCD64F6B is powered down by the system, those frontplane and backplane pins goes high impedance, that were enabled before entering Standby mode.

If the LCD64F6B is not powered down by the system, and is requested to enter Standby mode and LCDRST is cleared while LCDEN is set the LCD waveform generation clocks are stopped and the LCD64F6B drivers pull down to ground those frontplane and backplane pins that were enabled before entering Standby mode.The contents of the LCD RAM and the LCD registers retain the values they had prior to entering Standby mode.

If the LCD64F6B is not powered down by the system, and is requested to enter Standby mode and LCDRST is set while LCDEN is set the LCD waveform generation is continued.

NOTE

The user needs to take care that the system keeps the clock applied to the running LCD64F6B module during all modes where it is enabled. If no clock is applied while the LCD64F6B module is running the LCD could be damaged.

22.5.9 Other power saving

The LCD64F6B has features to adjust the frontplane and backplane drive strength and to boost the drive strength while the planes are switching.

22.5.9.1 LCD reference clock select

Using LCDRCS the LCD reference clock can be selected. If LCDRCS is cleared, the system clock is applied as reference clock to the prescaler input. If LCDRCS is set, the source clock for LCD Driver system (prescaler input) is OSC clock. See device level documentation for detail on OSC clk. Selecting

the lower power consuming clock of both as reference clock for the prescaler input can save power consumption.

22.5.9.2 Boost at switching

Since the LCD appears like a capacitance to the driver it is sometimes useful to boost the output current during transitions to increase the slew rate of the driver. If boosting is enabled, the drive strength capability is increased a little ahead in time when the planes are switching, and reduced again to normal drive after a certain time. To enable the boosting BSTEN need to be set. If BSTEN is not set, standard drive strength is used.

Using the BST bit if BSTEN is set, LCD output current magnification (boost) can be selected. Available is 8 times boosting, when BST is deasserted and 16 times boosting when BST is asserted.

The selected boost via BST bit can be switched to be always on by setting BSTAO, if BSTEN is set.

22.5.9.3 Standard drive selection

The output current has a direct impact on the power consumption and drive capability of the LCD64F6B module. The PWR bits select the output current height and can be used to influence power consumption and drive strength. Table 22-30 lists the possible selections of PWR.

Table 22-30. Output Current Selection (PWR)

PWR	Current
00	Standard Current
01	2 * Standard Current
10	3 * Standard Current
11	4 * Standard Current

22.5.9.4 Usage recommendation

It is recommended to start with maximum output current and maximum enabled boosting. As long as the LCD is fully functional in all environments a reduction of Output Current and boosting can be done.

When reducing the Output Current (PWR), the power consumption of the LCD module and the maximum switching current of the planes is reduced.

When reducing or disabling the boosting (BST,BSTEN), the output current, while planes are switching, is reduced.

The user can select any combination of PWR, BST, BSTEN, BSTAO to what fit his needs best. Figure 22-24 gives an overview what is the impact when taking advance of Boost at switching and Standard drive selection.

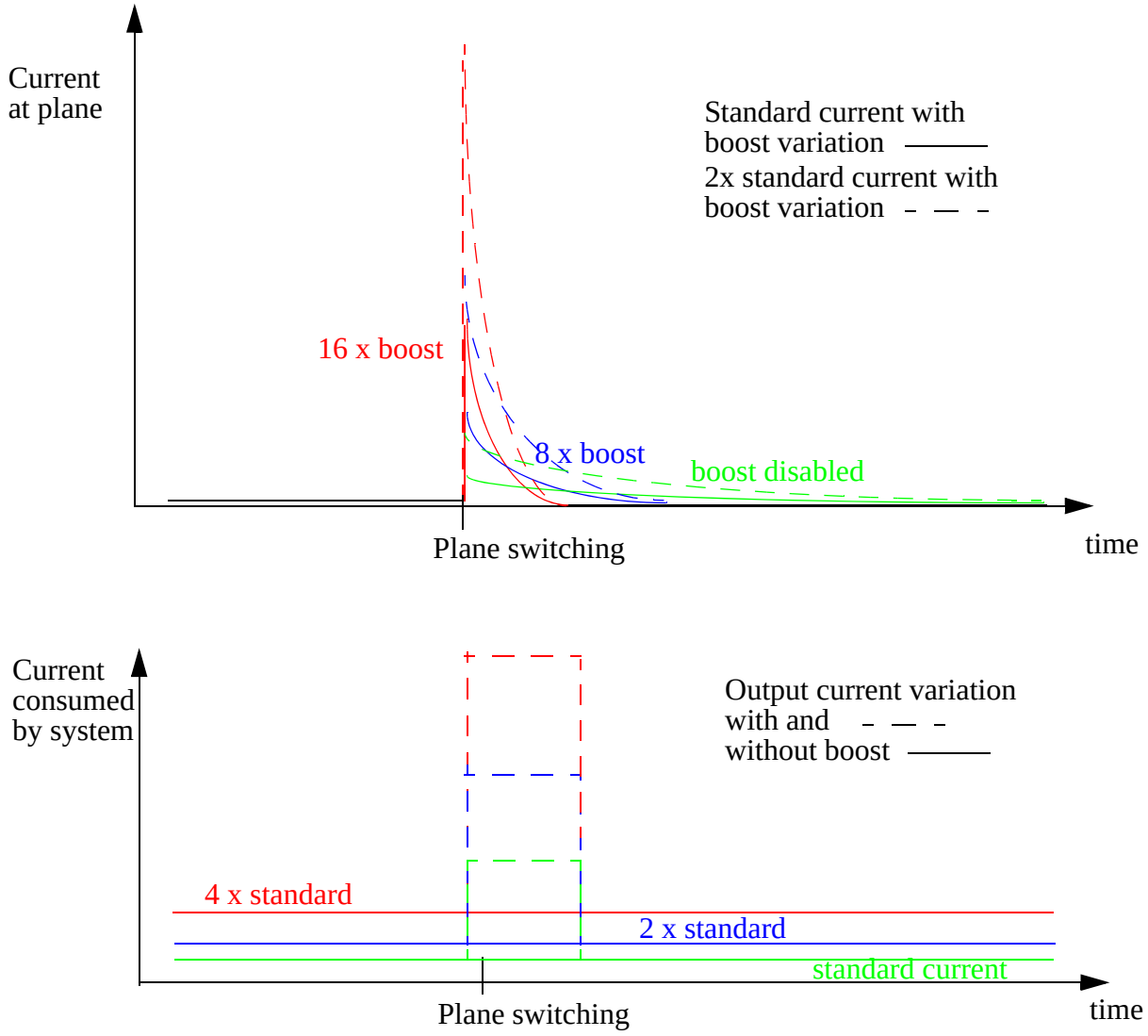


Figure 22-24. PWR, BST, BSTEN example diagram

22.5.10 Interrupts

This section describes all interrupts originated by LCD64F6B.

22.5.10.1 EOF interrupt

LCDEN must be set to enable the interrupt at end of frame feature. Every time a frame generation was completed a counter is decremented by 1. If the counter reaches zero End of Frame interrupt Flag (EOF) bit is set and the counter is reset to NOF. The counter is reset to request interrupt at the end of every frame, as if NOF would be 0x00, if LCDEN is asserted while the LCD is off.

The EOF flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LCDINT = 1), EOF causes an interrupt request. The number of frames (NOF) bits determine how many frames are count until EOF flag is set. The possible number of frames are:

NOF= 0x00: An interrupt is requested at the end of every frame.

NOF= 0x01: An interrupt is requested at the end of every second frame.

NOF= 0x02: An interrupt is requested at the end of every third frame.

...

NOF= 0xFF: An interrupt is requested at the end of every 256th frame.

22.6 LCD waveform examples

The following figures show the timing examples of the LCD output waveforms for the available modes of operation. The contrast control is disabled in all examples (CCEN = 0 in LCDCCR). Selected reference voltage is VLCD, but could also be VDDX.

22.6.1 1/1 Duty multiplexed with 1/1 Bias mode

Duty = 1/1:DUTY = 000
Bias = 1/1:BIAS = 0 or BIAS = 1
$V_0 = V_1 = VSSX$ $V_2 = V_3 = VLCD$
- Only BP0 is used, a maximum of 64 segments are displayed.

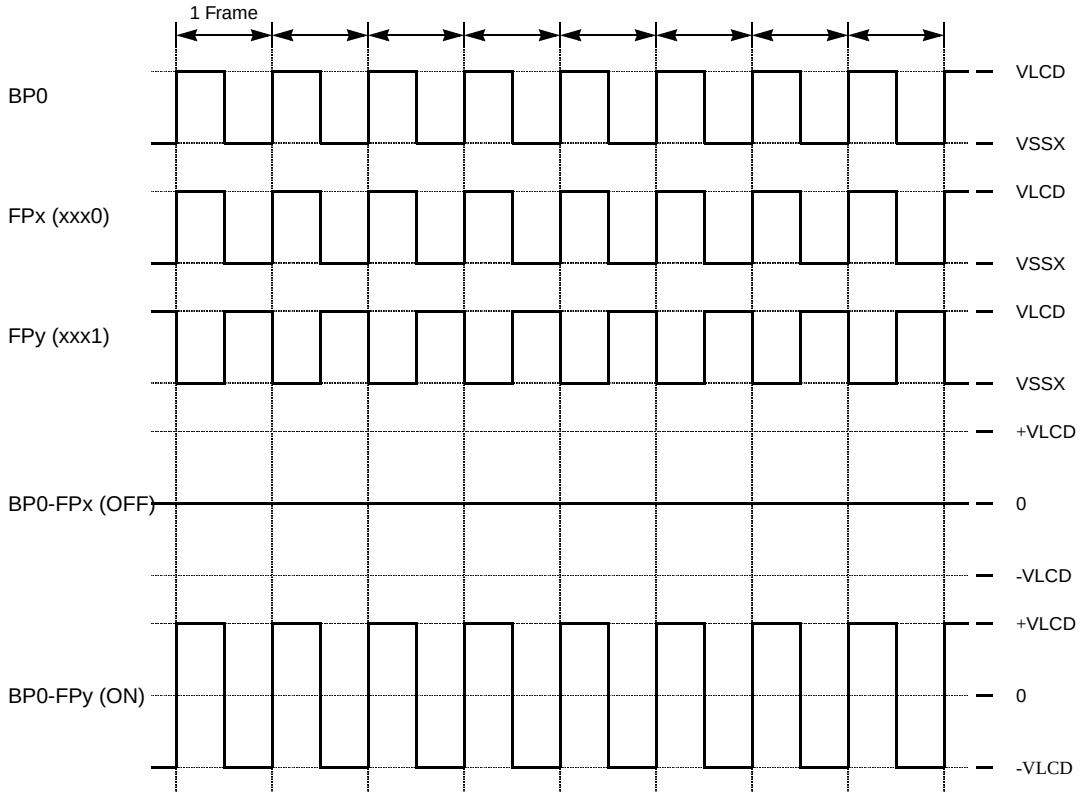


Figure 22-25. 1/1 Duty and 1/1 Bias

22.6.2 1/2 duty multiplexed with 1/2 Bias mode

Duty = 1/2:DUTY = 001
Bias = 1/2:BIAS = 0
V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD
- Only BP0 and BP1 are used, a maximum of 128 segments are displayed.

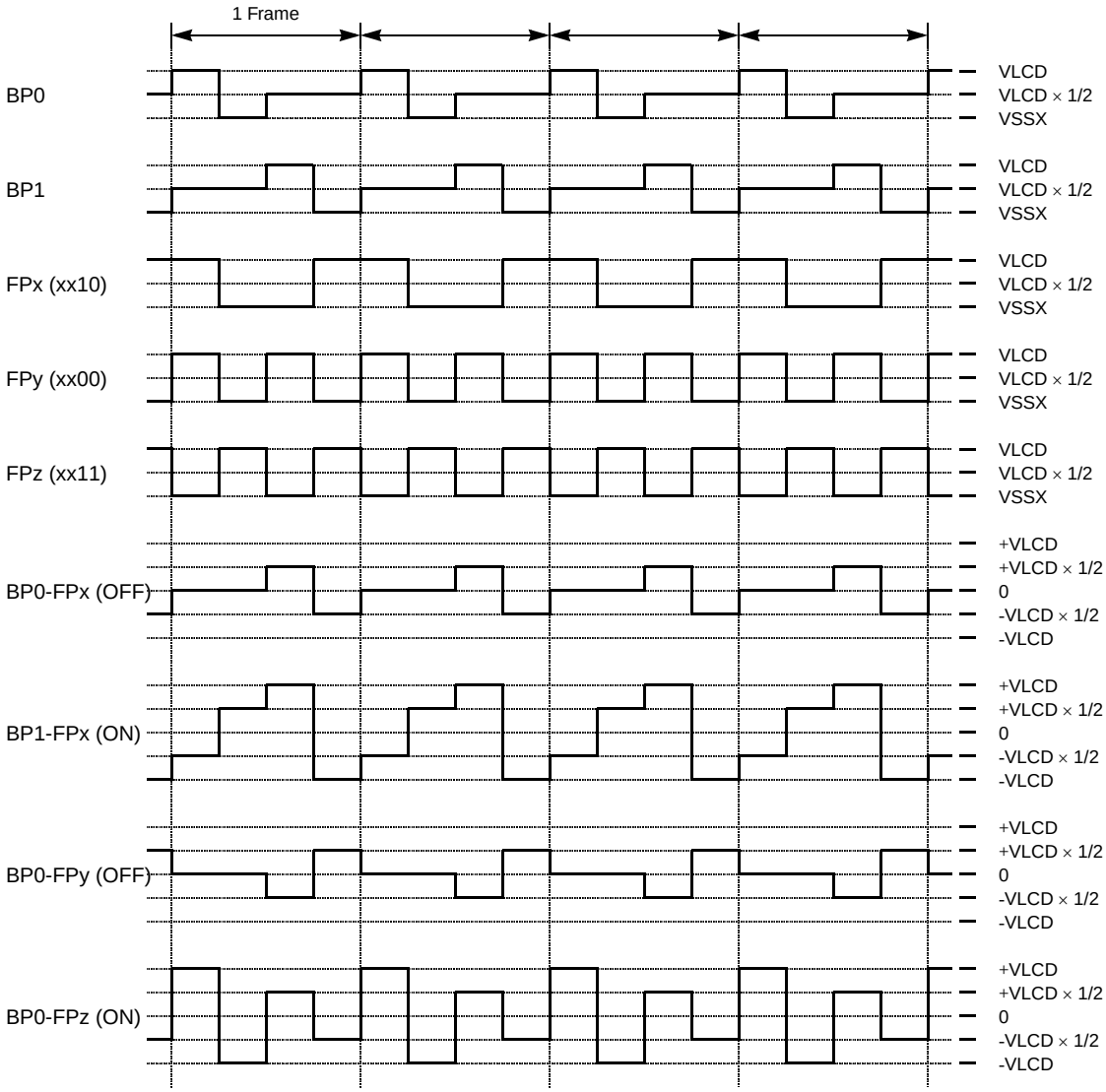


Figure 22-26. 1/2 Duty and 1/2 Bias

22.6.3 1/2 duty multiplexed with 1/3 Bias mode

Duty = 1/2:DUTY = 001
Bias = 1/3:BIAS = 1
V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD
- Only BP0 and BP1 are used, a maximum of 128 segments are displayed.

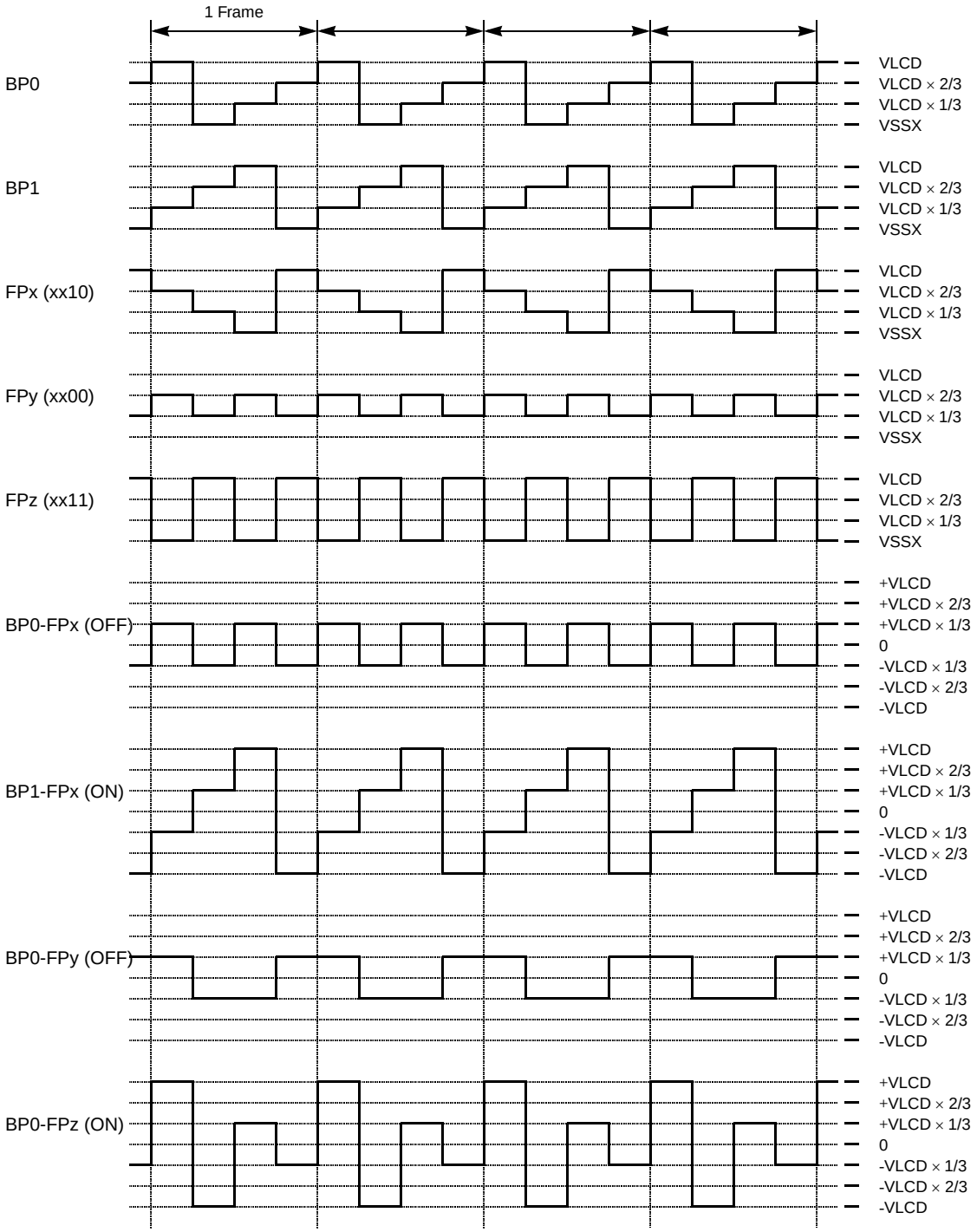


Figure 22-27. 1/2 Duty and 1/3 Bias

22.6.4 1/3 Duty multiplexed with 1/3 Bias mode

Duty = 1/3:DUTY = 010
Bias = 1/3:BIAS = 0 or BIAS = 1

V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD
- Only BP0, BP1 and BP2 are used, a maximum of 192 segments are displayed.

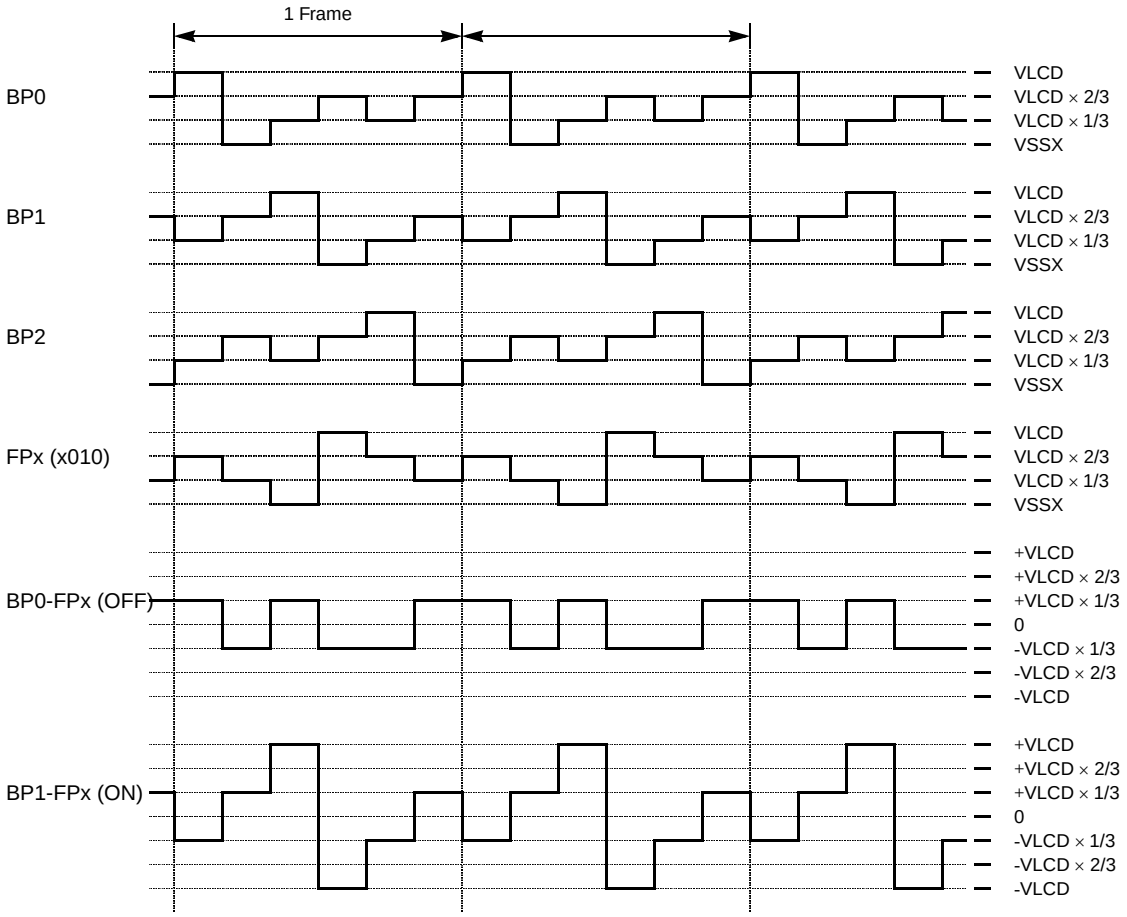


Figure 22-28. 1/3 Duty and 1/3 Bias

22.6.5 1/4 Duty multiplexed with 1/3 Bias mode

Duty = 1/4:DUTY = 011
Bias = 1/3:BIAS = 0 or BIAS = 1
V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD
- BP4 and BP5 are not used, a maximum of 256 segments are displayed.

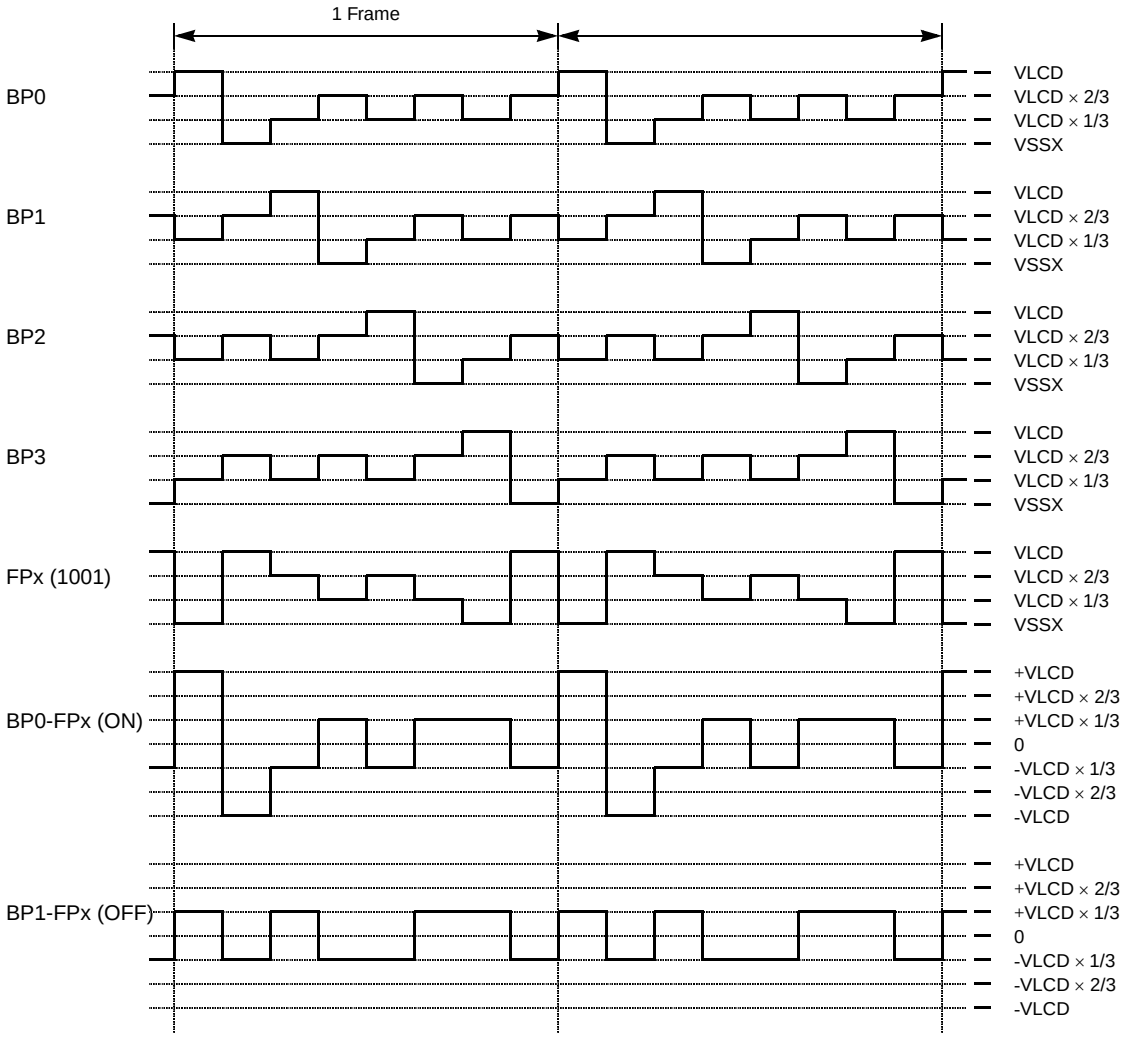


Figure 22-29. 1/4 Duty and 1/3 Bias

22.6.6 1/5 Duty multiplexed with 1/3 Bias

Duty = 1/5:DUTY = 100
Bias = 1/3:BIAS = 0 or BIAS = 1
V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD
- BP5 is not used, a maximum of 320 segments are displayed.

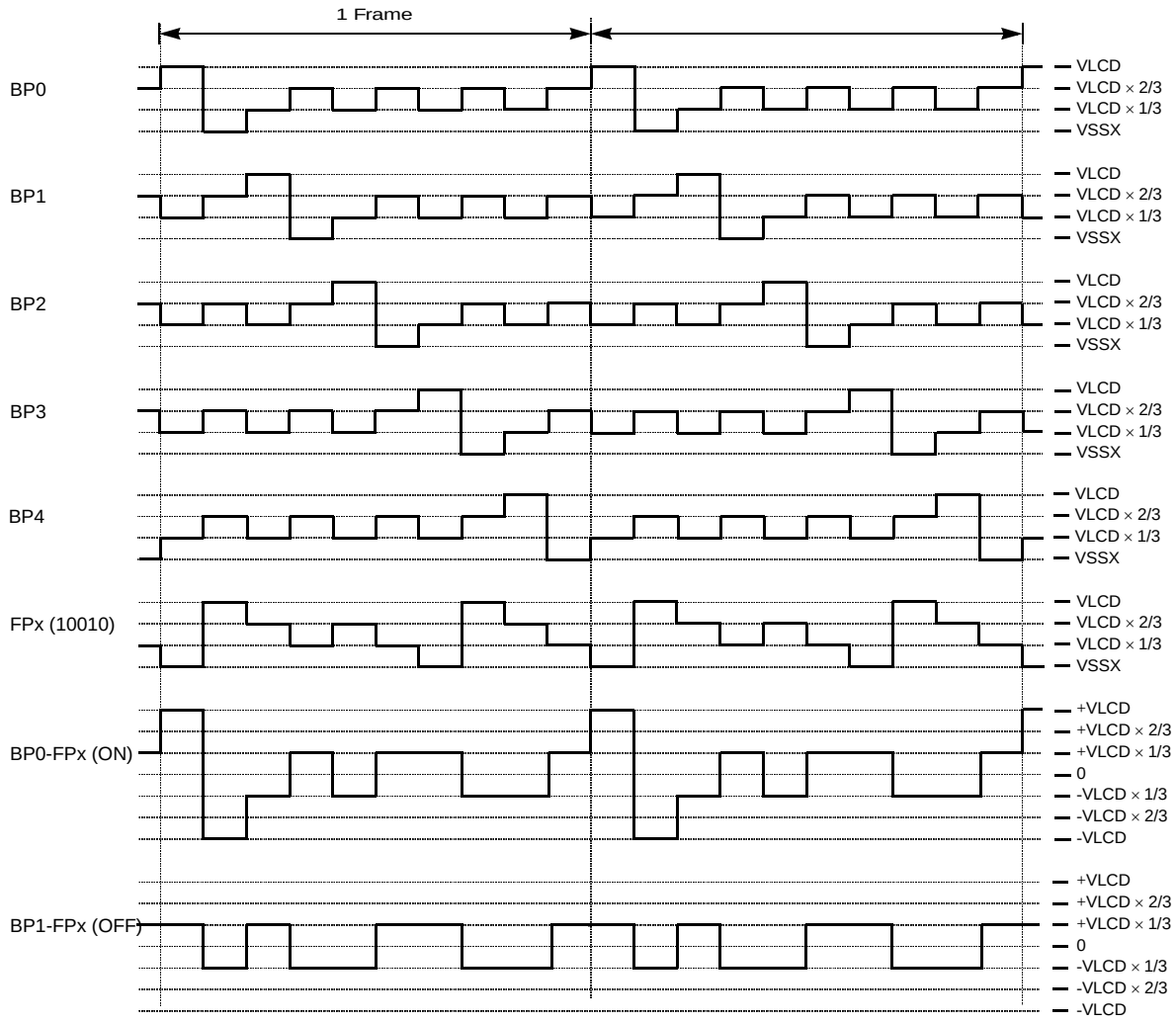


Figure 22-30. 1/5 Duty and 1/3 Bias

22.6.7 1/6 Duty multiplexed with 1/3 Bias mode

Duty = 1/5:DUTY = 101
Bias = 1/3:BIAS = 0 or BIAS = 1
V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD
- All backplanes are used, a maximum of 384 segments are displayed.

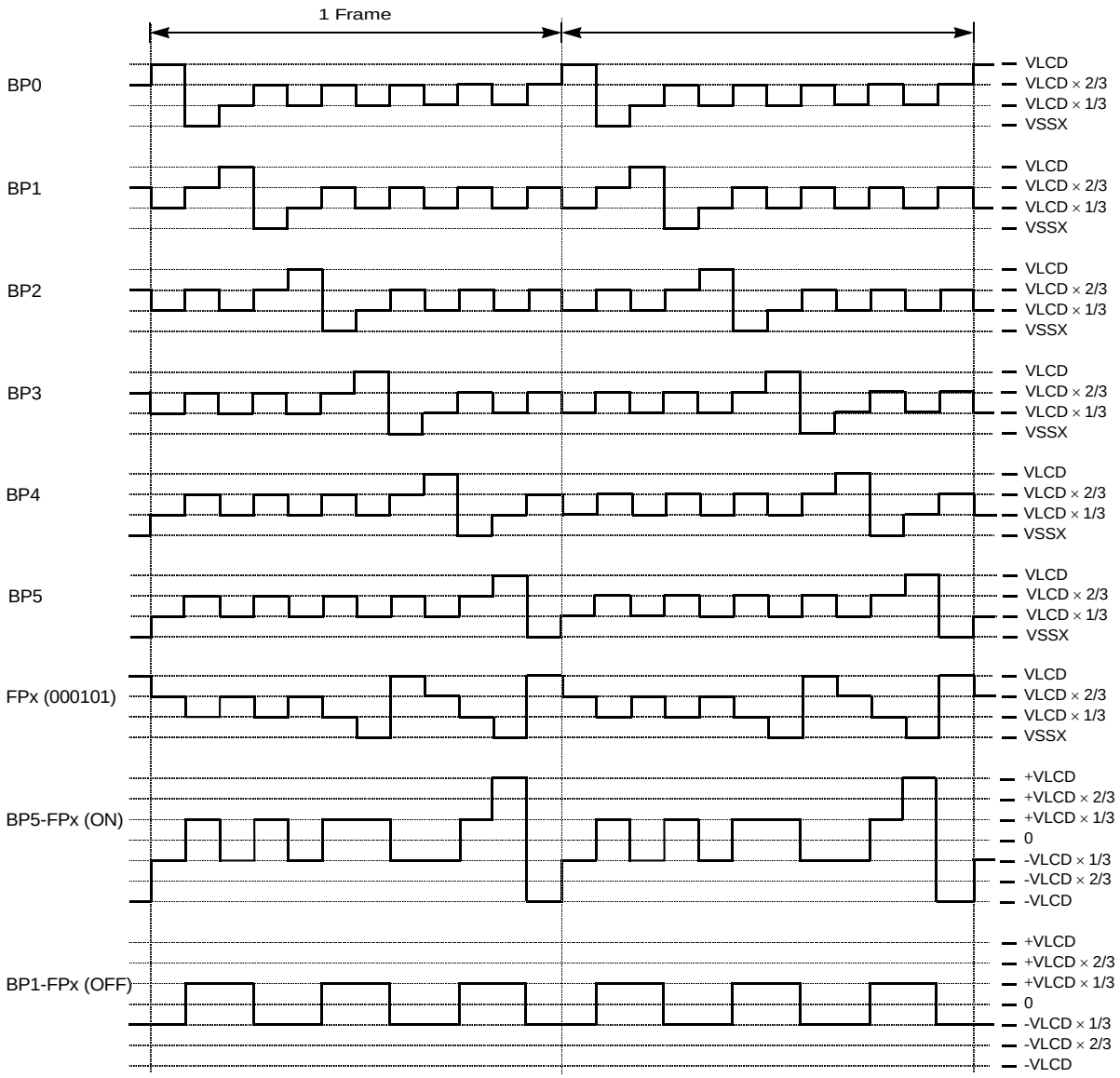


Figure 22-31. 1/6 Duty and 1/3 Bias

22.7 Initialization information

This is a step-wise example instruction for initializing. The initial values of all registers are the reset values.

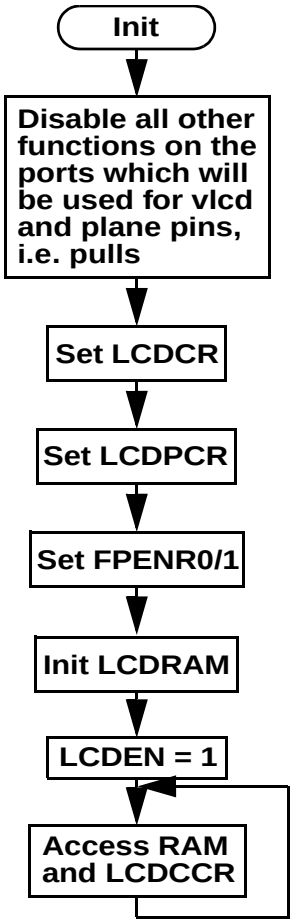


Figure 22-32. Example initialization diagram

Chapter 23
LIN Controller (LINFlex)

23.1 Introduction

The LINFlex (Local Interconnect Network Flexible) controller interfaces the LIN network and supports the LIN protocol versions 1.3; 2.0 and 2.1; and J2602 in both Master and Slave modes. LINFlex includes a LIN mode that provides additional features (compared to standard UART) to ease LIN implementation, improve system robustness, minimize CPU load and allow slave node resynchronization.

This device contains two LINFlex modules, LINFlex_0 and LINFlex_1. Of these, only LINFlex_0 can be configured in Slave mode.

23.2 Main features

23.2.1 LIN mode features

- Supports LIN protocol versions 1.3, 2.0, 2.1 and J2602
- Master mode with autonomous message handling
- Classic and enhanced checksum calculation and check
- Single 8-byte buffer for transmission/reception
- Extended frame mode for In-Application Programming (IAP) purposes
- Wake-up event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response and frame timeout
- Slave mode
  - — Autonomous header handling
  - — Autonomous transmit/receive data handling
- LIN automatic resynchronization, allowing operation with 16 MHz fast internal RC oscillator as clock source
- 16 identifier filters for autonomous message handling in Slave mode

23.2.2 UART mode features

- Full duplex communication
- 8- or 9-bit with parity
- 4-byte buffer for reception, 4-byte buffer for transmission
- 8-bit counter for timeout management

23.2.3 Features common to LIN and UART

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock:
  - Initialization
  - Normal
  - Sleep
- 2 test modes:
  - Loopback
  - Self-test
- Maskable interrupts

23.3 General description

The increasing number of communication peripherals embedded on microcontrollers, for example CAN, LIN and SPI, requires more and more CPU resources for communication management. Even a 32-bit microcontroller is overloaded if its peripherals do not provide high-level features to autonomously handle the communication.

Even though the LIN protocol with a maximum baud rate of 20 kbit/s is relatively slow, it still generates a non-negligible load on the CPU if the LIN is implemented on a standard UART, as usually the case.

To minimize the CPU load in Master mode, LINFlex handles the LIN messages autonomously.

In Master mode, once the software has triggered the header transmission, LINFlex does not request any software intervention until the next header transmission request in transmission mode or until the checksum reception in reception mode.

To minimize the CPU load in Slave mode, LINFlex requires software intervention only to:

- Trigger transmission or reception or data discard depending on the identifier
- Write data into the buffer (transmission mode) or read data from the buffer (reception mode) after checksum reception

If filter mode is activated for Slave mode, LINFlex requires software intervention only to write data into the buffer (transmission mode) or read data from the buffer (reception mode)

The software uses the control, status and configuration registers to:

- Configure LIN parameters (for example, baud rate or mode)
- Request transmissions
- Handle receptions
- Manage interrupts
- Configure LIN error and timeout detection
- Process diagnostic information

The message buffer stores transmitted or received LIN frames.

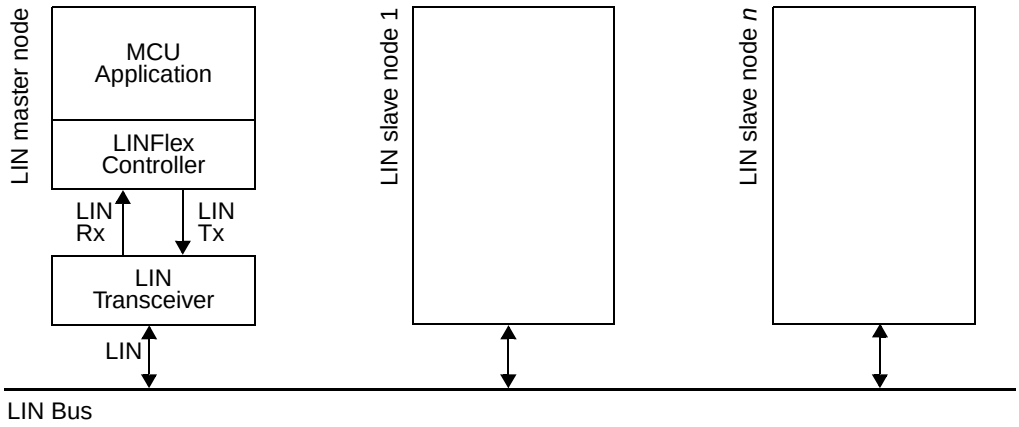
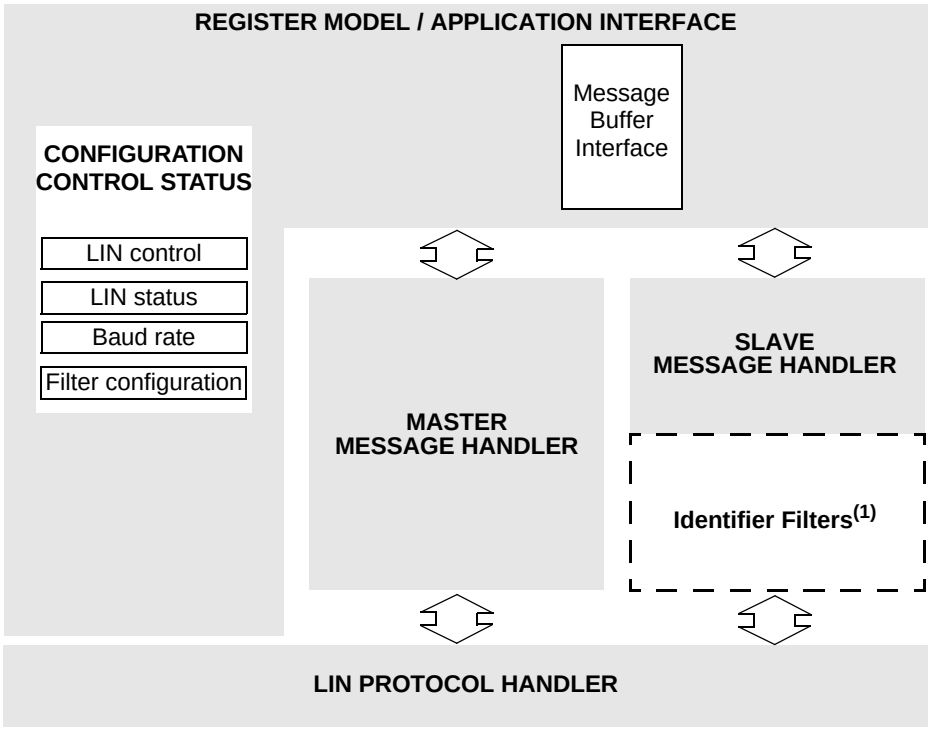


Figure 23-1. LIN topology network



1. Filter activation optional

Figure 23-2. LINFlex block diagram

23.4 Fractional baud rate generation

The baud rates for the receiver and transmitter are both set to the same value as programmed in the Mantissa (LINIBRR) and Fraction (LINFBRR) registers.

Eqn. 23-1

$$\text{Tx/ Rx baud} = \frac{f_{periph\_set\_1\_clk}}{(16 \times LFDIV)}$$

LFDIV is an unsigned fixed point number. The 12-bit mantissa is coded in the LINIBRR and the fraction is coded in the LINFBRR.

The following examples show how to derive LFDIV from LINIBRR and LINFBRR register values:

Example 23-1. Deriving LFDIV from LINIBRR and LINFBRR register values

If LINIBRR = 27d and LINFBRR = 12d, then

Mantissa (LFDIV) = 27d

Fraction (LFDIV) = 12/16 = 0.75d

Therefore LFDIV = 27.75d

Example 23-2. Programming LFDIV from LINIBRR and LINFBRR register values

To program LFDIV = 25.62d,

LINFBRR = 16 × 0.62 = 9.92, nearest real number 10d = 0xA

LINIBRR = mantissa (25.620d) = 25d = 0x19

NOTE

The baud counters are updated with the new value of the baud registers after a write to LINIBRR. Hence the baud register value must not be changed during a transaction. The LINFBRR (containing the Fraction bits) must be programmed before the LINIBRR.

NOTE

LFDIV must be greater than or equal to 1.5d, i.e. LINIBRR = 1 and LINFBRR = 8. Therefore, the maximum possible baudrate is fperiph_set_1_clk / 24.

Table 23-1. Error calculation for programmed baud rates

Baud rate	$f_{periph\_set\_1\_clk}$ = MHz				$f_{periph\_set\_1\_clk}$ = 16 MHz			
	Actual	Value programmed in the baud rate register		% Error = (Calculated – Desired) baud rate / Desired baud rate	Actual	Value programmed in the baud rate register		% Error = (Calculated – Desired) baud rate / Desired baud rate
		LINIBRR	LINFBRR			LINIBRR	LINFBRR	
10417	10416.7	384	0	–0.003	10416.7	96	0	–0.003

23.5 Operating modes

LINFlex has three main operating modes: Initialization, Normal and Sleep. After a hardware reset, LINFlex is in Sleep mode to reduce power consumption. The software instructs LINFlex to enter Initialization mode or Sleep mode by setting the INIT bit or SLEEP bit in the LINCR1.

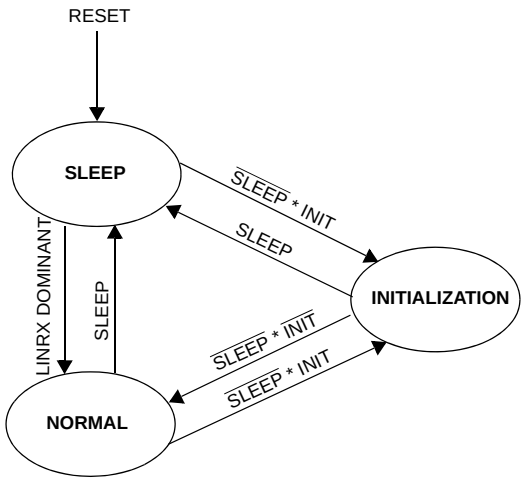


Figure 23-3. LINFlex operating modes

23.5.1 Initialization mode

The software can be initialized while the hardware is in Initialization mode. To enter this mode the software sets the INIT bit in the LINCR1.

To exit Initialization mode, the software clears the INIT bit.

While in Initialization mode, all message transfers to and from the LIN bus are stopped and the status of the LIN bus output LINTX is recessive (high).

Entering Initialization mode does not change any of the configuration registers.

To initialize the LINFlex controller, the software selects the mode (LIN Master, LIN Slave or UART), sets up the baud rate register and, if LIN Slave mode with filter activation is selected, initializes the identifier list.

23.5.2 Normal mode

Once initilization is complete, software clears the INIT bit in the LINCR1 to put the hardware into Normal mode.

23.5.3 Low-power mode (Sleep)

To reduce power consumption, LINFlex has a low-power mode called Sleep mode. To enter Sleep mode, software sets the SLEEP bit in the LINCR1. In this mode, the LINFlex clock is stopped. Consequently, the LINFlex will not update the status bits but software can still access the LINFlex registers.

LINFlex can be awakened (exit Sleep mode) either by software clearing the SLEEP bit or on detection of LIN bus activity if automatic wakeup mode is enabled (AWUM bit is set).

On LIN bus activity detection, hardware automatically performs the wakeup sequence by clearing the SLEEP bit if the AWUM bit in the LINCR1 is set. To exit from Sleep mode if the AWUM bit is cleared, software clears the SLEEP bit when a wakeup event occurs.

23.6 Test modes

Two test modes are available to the user: Loopback mode and Self-test mode. They can be selected by the LBKM and SFTM bits in the LINCR1. These bits must be configured while LINFlex is in Initialization mode. Once one of the two test modes has been selected, LINFlex must be started in Normal mode.

23.6.1 Loopback mode

LINFlex can be put in Loopback mode by setting the LBKM bit in the LINCR. In Loopback mode, the LINFlex treats its own transmitted messages as received messages.

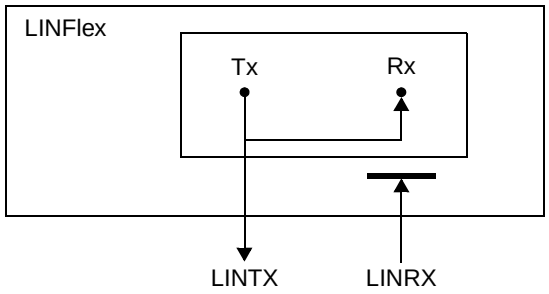


Figure 23-4. LINFlex in Loopback mode

This mode is provided for self-test functions. To be independent of external events, the LIN core ignores the LINRX signal. In this mode, the LINFlex performs an internal feedback from its Tx output to its Rx input. The actual value of the LINRX input pin is disregarded by the LINFlex. The transmitted messages can be monitored on the LINTX pin.

23.6.2 Self-test mode

LINFlex can be put in Self-test mode by setting the LBKM and SFTM bits in the LINCR. This mode can be used for a “Hot Self-test”, meaning the LINFlex can be tested as in Loopback mode but without affecting a running LIN system connected to the LINTX and LINRX pins. In this mode, the LINRX pin is disconnected from the LINFlex and the LINTX pin is held recessive.

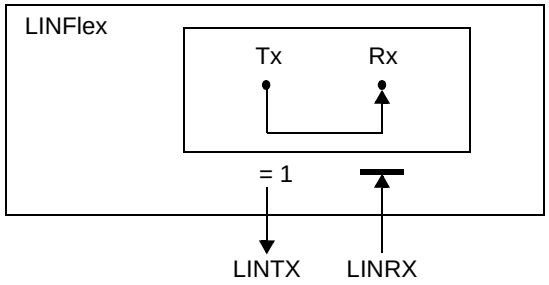


Figure 23-5. LINFlex in Self-test mode

23.7 Memory map and registers description

23.7.1 Memory map

The base addresses for the LINFlex modules are as follows:

- 0xFFE4_0000 (LINFlex_0)
- 0xFFE4_4000 (LINFlex_1)

Table 23-2 shows the LINFlex memory map.

Table 23-2. LINFlex memory map

Offset from LINFLEX_BASE	Register	Access	Reset value	Location
0x0000	LIN control register 1 (LINCR1)	R/W	See note [1]	on page 837
0x0004	LIN interrupt enable register (LINIER)	R/W	0x0000_0000	on page 840
0x0008	LIN status register (LINSR)	R/W	0x0000_0080	on page 842
0x000C	LIN error status register (LINESR)	R/W	0x0000_0000	on page 845
0x0010	UART mode control register (UARTCR)	R/W	0x0000_0000	on page 846
0x0014	UART mode status register (UARTSR)	R/W	0x0000_0000	on page 848
0x0018	LIN timeout control status register (LINTCSR)	R/W	0x0000_0040	on page 850
0x001C	LIN output compare register (LINOCR)	R/W	0x0000_FFFF	on page 851
0x0020	LIN timeout control register (LINTOCR)	R/W	See note [2]	on page 851
0x0024	LIN fractional baud rate register (LINFBRR)	R/W	0x0000_0000	on page 852
0x0028	LIN integer baud rate register (LINIBRR)	R/W	0x0000_0000	on page 853
0x002C	LIN checksum field register (LINCFR)	R/W	0x0000_0000	on page 854
0x0030	LIN control register 2 (LINCR2)	R/W	See note [3]	on page 854
0x0034	Buffer identifier register (BIDR)	R/W	0x0000_0000	on page 856
0x0038	Buffer data register LSB (BDRL)[4]	R/W	0x0000_0000	on page 857

Table 23-2. LINFlex memory map (continued)

Offset from LINFLEX_BASE	Register	Access	Reset value	Location
0x003C	Buffer data register MSB (BDRM)[5]	R/W	0x0000_0000	on page 857
0x0040	Identifier filter enable register (IFER)[6]	R/W	0x0000_0000	on page 858
0x0044	Identifier filter match index (IFMI)[6]	R	0x0000_0000	on page 859
0x0048	Identifier filter mode register (IFMR)[6]	R/W	0x0000_0000	on page 860
0x004C	Identifier filter control register 0 (IFCR0)[6]	R/W	0x0000_0000	on page 861
0x0050	Identifier filter control register 1 (IFCR1)[6]	R/W	0x0000_0000	on page 862
0x0054	Identifier filter control register 2 (IFCR2)[6]	R/W	0x0000_0000	on page 862
0x0058	Identifier filter control register 3 (IFCR3)[6]	R/W	0x0000_0000	on page 862
0x005C	Identifier filter control register 4 (IFCR4)[6]	R/W	0x0000_0000	on page 862
0x0060	Identifier filter control register 5 (IFCR5)[6]	R/W	0x0000_0000	on page 862
0x0064	Identifier filter control register 6 (IFCR6)[6]	R/W	0x0000_0000	on page 862
0x0068	Identifier filter control register 7 (IFCR7)[6]	R/W	0x0000_0000	on page 862
0x006C	Identifier filter control register 8 (IFCR8)[6]	R/W	0x0000_0000	on page 862
0x0070	Identifier filter control register 9 (IFCR9)[6]	R/W	0x0000_0000	on page 862
0x0074	Identifier filter control register 10 (IFCR10)[6]	R/W	0x0000_0000	on page 862
0x0078	Identifier filter control register 11 (IFCR11)[6]	R/W	0x0000_0000	on page 862
0x007C	Identifier filter control register 12 (IFCR12)[6]	R/W	0x0000_0000	on page 862
0x0080	Identifier filter control register 13 (IFCR13)[6]	R/W	0x0000_0000	on page 862
0x0084	Identifier filter control register 14 (IFCR14)[6]	R/W	0x0000_0000	on page 862
0x0088	Identifier filter control register 15 (IFCR15)[6]	R/W	0x0000_0000	on page 862
0x008C–0x000F	Reserved			

1 0000_0082h for LINFlex_0, 0000_0092h for LINFlex_1
2 0000_0E2Ch for LINFlex_0, 0000_0E1Ch for LINFlex_1
3 0006_0000h for LINFlex_0, 0004_0000h for LINFlex_1
4 LSB: Least significant byte
5 MSB: Most significant byte
6 This register is not implemented on LINFlex_1.

23.7.2 Register description

This section describes in address order all the LINFlex registers. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

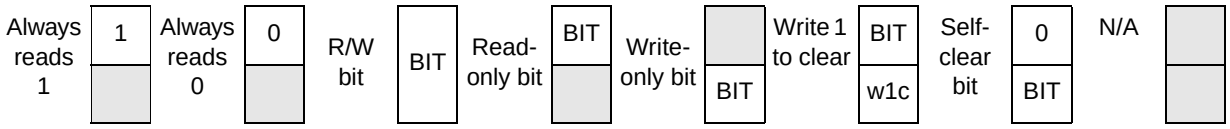


Figure 23-6. Key to register fields

NOTE

32-bit read/write access is preferred for all LINFlex registers.16-bit and 8-bit access is possible for all registers except for BDRL and BDRM. For these registers, 16- or 8-bit access is not permitted; they must be accessed in 32-bit mode only.

23.7.2.1 LIN control register 1 (LINCR1)

Address: Base + 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD	CFD	LASE	AWUM	MBL[0:3]				BF	SFTM	LBKM	MME	SBDT	RBLM	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	*	0	0	1	0

*: This field resets to 0 for LINFlex_0 and 1 for LINFlex_1.

Figure 23-7. LIN control register 1 (LINCR1)

Table 23-3. LINCR1 field descriptions

Field	Description
0:15	Reserved
CCD 16	Checksum calculation disable This bit disables the checksum calculation (see Table 23-4). 0 Checksum calculation is done by hardware. When this bit is 0, the LINCFR is read-only. 1 Checksum calculation is disabled. When this bit is set the LINCFR is read/write. User can program this register to send a software-calculated CRC (provided CFD is 0). **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
CFD 17	Checksum field disable This bit disables the checksum field transmission (see Table 23-4). 0 Checksum field is sent after the required number of data bytes is sent. 1 No checksum field is sent. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.

Table 23-3. LINCR1 field descriptions (continued)

Field	Description
LASE 18	LIN Slave Automatic Resynchronization Enable 0 Automatic resynchronization disable. 1 Automatic resynchronization enable. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
AWUM 19	Automatic Wake-Up Mode This bit controls the behavior of the LINFlex hardware during Sleep mode. 0 The Sleep mode is exited on software request by clearing the SLEEP bit of the LINCR. 1 The Sleep mode is exited automatically by hardware on LINRX dominant state detection. The SLEEP bit of the LINCR is cleared by hardware whenever WUF bit in the LINSR is set. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
MBL[0:3] 20:23	LIN Master Break Length These bits indicate the Break length in Master mode (see Table 23-5). **Note:** These bits can be written in Initialization mode only. They are read-only in Normal or Sleep mode.
BF 24	Bypass filter 0 No interrupt if identifier does not match any filter. 1 An RX interrupt is generated on identifier not matching any filter. **Note:** • If no filter is activated, this bit is reserved. • This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
SFTM 25	Self-test Mode This bit controls the Self-test mode. For more details, see Section 23.6.2, Self-test mode. 0 Self-test mode disable. 1 Self-test mode enable. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
LBKM 26	Loopback Mode This bit controls the Loopback mode. For more details see Section 23.6.1, Loopback mode. 0 Loopback mode disable. 1 Loopback mode enable. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode
MME 27	Master Mode Enable 0 Slave mode enable. 1 Master mode enable. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
SBDT 28	Slave Mode Break Detection Threshold 0 11-bit break. 1 10-bit break. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
RBLM 29	Receive Buffer Locked Mode 0 Receive Buffer not locked on overrun. Once the Slave Receive Buffer is full the next incoming message overwrites the previous one. 1 Receive Buffer locked against overrun. Once the Receive Buffer is full the next incoming message is discarded. **Note:** This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.

Table 23-3. LINCR1 field descriptions (continued)

Field	Description
SLEEP 30	Sleep Mode Request This bit is set by software to request LINFlex to enter Sleep mode. This bit is cleared by software to exit Sleep mode or by hardware if the AWUM bit in LINCR1 and the WUF bit in LINSR are set (see Table 23-6).
INIT 31	Initialization Request The software sets this bit to switch hardware into Initialization mode. If the SLEEP bit is reset, LINFlex enters Normal mode when clearing the INIT bit (see Table 23-6).

Table 23-4. Checksum bits configuration

CFD	CCD	LINCFR	Checksum sent
1	1	Read/Write	None
1	0	Read-only	None
0	1	Read/Write	Programmed in LINCFR by bits CF[0:7]
0	0	Read-only	Hardware calculated

Table 23-5. LIN master break length selection

MBL[0:3]	Length
0000	10-bit
0001	11-bit
0010	12-bit
0011	13-bit
0100	14-bit
0101	15-bit
0110	16-bit
0111	17-bit
1000	18-bit
1001	19-bit
1010	20-bit
1011	21-bit
1100	22-bit
1101	23-bit
1110	36-bit
1111	50-bit

Table 23-6. Operating mode selection

SLEEP	INIT	Operating mode
1	0	Sleep (reset value)
x	1	Initialization
0	0	Normal

23.7.2.2 LIN interrupt enable register (LINIER)

Address: Base + 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0	0	FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-8. LIN interrupt enable register (LINIER)

Table 23-7. LINIER field descriptions

Field	Description
0:15	Reserved
SZIE 16	Stuck at Zero Interrupt Enable 0 No interrupt when SZF bit in LINESR or UARTSR is set. 1 Interrupt generated when SZF bit in LINESR or UARTSR is set.
OCIE 17	Output Compare Interrupt Enable 0 No interrupt when OCF bit in LINESR or UARTSR is set. 1 Interrupt generated when OCF bit in LINESR or UARTSR is set.
BEIE 18	Bit Error Interrupt Enable 0 No interrupt when BEF bit in LINESR is set. 1 Interrupt generated when BEF bit in LINESR is set.
CEIE 19	Checksum Error Interrupt Enable 0 No interrupt on Checksum error. 1 Interrupt generated when checksum error flag (CEF) in LINESR is set.
HEIE 20	Header Error Interrupt Enable 0 No interrupt on Break Delimiter error, Synch Field error, Identifier field error. 1 Interrupt generated on Break Delimiter error, Synch Field error, Identifier field error.
21:22	Reserved

Table 23-7. LINIER field descriptions (continued)

Field	Description
FEIE 23	Framing Error Interrupt Enable 0 No interrupt on Framing error. 1 Interrupt generated on Framing error.
BOIE 24	Buffer Overrun Interrupt Enable 0 No interrupt on Buffer overrun. 1 Interrupt generated on Buffer overrun.
LSIE 25	LIN State Interrupt Enable 0 No interrupt on LIN state change. 1 Interrupt generated on LIN state change. This interrupt can be used for debugging purposes. It has no status flag but is reset when writing '1111' into LINS[0:3] in the LINSR.
WUIE 26	Wake-up Interrupt Enable 0 No interrupt when WUF bit in LINSR or UARTSR is set. 1 Interrupt generated when WUF bit in LINSR or UARTSR is set.
DBFIE 27	Data Buffer Full Interrupt Enable 0 No interrupt when buffer data register is full. 1 Interrupt generated when data buffer register is full.
DBEIE 28	Data Buffer Empty Interrupt Enable 0 No interrupt when buffer data register is empty. 1 Interrupt generated when data buffer register is empty.
DRIE 29	Data Reception Complete Interrupt Enable 0 No interrupt when data reception is completed. 1 Interrupt generated when data received flag (DRF) in LINSR or UARTSR is set.
DTIE 30	Data Transmitted Interrupt Enable 0 No interrupt when data transmission is completed. 1 Interrupt generated when data transmitted flag (DTF) is set in LINSR or UARTSR.
HRIE 31	Header Received Interrupt Enable 0 No interrupt when a valid LIN header has been received. 1 Interrupt generated when a valid LIN header has been received, that is, HRF bit in LINSR is set.

23.7.2.3 LIN status register (LINSR)

Address: Base + 0x0008 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS[0:3]				0	0	RMB	0	RBSY	RPS	WUF	DBFF	DBEF	DRF	DTF	HRF
W	w1c	w1c	w1c	w1c			w1c		w1c		w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 23-9. LIN status register (LINSR)

Table 23-8. LINSR field descriptions

Field	Description
0:15	Reserved

Table 23-8. LINSR field descriptions (continued)

Field	Description
LINS[0:3] 16:19	LIN modes / normal mode states **0000: Sleep mode** LINFlex is in Sleep mode to save power consumption. **0001: Initialization mode** LINFlex is in Initialization mode. Normal mode states **0010: Idle** This state is entered on several events: • SLEEP bit and INIT bit in LINCR1 have been cleared by software, • A falling edge has been received on RX pin and AWUM bit is set, • The previous frame reception or transmission has been completed or aborted. **0011: Break** In Slave mode, a falling edge followed by a dominant state has been detected. Receiving Break. Note: In Slave mode, in case of error new LIN state can be either Idle or Break depending on last bit state. If last bit is dominant new LIN state is Break, otherwise Idle. In Master mode, Break transmission ongoing. **0100: Break Delimiter** In Slave mode, a valid Break has been detected. Refer to Section 23.7.2.1, LIN control register 1 (LINCR1) for break length configuration (10-bit or 11-bit). Waiting for a rising edge. In Master mode, Break transmission has been completed. Break Delimiter transmission is ongoing. **0101: Synch Field** In Slave mode, a valid Break Delimiter has been detected (recessive state for at least one bit time). Receiving Synch Field. In Master mode, Synch Field transmission is ongoing. **0110: Identifier Field** In Slave mode, a valid Synch Field has been received. Receiving Identifier Field. In Master mode, identifier transmission is ongoing. **0111: Header reception/transmission completed** In Slave mode, a valid header has been received and identifier field is available in the BIDR. In Master mode, header transmission is completed. **1000: Data reception/transmission** Response reception/transmission is ongoing. **1001: Checksum** Data reception/transmission completed. Checksum reception/transmission ongoing. In UART mode, only the following states are flagged by the LIN state bits: • Init • Sleep • Idle • Data transmission/reception
20:21	Reserved
RMB 22	Release Message Buffer 0 Buffer is free. 1 Buffer ready to be read by software. This bit must be cleared by software after reading data received in the buffer. This bit is cleared by hardware in Initialization mode.
23	Reserved

Table 23-8. LINSR field descriptions (continued)

Field	Description
RBSY 24	Receiver Busy Flag 0 Receiver is idle 1 Reception ongoing **Note:** In Slave mode, after header reception, if DIR bit in BIDR is reset and reception starts then this bit is set. In this case, user cannot set DTRQ bit in LINCR2.
RPS 25	LIN receive pin state This bit reflects the current status of LINRX pin for diagnostic purposes.
WUF 26	Wake-up Flag This bit is set by hardware and indicates to the software that LINFlex has detected a falling edge on the LINRX pin when: • Slave is in Sleep mode • Master is in Sleep mode or idle state This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupt is generated if WUIE bit in LINIER is set.
DBFF 27	Data Buffer Full Flag This bit is set by hardware and indicates the buffer is full. It is set only when receiving extended frames (DFL > 7). This bit must be cleared by software. It is reset by hardware in Initialization mode.
DBEF 28	Data Buffer Empty Flag This bit is set by hardware and indicates the buffer is empty. It is set only when transmitting extended frames (DFL > 7). This bit must be cleared by software, once buffer has been filled again, in order to start transmission. This bit is reset by hardware in Initialization mode.
DRF 29	Data Reception Completed Flag This bit is set by hardware and indicates the data reception is completed. This bit must be cleared by software. It is reset by hardware in Initialization mode. **Note:** This flag is not set in case of bit error or framing error.
DTF 30	Data Transmission Completed Flag This bit is set by hardware and indicates the data transmission is completed. This bit must be cleared by software. It is reset by hardware in Initialization mode. **Note:** This flag is not set in case of bit error if IOBE bit is reset.
HRF 31	Header Reception Flag This bit is set by hardware and indicates a valid header reception is completed. This bit must be cleared by software. This bit is reset by hardware in Initialization mode and at end of completed or aborted frame. Note: If filters are enabled, this bit is set only when identifier software filtering is required, that is to say: • All filters are inactive and BF bit in LINCR1 is set • No match in any filter and BF bit in LINCR1 is set • TX filter match

23.7.2.4 LIN error status register (LINESR)

Address: Base + 0x000C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	BEF	CEF	SFEF	BDEF	IDPEF	FEF	BOF	0	0	0	0	0	0	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-10. LIN error status register (LINESR)

Table 23-9. LINESR field descriptions

Field	Description
0:15	Reserved
SZF 16	Stuck at Zero Flag This bit is set by hardware when the bus is dominant for more than a 100-bit time. If the dominant state continues, SZF flag is set again after 87-bit time. It is cleared by software.
OCF 17	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of OC1[0:7] or OC2[0:7] in LINOCR. If this bit is set and IOT bit in LINTCSR is set, LINFlex moves to Idle state. If LTOM bit in LINTCSR is set, then OCF is cleared by hardware in Initialization mode. If LTOM bit is cleared, then OCF maintains its status whatever the mode is.
BEF 18	Bit Error Flag This bit is set by hardware and indicates to the software that LINFlex has detected a bit error. This error can occur during response field transmission (Slave and Master modes) or during header transmission (in Master mode). This bit is cleared by software.
CEF 19	Checksum Error Flag This bit is set by hardware and indicates that the received checksum does not match the hardware calculated checksum. This bit is cleared by software. **Note:** This bit is never set if CCD or CFD bit in LINCR1 is set.
SFEF 20	Synch Field Error Flag This bit is set by hardware and indicates that a Synch Field error occurred (inconsistent Synch Field).
BDEF 21	Break Delimiter Error Flag This bit is set by hardware and indicates that the received Break Delimiter is too short (less than one bit time).

Table 23-9. LINESR field descriptions (continued)

Field	Description
IDPEF 22	Identifier Parity Error Flag This bit is set by hardware and indicates that a Identifier Parity error occurred. **Note:** Header interrupt is triggered when SFEF or BDEF or IDPEF bit is set and HEIE bit in LINIER is set.
FEF 23	Framing Error Flag This bit is set by hardware and indicates to the software that LINFlex has detected a framing error (invalid stop bit). This error can occur during reception of any data in the response field (Master or Slave mode) or during reception of Synch Field or Identifier Field in Slave mode.
BOF 24	Buffer Overrun Flag This bit is set by hardware when a new data byte is received and the buffer full flag is not cleared. If RBLM in LINCR1 is set then the new byte received is discarded. If RBLM is reset then the new byte overwrites the buffer. It can be cleared by software.
25:30	Reserved
NF 31	Noise Flag This bit is set by hardware when noise is detected on a received character. This bit is cleared by software.

23.7.2.5 UART mode control register (UARTCR)

Address: Base + 0x0010 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TDFL[0:1]		0	RDFL[0:1]		0	0	0	0	RXEN	TXEN	OP	PCE	WL	UART
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-11. UART mode control register (UARTCR)

Table 23-10. UARTCR field descriptions

Field	Description
0:16	Reserved
TDFL[0:1] 17:18	Transmitter Data Field length These bits set the number of bytes to be transmitted in UART mode. These bits can be programmed only when the UART bit is set. TDFL[0:1] = Transmit buffer size – 1. 00 Transmit buffer size = 1. 01 Transmit buffer size = 2. 10 Transmit buffer size = 3. 11 Transmit buffer size = 4.

Table 23-10. UARTCR field descriptions (continued)

Field	Description
19	Reserved
RDFL[0:1] 20:21	Receiver Data Field length These bits set the number of bytes to be received in UART mode. These bits can be programmed only when the UART bit is set. RDFL[0:1] = Receive buffer size – 1. 00 Receive buffer size = 1. 01 Receive buffer size = 2. 10 Receive buffer size = 3. 11 Receive buffer size = 4.
22:25	Reserved
RXEN 26	Receiver Enable 0 Receiver disable. 1 Receiver enable. This bit can be programmed only when the UART bit is set.
TXEN 27	Transmitter Enable 0 Transmitter disable. 1 Transmitter enable. This bit can be programmed only when the UART bit is set. **Note:** Transmission starts when this bit is set and when writing DATA0 in the BDRL register.
OP 28	Odd Parity 0 Sent parity is even. 1 Sent parity is odd. This bit can be programmed in Initialization mode only when the UART bit is set.
PCE 29	Parity Control Enable 0 Parity transmit/check disable. 1 Parity transmit/check enable. This bit can be programmed in Initialization mode only when the UART bit is set.
WL 30	Word Length in UART mode 0 7-bit data + parity bit. 1 8-bit data (or 9-bit if PCE is set). This bit can be programmed in Initialization mode only when the UART bit is set.
UART 31	UART mode enable 0 LIN mode. 1 UART mode. This bit can be programmed in Initialization mode only.

23.7.2.6 UART mode status register (UARTSR)

Address: Base + 0x0014 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RPS	WUF	0	0	DRF	DTF	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-12. UART mode status register (UARTSR)

Table 23-11. UARTSR field descriptions

Field	Description
0:15	Reserved
SZF 16	Stuck at Zero Flag This bit is set by hardware when the bus is dominant for more than a 100-bit time. It is cleared by software.
OCF 17	OCF Output Compare Flag 0 No output compare event occurred. 1 The content of the counter has matched the content of OC1[0:7] or OC2[0:7] in LINOCR. An interrupt is generated if the OCIE bit in LINIER register is set.
PE3 18	Parity Error Flag Rx3 This bit indicates if there is a parity error in the corresponding received byte (Rx3). See Section 23.8.1.1, Buffer in UART mode. No interrupt is generated if this error occurs. 0 No parity error. 1 Parity error.
PE2 19	Parity Error Flag Rx2 This bit indicates if there is a parity error in the corresponding received byte (Rx2). See Section 23.8.1.1, Buffer in UART mode. No interrupt is generated if this error occurs. 0 No parity error. 1 Parity error.
PE1 20	Parity Error Flag Rx1 This bit indicates if there is a parity error in the corresponding received byte (Rx1). See Section 23.8.1.1, Buffer in UART mode. No interrupt is generated if this error occurs. 0 No parity error. 1 Parity error.
PE0 21	Parity Error Flag Rx0 This bit indicates if there is a parity error in the corresponding received byte (Rx0). See Section 23.8.1.1, Buffer in UART mode. No interrupt is generated if this error occurs. 0 No parity error. 1 Parity error.

Table 23-11. UARTSR field descriptions (continued)

Field	Description
RMB 22	Release Message Buffer 0 Buffer is free. 1 Buffer ready to be read by software. This bit must be cleared by software after reading data received in the buffer. This bit is cleared by hardware in Initialization mode.
FEF 23	Framing Error Flag This bit is set by hardware and indicates to the software that LINFlex has detected a framing error (invalid stop bit).
BOF 24	Buffer Overrun Flag This bit is set by hardware when a new data byte is received and the buffer full flag is not cleared. If RBLM in LINCR1 is set then the new byte received is discarded. If RBLM is reset then the new byte overwrites buffer. it can be cleared by software.
RPS 25	LIN Receive Pin State This bit reflects the current status of LINRX pin for diagnostic purposes.
WUF 26	Wake-up Flag This bit is set by hardware and indicates to the software that LINFlex has detected a falling edge on the LINRX pin in Sleep mode. This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupt i generated if WUIE bit in LINIER is set.
27:28	Reserved
DRF 29	Data Reception Completed Flag This bit is set by hardware and indicates the data reception is completed, that is, the number of bytes programmed in RDFL[0:1] in UARTCR have been received. This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupt is generated if DRIE bit in LINIER is set. Note: In UART mode, this flag is set in case of framing error, parity error or overrun.
DTF 30	Data Transmission Completed Flag This bit is set by hardware and indicates the data transmission is completed, that is, the number of bytes programmed in TDFL[0:1] have been transmitted. This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupt is generated if DTIE bit in LINIER is set.
NF 31	Noise Flag This bit is set by hardware when noise is detected on a received character. This bit is cleared by software.

23.7.2.7 LIN timeout control status register (LINTCSR)

Address: Base + 0x0018 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	LTOM	IOT	TOCE	CNT[0:7]							
W																
Reset	0	0	0	0	0	0		0	0	0		0	0	0	0	0

Figure 23-13. LIN timeout control status register (LINTCSR)

Table 23-12. LINTCSR field descriptions

Field	Description
0:20	Reserved
LTOM 21	LIN timeout mode 0 LIN timeout mode (header, response and frame timeout detection). 1 Output compare mode. This bit can be set/cleared in Initialization mode only.
IOT 22	Idle on Timeout 0 LIN state machine not reset to Idle on timeout event. 1 LIN state machine reset to Idle on timeout event. This bit can be set/cleared in Initialization mode only.
TOCE 23	Timeout counter enable 0 Timeout counter disable. OCF bit in LINESR or UARTSR is not set on an output compare event. 1 Timeout counter enable. OCF bit is set if an output compare event occurs. TOCE bit is configurable by software in Initialization mode. If LIN state is not Init and if timer is in LIN timeout mode, then hardware takes control of TOCE bit.
CNT[0:7] 24:31	Counter Value These bits indicate the LIN Timeout counter value.

23.7.2.8 LIN output compare register (LINOCR)

Address: Base + 0x001C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OC2[0:7][1]								OC1[0:7][1]							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

[1] If LTOM in the LINTCSR register is set, these bits are read-only.

Figure 23-14. LIN output compare register (LINOCR)

Table 23-13. LINOCR field descriptions

Field	Description
0:15	Reserved
OC2[0:7] 16:23	Output compare 2 value These bits contain the value to be compared to the value of bits CNT[0:7] in LINTCSR.
OC1[0:7] 24:31	Output compare 1 value These bits contain the value to be compared to the value of bits CNT[0:7] in LINTCSR.

23.7.2.9 LIN timeout control register (LINTOCR)

Address: Base + 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RTO[0:3]				0	HTO[0:6]						
W																
Reset	0	0	0	0	1	1	1	0	0	0	*	*	1	1	0	0

*: These bits reset to 10 for LINFlex_0 and 01 for LINFlex_1.

Figure 23-15. LIN timeout control register (LINTOCR)

Table 23-14. LINTOCR field descriptions

Field	Description
0:19	Reserved
RTO[0:3] 20:23	Response timeout value This register contains the response timeout duration (in bit time) for 1 byte. The reset value is 0xE = 14, corresponding to $T_{Response\_Maximum} = 1.4 \times T_{Response\_Nominal}$
24	Reserved
HTO[0:6] 25:31	Header timeout value This register contains the header timeout duration (in bit time). This value does not include the Break and the Break Delimiter. The reset value is the 0x2C = 44, corresponding to $T_{Header\_Maximum}$. Setting MME bit in LINSR changes HTO[0:6] value to 0x1C = 28. HTO[0:6] can be written only in Slave mode.

23.7.2.10 LIN fractional baud rate register (LINFBRR)

Address: Base + 0x0024 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	DIV_F[0:3]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-16. LIN fractional baud rate register (LINFBRR)

Table 23-15. LINFBRR field descriptions

Field	Description
0:27	Reserved
DIV_F[0:3] 28:31	Fraction bits of LFDIV The 4 fraction bits define the value of the fraction of the LINFlex divider (LFDIV). Fraction (LFDIV) = Decimal value of DIV_F [0:3] / 16. This register can be written in Initialization mode only.

23.7.2.11 LIN integer baud rate register (LINIBRR)

Address: Base + 0x0028 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	DIV_M[0:12]												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-17. LIN integer baud rate register (LINIBRR)

Table 23-16. LINIBRR field descriptions

Field	Description
0:18	Reserved
DIV_M[0:12] 19:31	LFDIV mantissa These 12 bits define the LINFlex divider (LFDIV) mantissa value (see Table 23-17). This register can be written in Initialization mode only.

Table 23-17. Integer baud rate selection

DIV_M[0:12]	Mantissa
0x0000	LIN clock disabled
0x0001	1
...	...
0x1FFE	8190
ox1FFF	8191

23.7.2.12 LIN checksum field register (LINCFR)

Address: Base + 0x002C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CF[0:7]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-18. LIN checksum field register (LINCFR)

Table 23-18. LINCFR field descriptions

Field	Description
0:23	Reserved
CF[0:7] 24:31	Checksum bits When CCD bit in LINCR1 is cleared these bits are read-only. When CCD bit is set, these bits are read/write. See Table 23-4.

23.7.2.13 LIN control register 2 (LINCR2)

Address: Base + 0x0030 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	IOBE	IOPE						0	0	0	0	0	0	0	0
W				WURQ	DDRQ	DTRQ	ABRQ	HTRQ								
Reset	0	1	*	0	0	0	0	0	0	0	0	0	0	0	0	0

*: This bit resets to 1 for LINFlex_0 and 0 for LINFlex_1.

Figure 23-19. LIN control register 2 (LINCR2)

Table 23-19. LINCR2 field descriptions

Field	Description
0:16	Reserved
IOBE 17	Idle on Bit Error 0 Bit error does not reset LIN state machine. 1 Bit error reset LIN state machine. This bit can be set/cleared in Initialization mode only.
IOPE 18	Idle on Identifier Parity Error 0 Identifier Parity error does not reset LIN state machine. 1 Identifier Parity error reset LIN state machine. This bit can be set/cleared in Initialization mode only.
WURQ 19	Wake-up Generation Request Setting this bit generates a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent is copied from DATA0 in BDRL buffer. Note that this bit cannot be set in Sleep mode. Software has to exit Sleep mode before requesting a wakeup. Bit error is not checked when transmitting the wakeup request.
DDRQ 20	Data Discard Request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlex has moved to idle state. In Slave mode, this bit can be set only when HRF bit in LINSR is set and identifier did not match any filter.
DTRQ 21	Data Transmission Request Set by software in Slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when HRF bit in LINSR is set. Cleared by hardware when the request has been completed or aborted or on an error condition. In Master mode, this bit is set by hardware when DIR bit in BIDR is set and header transmission is completed.
ABRQ 22	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlex aborts the transmission at the end of the current bit. This bit can also abort a wakeup request. It can also be used in UART mode.
HTRQ 23	Header Transmission Request Set by software to request the transmission of the LIN header. Cleared by hardware when the request has been completed or aborted. This bit has no effect in UART mode.
24:31	Reserved

23.7.2.14 Buffer identifier register (BIDR)

Address: Base + 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL[0:5]						DIR	CCS	0	0	ID[0:5]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-20. Buffer identifier register (BIDR)

Table 23-20. BIDR field descriptions

Field	Description
0:15	Reserved
DFL[0:5] 16:21	Data Field Length These bits define the number of data bytes in the response part of the frame. DFL[0:5] = Number of data bytes – 1. Normally, LIN uses only DFL[0:2] to manage frames with a maximum of 8 bytes of data. Identifier filters are compatible with DFL[0:2] only. DFL[3:5] are provided to manage extended frames.
DIR 22	Direction This bit controls the direction of the data field. 0 LINFlex receives the data and copies them in the BDR registers. 1 LINFlex transmits the data from the BDR registers.
CCS 23	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and earlier. In LIN Slave mode (MME bit cleared in LINCR1), this bit must be configured before the header reception. If the slave has to manage frames with 2 types of checksum, filters must be configured.
24:25	Reserved
ID[0:5] 26:31	Identifier Identifier part of the identifier field without the identifier parity.

23.7.2.15 Buffer data register LSB (BDRL)

Address: Base + 0x0038 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA3[0:7]								DATA2[0:7]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA1[0:7]								DATA0[0:7]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-21. Buffer data register LSB (BDRL)

Table 23-21. BDRL field descriptions

Field	Description
DATA3[0:7] 0:7	Data Byte 3 Data byte 3 of the data field.
DATA2[0:7] 8:15	Data Byte 2 Data byte 2 of the data field.
DATA1[0:7] 16:23	Data Byte 1 Data byte 1 of the data field.
DATA0[0:7] 24:31	Data Byte 0 Data byte 0 of the data field.

23.7.2.16 Buffer data register MSB (BDRM)

Address: Base + 0x003C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA7[0:7]								DATA6[0:7]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA5[0:7]								DATA4[0:7]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-22. Buffer data register MSB (BDRM)

Table 23-22. BDRM field descriptions

Field	Description
DATA7[0:7] 0:7	Data Byte 7 Data byte 7 of the data field.
DATA6[0:7] 8:15	Data Byte 6 Data byte 6 of the data field.
DATA5[0:7] 16:23	Data Byte 5 Data byte 5 of the data field.
DATA4[0:7] 24:31	Data Byte 4 Data byte 4 of the data field.

23.7.2.17 Identifier filter enable register (IFER)

This register is not implemented on LINFlex_1.

Address: Base + 0x0040 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FACT[0:7]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-23. Identifier filter enable register (IFER)

Table 23-23. IFER field descriptions

Field	Description
0:23	Reserved
FACT[0:7] 24:31	Filter activation 0 Filters 2*n* and 2*n* + 1 are activated. 1 Filters 2*n* and 2*n* + 1 are deactivated. (Refer to Table 23-24.) These bits can be set/cleared in Initialization mode only.

Table 23-24. IFER[FACT] configuration

Bit	Value	Result
FACT[0]	0	Filters 0 and 1 are deactivated.
	1	Filters 0 and 1 are activated.

Table 23-24. IFER[FACT] configuration (continued)

Bit	Value	Result
FACT[1]	0	Filters 2 and 3 are deactivated.
	1	Filters 2 and 3 are activated.
FACT[2]	0	Filters 4 and 5 are deactivated.
	1	Filters 4 and 5 are activated.
FACT[3]	0	Filters 6 and 7 are deactivated.
	1	Filters 6 and 7 are activated.
FACT[4]	0	Filters 8 and 9 are deactivated.
	1	Filters 8 and 9 are activated.
FACT[5]	0	Filters 10 and 11 are deactivated.
	1	Filters 10 and 11 are activated.
FACT[6]	0	Filters 12 and 13 are deactivated.
	1	Filters 12 and 13 are activated.
FACT[7]	0	Filters 14 and 15 are deactivated.
	1	Filters 14 and 15 are activated.

23.7.2.18 Identifier filter match index (IFMI)

This register is not implemented on LINFlex_1.

Address: Base + 0x0044 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	IFMI[0:4]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-24. Identifier filter match index (IFMI)

Table 23-25. IFMI field descriptions

Field	Description
0:26	Reserved

Table 23-25. IFMI field descriptions

Field	Description
IFMI[0:4] 27:31	Filter match index This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Section 23.8.2.2, Slave mode for more details). When no filter matches, IFMI[0:4] = 0. When Filter *n* is matching, IFMI[0:4] = *n* + 1.

23.7.2.19 Identifier filter mode register (IFMR)

This register is not implemented on LINFlex_1.

Address: Base + 0x0048 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	IFM			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-25. Identifier filter mode register (IFMR)

Table 23-26. IFMR field descriptions

Field	Description
IFM	Filter mode 0 Filters 2*n* and 2*n* + 1 are in identifier list mode. 1 Filters 2*n* and 2*n* + 1 are in mask mode (filter 2*n* + 1 is the mask for the filter 2*n*). (Refer to Table 23-27.)

Table 23-27. IFMR[IFM] configuration

Bit	Value	Result
IFM[0]	0	Filters 0 and 1 are in identifier list mode.
	1	Filters 0 and 1 are in mask mode (filter 1 is the mask for the filter 0).
IFM[1]	0	Filters 2 and 3 are in identifier list mode.
	1	Filters 2 and 3 are in mask mode (filter 3 is the mask for the filter 2).
IFM[2]	0	Filters 4 and 5 are in identifier list mode.
	1	Filters 4 and 5 are in mask mode (filter 5 is the mask for the filter 4).
IFM[3]	0	Filters 6 and 7 are in identifier list mode.
	1	Filters 6 and 7 are in mask mode (filter 7 is the mask for the filter 6).

23.7.2.20 Identifier filter control register (IFCR2*n*)

This register is not implemented on LINFlex_1.

n = 0: Address: Base + 0x004C
n = 1: Address: Base + 0x0054
n = 2: Address: Base + 0x005C
n = 3: Address: Base + 0x0064
n = 4: Address: Base + 0x006C
n = 5: Address: Base + 0x0074
n = 6: Address: Base + 0x007C
n = 7: Address: Base + 0x0084

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	DFL[0:2]			DIR	CCS	0	0	ID[0:5]					
W											w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-26. Identifier filter control register (IFCR2*n*)

NOTE

This register can be written in Initialization mode only.

Table 23-28. IFCR2*n* field descriptions

Field	Description
0:18	Reserved
DFL[0:2] 19:21	Data Field Length These bits define the number of data bytes in the response part of the frame.
DIR 22	Direction This bit controls the direction of the data field. 0 LINFlex receives the data and copies them in the BDRL and BDRM registers. 1 LINFlex transmits the data from the BDRL and BDRM registers.
CCS 23	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and earlier.
24:25	Reserved
ID[0:5] 26:31	Identifier Identifier part of the identifier field without the identifier parity.

23.7.2.21 Identifier filter control register (IFCR2*n* + 1)

This register is not implemented on LINFlex_1.

n = 0: Address: Base + 0x0050
n = 1: Address: Base + 0x0058
n = 2: Address: Base + 0x0060
n = 3: Address: Base + 0x0068
n = 4: Address: Base + 0x0070
n = 5: Address: Base + 0x0078
n = 6: Address: Base + 0x0080
n = 7: Address: Base + 0x0088

Access: User read/write

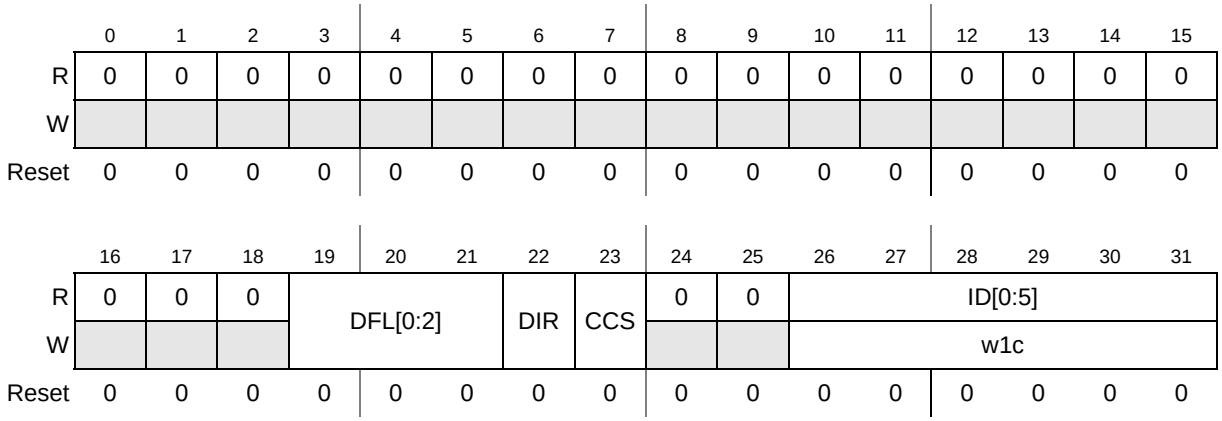


Figure 23-27. Identifier filter control register (IFCR2*n* + 1)

NOTE

This register can be written in Initialization mode only.

Table 23-29. IFCR2*n* + 1 field descriptions

Field	Description
0:18	Reserved
DFL[0:2] 19:21	Data Field Length These bits define the number of data bytes in the response part of the frame. DFL[0:2] = Number of data bytes – 1.
DIR 22	Direction This bit controls the direction of the data field. 0 LINFlex receives the data and copies them in the BDRL and BDRM registers. 1 LINFlex transmits the data from the BDRL and BDRM registers.
CCS 23	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and earlier.
24:25	Reserved
ID[0:5] 26:31	Identifier Identifier part of the identifier field without the identifier parity

23.7.3 Register map and reset values

Table 23-30. Register map and reset values

Address offset	Register name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	LINCR1 Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		CCD 0	CFD 0	LASE 0	AWUM 0	MBL0 0	MBL1 0	MBL2 0	MBL3 0	BF 1	SLFM 0	LBKM 0	MME 0	SBDT 0	RBLM 0	SLEEP 1	INIT 0
4	LINIER Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		SZIE 0	OCIE 0	BEIE 0	CEIE 0	HEIE 0	0	0	FEIE 0	BOIE 0	LSIE 0	WUIE 0	DBFIE 0	DBEIE 0	DRIE 0	DTIE 0	HRIE 0
8	LINSR Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		LINS0 0	LINS1 0	LINS2 0	LINS3 0	0	0	RMB 0	0	RBSY 0	RPS 1	WUF 0	DBFF 0	DBEF 0	DRF 0	DTF 0	HRF 0
C	LINESR Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		SZF 0	OCF 0	BEF 0	CEF 0	SFEF 0	BDEF 0	IDPEF 0	FEF 0	BOF 0	0	0	0	0	0	0	NF 0
10	UARTCR Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	TDFL0 0	TDFL1 0	0	RDFL0 0	RDFL1 0	0	0	0	0	RXEN 0	TXEN 0	OP 0	PCE 0	WL 0	UART 0
14	UARTSR Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		SZF 0	OCF 0	PE0 0	PE1 0	PE2 0	PE3 0	RMB 0	FEF 0	BOF 0	RPS 0	WUF 0	0	0	DRF 0	DTF 0	NF 0

Table 23-30. Register map and reset values (continued)

Address offset	Register name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
18	LINTCSR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	LTOM 0	IOT 1	TOCE 0	CNT0 0	CNT1 0	CNT2 0	CNT3 0	CNT4 0	CNT5 0	CNT6 0	CNT7 0
1C	LINOCR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		OC20 1	OC21 1	OC22 1	OC23 1	OC24 1	OC25 1	OC26 1	OC27 1	OC10 1	OC11 1	OC12 1	OC13 1	OC14 1	OC15 1	OC16 1	OC17 1
20	LINTOCR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	RTO0 1	RTO1 1	RTO2 1	RTO3 0	0	HTO0 0	HTO1 1	HTO2 0	HTO3 1	HTO4 1	HTO5 0	HTO6 0
24	LINFBRR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	DIV_F0 0	DIV_F1 0	DIV_F2 0	DIV_F3 0
28	LINIBRR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	DIV_M0 0	DIV_M1 0	DIV_M2 0	DIV_M3 0	DIV_M4 0	DIV_M5 0	DIV_M6 0	DIV_M7 0	DIV_M8 0	DIV_M9 0	DIV_M10 0	DIV_M11 0	DIV_M12 0
2C	LINCFR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	CF0 0	CF1 0	CF2 0	CF3 0	CF4 0	CF5 0	CF6 0	CF7 0
30	LINCR2																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	IOBE 1	IOPE 1	WURQ 0	DDRQ 0	DTRQ 0	ABRQ 0	HTRQ 0	0	0	0	0	0	0	0	0

Table 23-30. Register map and reset values (continued)

Address offset	Register name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
34	BIDR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		DFL0	DFL1	DFL2	DFL3	DFL4	DFL5	DIR	CCS			ID0	ID1	ID2	ID3	ID4	ID5
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	BDRL																
	Reset value	DATA30	DATA31	DATA32	DATA33	DATA34	DATA35	DATA36	DATA37	DATA20	DATA21	DATA22	DATA23	DATA24	DATA25	DATA26	DATA27
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		DATA10	DATA11	DATA12	DATA13	DATA14	DATA15	DATA16	DATA17	DATA00	DATA01	DATA02	DATA03	DATA04	DATA05	DATA06	DATA07
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3C	BDRM																
	Reset value	DATA70	DATA71	DATA72	DATA73	DATA74	DATA75	DATA76	DATA77	DATA60	DATA61	DATA62	DATA63	DATA64	DATA65	DATA66	DATA67
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		DATA50	DATA51	DATA52	DATA53	DATA54	DATA55	DATA56	DATA57	DATA40	DATA41	DATA42	DATA43	DATA44	DATA45	DATA46	DATA47
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	IFER																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	FACT0	FACT1	FACT2	FACT3	FACT4	FACT5	FACT6	FACT7
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	IFMI																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	IFMI0	IFMI1	IFMI2	IFMI3	IFMI4
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	IFMR																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	IFM0	IFM1	IFM2	IFM3
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-30. Register map and reset values (continued)

Address offset	Register name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4C 54 5C 64 6C 74 7C 84	IFCR2*n* Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	DFL0 0	DFL1 0	DFL2 0	DIR 0	CCS 0	0	0	ID0 0	ID1 0	ID2 0	ID3 0	ID4 0	ID5 0
50 58 60 68 70 78 80 88	IFCR2*n* + 1 Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	DFL0 0	DFL1 0	DFL2 0	DIR 0	CCS 0	0	0	ID0 0	ID1 0	ID2 0	ID3 0	ID4 0	ID5 0

23.8 Functional description

23.8.1 UART mode

The main features in the UART mode are

- Full duplex communication
- 8- or 9-bit data with parity
- 4-byte buffer for reception, 4-byte buffer for transmission
- 8-bit counter for timeout management

8-bit data frames: The 8th bit can be a data or a parity bit. Even/Odd Parity can be selected by the Odd Parity bit in the UARTCR. An even parity is set if the modulo-2 sum of the 7 data bits is 1. An odd parity is cleared in this case.

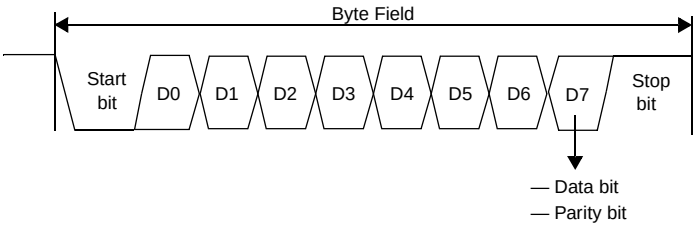


Figure 23-28. UART mode 8-bit data frame

9-bit frames: The 9th bit is a parity bit. Even/Odd Parity can be selected by the Odd Parity bit in the UARTCR. An even parity is set if the modulo-2 sum of the 7 data bits is 1. An odd parity is cleared in this case.

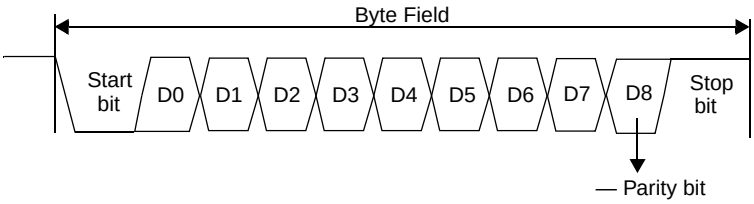


Figure 23-29. UART mode 9-bit data frame

23.8.1.1 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter as shown in Table 23-31.

Table 23-31. Message buffer

Buffer data register	LIN mode		UART mode	
BDRL[0:31]	Transmit/Receive buffer	DATA0[0:7]	Transmit buffer	Tx0
		DATA1[0:7]		Tx1
		DATA2[0:7]		Tx2
		DATA3[0:7]		Tx3
BDRM[0:31]		DATA4[0:7]	Receive buffer	Rx0
		DATA5[0:7]		Rx1
		DATA6[0:7]		Rx2
		DATA7[0:7]		Rx3

23.8.1.2 UART transmitter

In order to start transmission in UART mode, the UART bit and the transmitter enable (TXEN) bit in the UARTCR must be set. Transmission starts when DATA0 (least significant data byte) is programmed. The number of bytes transmitted is equal to the value configured by the TDFL[0:1] bits in the UARTCR (see Table 23-10).

The Transmit buffer is 4 bytes, hence a 4-byte maximum transmission can be triggered. Once the programmed number of bytes has been transmitted, the DTF bit is set in UARTSR. If the TXEN bit of UART is reset during a transmission then the current transmission is completed and no further transmission can be invoked.

23.8.1.3 UART receiver

The UART receiver is active as soon as the user exits Initialization mode and sets the RXEN bit in the UARTCR. There is a dedicated 4-byte data buffer for received data bytes. Once the programmed number (RDFL bits) of bytes has been received, the DRF bit is set in UARTSR. If the RXEN bit is reset during a reception then the current reception is completed and no further reception can be invoked until RXEN is set.

If a parity error occurs during reception of any byte, then the corresponding PE*x* bit in the UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if the FEIE bit in the LINIER is set.

If the last received frame has not been read from the buffer (that is, RMB bit is not reset by the user) then upon reception of the next byte an overrun error occurs (BOF bit in UARTSR is set) and one message will be lost. Which message is lost depends on the configuration of the RBLM bit of LINCR1.

- If the buffer lock function is disabled (RBLM bit in LINCR1 cleared) the last message stored in the buffer is overwritten by the new incoming message. In this case the latest message is always available to the application.
- If the buffer lock function is enabled (RBLM bit in LINCR1 set) the most recent message is discarded and the previous message is available in the buffer.

An interrupt is generated if the BOIE bit in the LINIER is set.

23.8.1.4 Clock gating

The LINFlex clock can be gated from the Mode Entry module (refer to Chapter 25, Mode Entry Module (MC_ME)). In UART mode, the LINFlex controller acknowledges a clock gating request once the data transmission and data reception are completed, that is, once the Transmit buffer is empty and the Receive buffer is full.

23.8.2 LIN mode

LIN mode comprises four submodes:

- Master mode
- Slave mode
- Slave mode with identifier filtering
- Slave mode with automatic resynchronization

These submodes are described in the following pages.

23.8.2.1 Master mode

In Master mode the application uses the message buffer to handle the LIN messages. Master mode is selected when the MME bit in LINCR1 is set.

23.8.2.1.1 LIN header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the Master sending a header. The header is transmitted by the Master task while the data is transmitted by the Slave task of a node.

To transmit a header with LINFlex the application must set up the identifier, the data field length and configure the message (direction and checksum type) in the BIDR before requesting the header transmission by setting the HTRQ bit in LINCR2.

23.8.2.1.2 Data transmission (transceiver as publisher)

When the master node is publisher of the data corresponding to the identifier sent in the header, then the slave task of the master has to send the data in the Response part of the LIN frame. Therefore, the application must provide the data to LINFlex before requesting the header transmission. The application stores the data in the message buffer BDR. According to the data field length, LINFlex transmits the data and the checksum. The application uses the CCS bit in the BIDR to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, the DTF bit in the LINSR is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to Section 23.8.2.1.6, Error handling).

It is possible to handle frames with a Response size larger than 8 bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than 8 data bytes, the DBEF bit in the LINSR is set once the first 8 bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset.

Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is controlled by the DIR bit in the BIDR. When the application sets this bit the response is sent by LINFlex (publisher). Resetting this bit configures the message buffer as subscriber.

23.8.2.1.3 Data reception (transceiver as subscriber)

To receive data from a slave node, the master sends a header with the corresponding identifier. LINFlex stores the data received from the slave in the message buffer and stores the message status in the LINSR.

If the response has been received successfully, the DRF bit in the LINSR is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to Section 23.8.2.1.6, Error handling).

It is possible to handle frames with a Response size larger than 8 bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than 8 data bytes, the DBFF bit in the LINSR is set once the first 8 bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

23.8.2.1.4 Data discard

To discard data from a slave, the DIR bit in the BIDR must be reset and the DDRQ bit in LINCR2 must be set before starting the header transmission.

23.8.2.1.5 Error detection

LINFlex is able to detect and handle LIN communication errors. A code stored in the LIN error status register (LINESR) signals the errors to the software.

In Master mode, the following errors are detected:

- **Bit error**: During transmission, the value read back from the bus differs from the transmitted value.
- **Framing error**: A dominant state has been sampled on the stop bit of the currently received character (synch field, identifier field or data field).
- **Checksum error**: The computed checksum does not match the received one.
- **Response and Frame timeout**: Refer to Section 23.8.3, 8-bit timeout counter for more details.

23.8.2.1.6 Error handling

In case of Bit Error detection during transmission, LINFlex stops the transmission of the frame after the corrupted bit. LINFlex returns to idle state and an interrupt is generated if the BEIE bit in the LINIER is set.

During reception, a Framing Error leads LINFlex to discard the current frame. LINFlex returns immediately to idle state. An interrupt is generated if the FEIE bit in the LINIER is set.

During reception, a Checksum Error leads LINFlex to discard the received frame. LINFlex returns to idle state. An interrupt is generated if the CEIE bit in the LINIER is set.

23.8.2.1.7 Overrun

Once the message buffer is full, the next valid message reception leads to an overrun and a message is lost. The hardware sets the BOF bit in the LINSR to signal the overrun condition. Which message is lost depends on the configuration of the RX message buffer:

- If the buffer lock function is disabled (LINCR1[RBLM] = 0) the last message stored in the buffer is overwritten by the new incoming message. In this case the latest message is always available to the application.
- If the buffer lock function is enabled (LINCR1[RBLM] = 0) the most recent message is discarded and the previous message is available in the buffer.

23.8.2.2 Slave mode

In Slave mode the application uses the message buffer to handle the LIN messages. Slave mode is selected when the MME bit in LINCR1 is cleared.

23.8.2.2.1 Data transmission (transceiver as publisher)

When LINFlex receives the identifier, the HRF bit in the LINSR is set and, if the HRIE bit in the LINIER is set, an RX interrupt is generated. The software must read the received identifier in the BIDR, fill the BDR registers, specify the data field length using the DFL[0:2] bits in the BIDR and trigger the data transmission by setting the DTRQ bit in LINCR2.

One or several identifier filters can be configured for transmission by setting the DIR bit in the IFCR*x* register(s) and activated by setting one or several bits in the IFER.

When at least one identifier filter is configured in transmission and activated, and if the received identifier matches the filter, a specific TX interrupt (instead of an RX interrupt) is generated.

Typically, the application has to copy the data from SRAM locations to the BDAR. To copy the data to the right location, the application has to identify the data by means of the identifier. To avoid this and to ease the access to the SRAM locations, the LINFlex controller provides a Filter Match Index. This index value is the number of the filter that matched the received identifier.

The software can use the index in the IFMI register to directly access the pointer that points to the right data array in the SRAM area and copy this data to the BDAR (see Figure 23-31).

Using a filter avoids the software having to configure the direction, the data field length and the checksum type in the BDIR. The software fills the BDAR and triggers the data transmission by setting the DTRQ bit in LINCR2.

If LINFlex cannot provide enough TX identifier filters to handle all identifiers the software has to transmit data for, then a filter can be configured in mask mode (refer to Section 23.8.2.3, Slave mode with identifier filtering") in order to manage several identifiers with one filter only.

23.8.2.2.2 Data reception (transceiver as subscriber)

When LINFlex receives the identifier, the HRF bit in the LINSR is set and, if the HRIE bit in the LINIER is set, an RX interrupt is generated. The software must read the received identifier in the BIDR and specify the data field length using the DFL[0:2] bits in the BDIR before receiving the stop bit of the first byte of data field.

When the checksum reception is completed, an RX interrupt is generated to allow the software to read the received data in the BDR registers.

One or several identifier filters can be configured for reception by resetting the DIR bit in the IFCR*x* register(s) and activated by setting one or several bits in the IFER.

When at least one identifier filter is configured in reception and activated, and if the received identifier matches the filter, an RX interrupt is generated after the checksum reception only.

Typically, the application has to copy the data from the BDAR to SRAM locations. To copy the data to the right location, the application has to identify the data by means of the identifier. To avoid this and to ease the access to the SRAM locations, the LINFlex controller provides a Filter Match Index. This index value is the number of the filter that matched the received identifier.

The software can use the index in the IFMI register to directly access the pointer that points to the right data array in the SRAM area and copy this data from the BDAR to the SRAM (see Figure 23-31).

Using a filter avoids the software reading the ID value in the BIDR, and configuring the direction, the data field length and the checksum type in the BDIR.

If LINFlex cannot provide enough RX identifier filters to handle all identifiers the software has to receive the data for, then a filter can be configured in mask mode (refer to Section 23.8.2.3, Slave mode with identifier filtering") in order to manage several identifiers with one filter only.

23.8.2.2.3 Data discard

When LINFlex receives the identifier, the HRF bit in the LINSR is set and, if the HRIE bit in the LINIER is set, an RX interrupt is generated. If the received identifier does not concern the node, the software must set the DDRQ bit in LINCR2. LINFlex returns to idle state after bit DDRQ is set.

23.8.2.2.4 Error detection

In Slave mode, the following errors are detected:

- **Header error**: An error occurred during header reception (Break Delimiter error, Inconsistent Synch Field, Header Timeout).
- **Bit error**: During transmission, the value read back from the bus differs from the transmitted value.
- **Framing error**: A dominant state has been sampled on the stop bit of the currently received character (synch field, identifier field or data field).

- **Checksum error**: The computed checksum does not match the received one.

23.8.2.2.5 Error handling

In case of Bit Error detection during transmission, LINFlex stops the transmission of the frame after the corrupted bit. LINFlex returns to idle state and an interrupt is generated if the BEIE bit in the LINIER is set.

During reception, a Framing Error leads LINFlex to discard the current frame. LINFlex returns immediately to idle state. An interrupt is generated if the FEIE bit in the LINIER is set.

During reception, a Checksum Error leads LINFlex to discard the received frame. LINFlex returns to idle state. An interrupt is generated if the CEIE bit in the LINIER is set.

During header reception, a Break Delimiter error, an Inconsistent Synch Field or a Timeout error leads LINFlex to discard the header. An interrupt is generated if the HEIE bit in the LINIER is set. LINFlex returns to idle state.

23.8.2.2.6 Valid header

A received header is considered as valid when it has been received correctly according to the LIN protocol.

If a valid Break Field and Break Delimiter come before the end of the current header or at any time during a data field, the current header or data is discarded and the state machine synchronizes on this new break.

23.8.2.2.7 Valid message

A received or transmitted message is considered as valid when the data has been received or transmitted without error according to the LIN protocol.

23.8.2.2.8 Overrun

Once the message buffer is full, the next valid message reception leads to an overrun and a message is lost. The hardware sets the BOF bit in the LINSR to signal the overrun condition. Which message is lost depends on the configuration of the RX message buffer:

- If the buffer lock function is disabled (RBLM bit in LINCR1 cleared) the last message stored in the buffer is overwritten by the new incoming message. In this case the latest message is always available to the application.
- If the buffer lock function is enabled (RBLM bit in LINCR1 set) the most recent message is discarded and the previous message is available in the buffer.

23.8.2.3 Slave mode with identifier filtering

In the LIN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On header reception a slave node decides—depending on the identifier value—whether the software needs to receive or send a response. If the message does not target the node, it must be discarded without software intervention.

To fulfill this requirement, the LINFlex controller provides configurable filters in order to request software intervention only if needed. This hardware filtering saves CPU resources that would otherwise be needed by software for filtering.

23.8.2.3.1 Filter mode

Usually each of the eight IFCR registers filters one dedicated identifier, but this limits the number of identifiers LINFlex can handle to the number of IFCR registers implemented in the device. Therefore, in order to be able to handle more identifiers, the filters can be configured in mask mode.

In **identifier list mode** (the default mode), both filter registers are used as identifier registers. All bits of the incoming identifier must match the bits specified in the filter register.

In **mask mode**, the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”. For the bit mapping and registers organization, please refer to Figure 23-30.

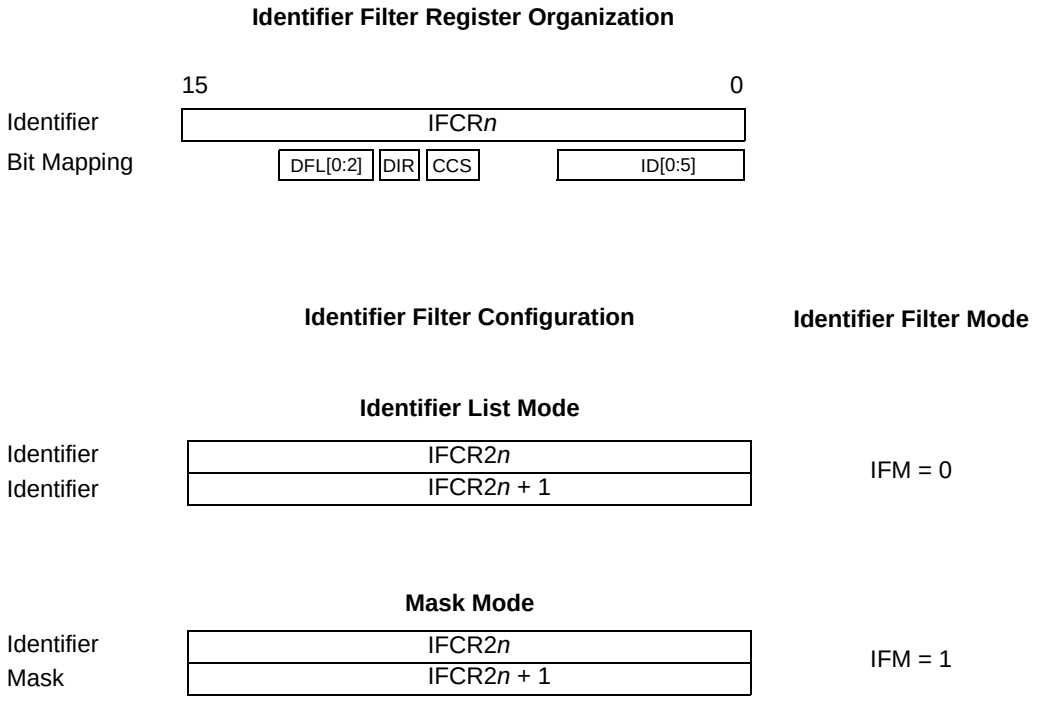


Figure 23-30. Filter configuration—register organization

23.8.2.3.2 Identifier filter mode configuration

The identifier filters are configured in the IFCR*x* registers. To configure an identifier filter the filter must first be deactivated by clearing the FACT bit in the IFER. The **identifier list** or **identifier mask** mode for the corresponding IFCR*x* registers is configured by the IFM bit in the IFMR. For each filter, the IFCR*x* register configures the ID (or the mask), the direction (TX or RX), the data field length, and the checksum type.

If no filter is active, an RX interrupt is generated on any received identifier event.

If at least one active filter is configured as TX, all received identifiers matching this filter generate a TX interrupt.

If at least one active filter is configured as RX, all received identifiers matching this filter generate an RX interrupt.

If no active filter is configured as RX, all received identifiers not matching TX filter(s) generate an RX interrupt.

Table 23-32. Filter to interrupt vector correlation

Number of active filters	Number of active filters configured as TX	Number of active filters configured as RX	Interrupt vector
0	0	0	RX interrupt on all identifiers
a (a > 0)	a	0	— TX interrupt on identifiers matching the filters, — RX interrupt on all other identifiers if BF bit is set, no RX interrupt if BF bit is reset
n (n = a + b)	a (a > 0)	b (b > 0)	— TX interrupt on identifiers matching the TX filters, — RX interrupt on identifiers matching the RX filters, — all other identifiers discarded (no interrupt)
b (b > 0)	0	b	— RX interrupt on identifiers matching the filters, — TX interrupt on all other identifiers if BF bit is set, no TX interrupt if BF bit is reset

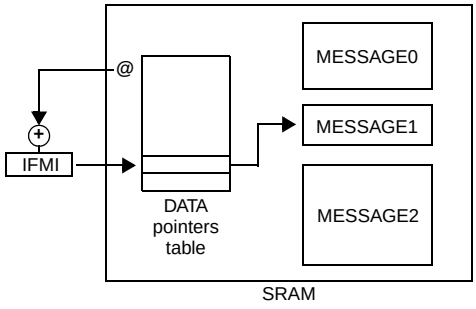


Figure 23-31. Identifier match index

23.8.2.4 Slave mode with automatic resynchronization

Automatic resynchronization must be enabled in Slave mode if $f_{periph\_set\_1\_clk}$ tolerance is greater than 1.5%. This feature compensates a $f_{periph\_set\_1\_clk}$ deviation up to 14%, as specified in LIN standard.

This mode is similar to Slave mode as described in Section 23.8.2.2, Slave mode with the addition of automatic resynchronization enabled by the LASE bit. In this mode LINFlex adjusts the fractional baud rate generator after each Synch Field reception.

Automatic resynchronization method

When automatic resynchronization is enabled, after each LIN Break, the time duration between five falling edges on RDI is sampled on $f_{periph\_set\_1\_clk}$ and the result of this measurement is stored in an internal 19-bit register called SM (not user accessible) (see Figure 23-32). Then the LFDIV value (and its associated registers LINIBRR and LINFBRR) are automatically updated at the end of the fifth falling edge. During LIN Synch Field measurement, the LINFlex state machine is stopped and no data is transferred to the data register.

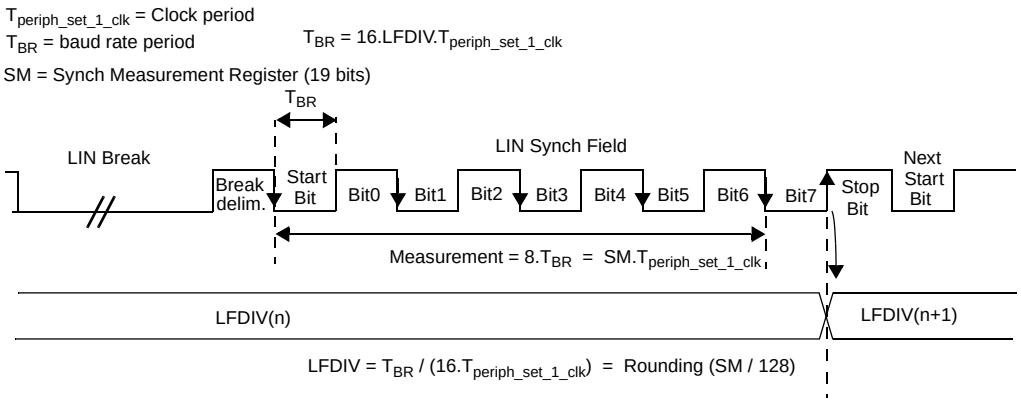


Figure 23-32. LIN synch field measurement

LFDIV is an unsigned fixed point number. The mantissa is coded on 12 bits in the LINIBRR and the fraction is coded on 4 bits in the LINFBRR.

If LASE bit = 1 then LFDIV is automatically updated at the end of each LIN Synch Field.

Three internal registers (not user-accessible) manage the auto-update of the LINFlex divider (LFDIV):

- LFDIV_NOM (nominal value written by software at LINIBRR and LINFBRR addresses)
- LFDIV_MEAS (results of the Field Synch measurement)
- LFDIV (used to generate the local baud rate)

On transition to idle, break or break delimiter state due to any error or on reception of a complete frame, hardware reloads LFDIV with LFDIV_NOM.

23.8.2.4.1 Deviation error on the Synch Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Synch Field (relative to the master oscillator). Two checks are performed in parallel.

The first check is based on a measurement between the first falling edge and the last falling edge of the Synch Field:

- If D1 > 14.84%, LHE is set.
- If D1 < 14.06%, LHE is not set.

- If 14.06% < D1 < 14.84%, LHE can be either set or reset depending on the dephasing between the signal on LINFlex_RX pin the $f_{periph\_set\_1\_clk}$ clock.

The second check is based on a measurement of time between each falling edge of the Synch Field:

- If D2 > 18.75%, LHE is set.
- If D2 < 15.62%, LHE is not set.
- If 15.62% < D2 < 18.75%, LHE can be either set or reset depending on the dephasing between the signal on LINFlex_RX pin the $f_{periph\_set\_1\_clk}$ clock.

Note that the LINFlex does not need to check if the next edge occurs slower than expected. This is covered by the check for deviation error on the full synch byte.

23.8.2.5 Clock gating

The LINFlex clock can be gated from the Mode Entry module (see Chapter 25, Mode Entry Module (MC_ME)). In LIN mode, the LINFlex controller acknowledges a clock gating request once the frame transmission or reception is completed.

23.8.3 8-bit timeout counter

23.8.3.1 LIN timeout mode

Setting the LTOM bit in the LINTCSR enables the LIN timeout mode. The LINOCR becomes read-only, and OC1[0:7] and OC2[0:7] output compare values in the LINOCR are automatically updated by hardware.

This configuration detects header timeout, response timeout, and frame timeout.

Depending on the LIN mode (selected by the MME bit in LINCR1), the 8-bit timeout counter will behave differently.

LIN timeout mode must not be enabled during LIN extended frames transmission or reception (that is, if the data field length in the BIDR is configured with a value higher than 8 data bytes).

23.8.3.1.1 LIN Master mode

Field RTO[0:3] in the LINTOCR can be used to tune response timeout and frame timeout values. Header timeout value is fixed to HTO[0:6] = 28-bit time.

Field OC1[0:7] checks T_{Header} and $T_{Response}$ and field OC2[0:7] checks T_{Frame} (refer to Figure 23-33).

When LINFlex moves from Break delimiter state to Synch Field state (refer to Section 23.7.2.3, LIN status register (LINSR)):

- OC1[0:7] is updated with the value of OC_{Header} (OC_{Header} = CNT[0:7] + 28),
- OC2[0:7] is updated with the value of OC_{Frame} (OC_{Frame} = CNT[0:7] + 28 + RTO[0:6] × 9 (frame timeout value for an 8-byte frame),
- the TOCE bit is set.

On the start bit of the first response data byte (and if no error occurred during the header reception), OC1[0:7] is updated with the value of $OC_{Response}$ ($OC_{Response}$ = CNT[0:7] + RTO[0:6] × 9 (response timeout value for an 8-byte frame)).

On the first response byte is received, OC1[0:7] and OC2[0:7] are automatically updated to check $T_{Response}$ and T_{Frame} according to RTO[0:6] (tolerance) and DFL[0:2].

On the checksum reception or in case of error in the header or response, the TOCE bit is reset.

If there is no response, frame timeout value does not take into account the DFL[0:2] value, and an 8-byte response (DFL = 7) is always assumed.

23.8.3.1.2 LIN Slave mode

Field RTO[0:3] in the LINTOCR can be used to tune response timeout and frame timeout values. Header timeout value is fixed to HTO[0:6].

OC1[0:7] checks T_{Header} and $T_{Response}$ and OC2[0:7] checks T_{Frame} (refer to Figure 23-33).

When LINFlex moves from Break state to Break Delimiter state (refer to Section 23.7.2.3, LIN status register (LINSR)):

- OC1[0:7] is updated with the value of OC_{Header} (OC_{Header} = CNT[0:7] + HTO[0:6]),
- OC2[0:7] is updated with the value of OC_{Frame} (OC_{Frame} = CNT[0:7] + HTO[0:6] + RTO[0:6] × 9 (frame timeout value for an 8-byte frame)),
- The TOCE bit is set.

On the start bit of the first response data byte (and if no error occurred during the header reception), OC1[0:7] is updated with the value of $OC_{Response}$ ($OC_{Response}$ = CNT[0:7] + RTO[0:7] × 9 (response timeout value for an 8-byte frame)).

Once the first response byte is received, OC1[0:7] and OC2[0:7] are automatically updated to check $T_{Response}$ and T_{Frame} according to RTO[0:6] (tolerance) and DFL[0:2].

On the checksum reception or in case of error in the header or data field, the TOCE bit is reset.

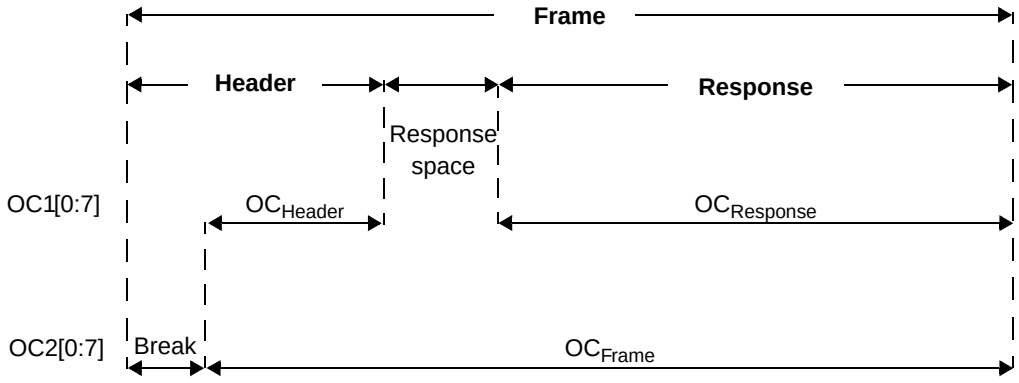


Figure 23-33. Header and response timeout

23.8.3.2 Output compare mode

Resetting the LTOM bit in the LINTCSR enables the output compare mode. This mode allows the user to fully customize the use of the counter.

OC1[0:7] and OC2[0:7] output compare values can be updated in the LINTOCR by software.

23.8.4 Interrupts

Table 23-33. LINFlex interrupt control

Interrupt event	Event flag bit	Enable control bit	Interrupt vector
Header Received interrupt	HRF	HRIE	RXI [1]
Data Transmitted interrupt	DTF	DTIE	TXI
Data Received interrupt	DRF	DRIE	RXI
Data Buffer Empty interrupt	DBEF	DBEIE	TXI
Data Buffer Full interrupt	DBFF	DBFIE	RXI
Wake-up interrupt	WUPF	WUPIE	RXI
LIN State interrupt [2]	LSF	LSIE	RXI
Buffer Overrun interrupt	BOF	BOIE	ERR
Framing Error interrupt	FEF	FEIE	ERR
Header Error interrupt	HEF	HEIE	ERR
Checksum Error interrupt	CEF	CEIE	ERR
Bit Error interrupt	BEF	BEIE	ERR
Output Compare interrupt	OCF	OCIE	ERR
Stuck at Zero interrupt	SZF	SZIE	ERR

1 In Slave mode, if at least one filter is configured as TX and enabled, header received interrupt vector is RXI or TXI depending on the value of identifier received.

2 For debug and validation purposes

Chapter 24
Memory Protection Unit (MPU)

24.1 Introduction

The AMBA-AHB Memory Protection Unit (MPU) provides hardware access control for all memory references generated in the device. Using preprogrammed region descriptors which define memory spaces and their associated access rights, the MPU concurrently monitors all system bus transactions and evaluates the appropriateness of each transfer. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with a protection error response. This module is commonly included as part of the platform.

24.1.1 Overview

The MPU module provides the following capabilities:

- Support for 12 program-visible 128-bit (4-word) region descriptors
  - Each region descriptor defines a modulo-32 byte space, aligned anywhere in memory
    - Region sizes can vary from a minimum of 32 bytes to a maximum of 4 GB
  - Two types of access control permissions defined in single descriptor word
    - Processors have separate {read, write, execute} attributes for supervisor and user accesses
    - Non-processor masters have {read, write} attributes
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
  - Alternate programming model view of the access control permissions word
- Memory-mapped platform device
  - Interface to 4 slave AHB ports: flash controller, system RAM controller, IPS peripherals bus, and QuadSPI module
    - Connections to the AHB address phase address and attributes
    - Typical location is immediately “downstream” of the platform’s crossbar switch
  - Connection to the IPS bus provides access to the MPU’s programming model

A simplified block diagram of the AHB_MPU module is shown in Figure 24-1. The AHB bus slave ports (s{0,1,2,3}_h*) are shown on the left side of the diagram, the region descriptor registers in the middle and the IPS bus interface (ips_*) on the right side. The evaluation macro contains the two magnitude comparators connected to the start and end address registers from each region descriptor (RGD*n*) as well as the combinational logic blocks to determine the region hit and the access protection error. For information on the details of the access evaluation macro, see Section 24.3.1, Access evaluation macro.

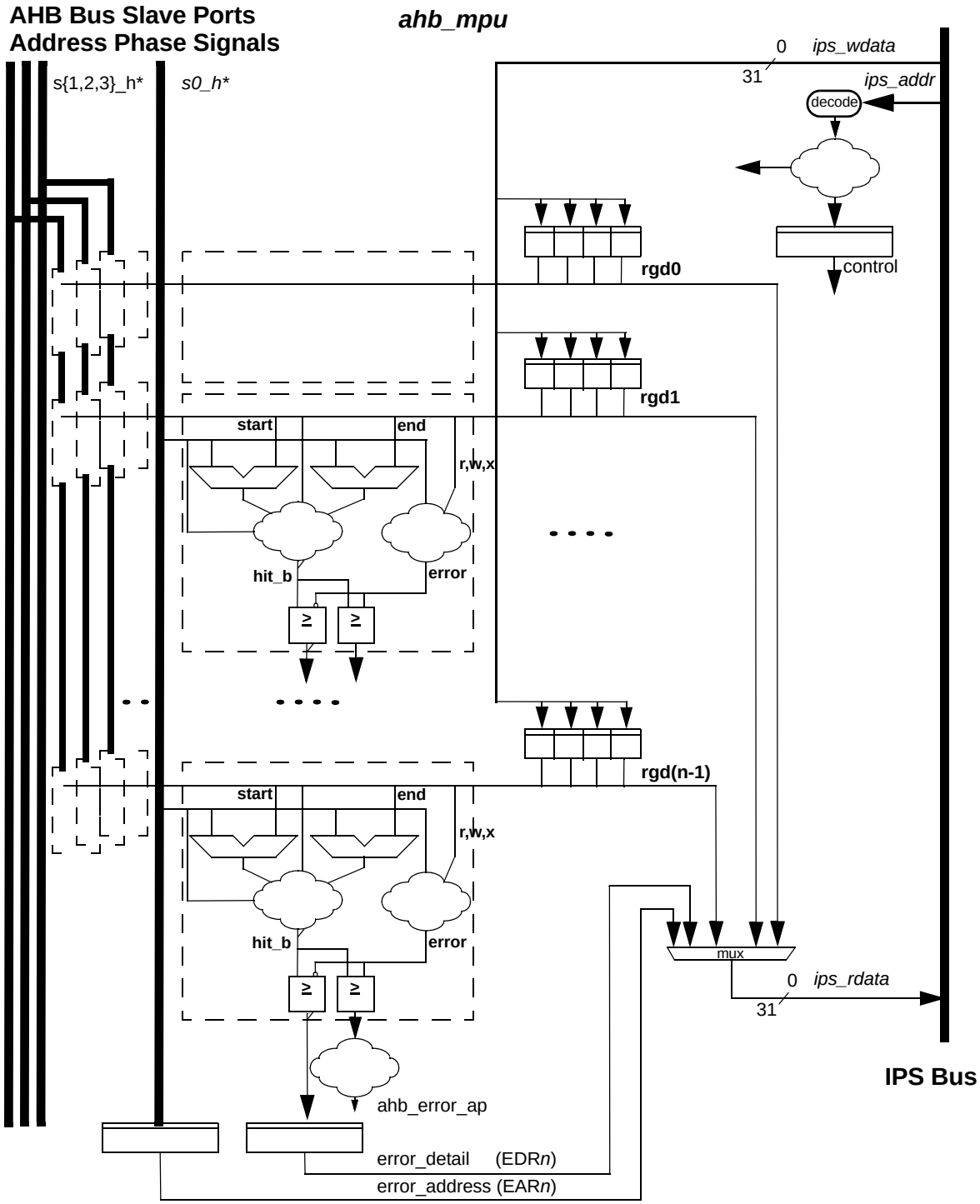


Figure 24-1. AHB_MPU block diagram

24.1.2 Features

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

- Support for 12 memory region descriptors, each 128 bits in size
  - Specification of start and end addresses provide granularity for region sizes from 32 bytes to 4 GB
  - Access control definitions: 2 bus masters (processor cores) support the traditional {read, write, execute} permissions with independent definitions for supervisor and user mode accesses
  - Automatic hardware maintenance of the region descriptor valid bit removes issues associated with maintaining a coherent image of the descriptor
  - Alternate memory view of the access control word for each descriptor provides an efficient mechanism to dynamically alter only the access rights of a descriptor
  - For overlapping region descriptors, priority is given to permission granting over access denying as this approach provides more flexibility to system software. See Section 24.3.2, Putting It All Together and AHB Error Terminations, for details and Section 24.5, Application information, for an example.
- Support for 3 AHB slave port connections: flash controller, system ram controller and IPS peripherals bus
  - MPU hardware continuously monitors every AHB slave port access using the preprogrammed memory region descriptors
  - An access protection error is detected if a memory reference does not hit in any memory region or the reference is flagged as illegal in all memory regions where it does hit. In the event of an access error, the AHB reference is terminated with an error response and the MPU inhibits the bus cycle being sent to the targeted slave device.
  - 64-bit error registers, one for each AHB slave port, capture the last faulting address, attributes and “detail” information
- Global MPU enable/disable control bit provides a mechanism to easily load region descriptors during system startup or allow complete access rights during debug with the module disabled

24.1.3 Modes of operation

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

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

24.1.4 External signal description

The MPU module does not include any external interface. The MPU's internal interfaces include an IPS connection for accessing the programming model and multiple connections to the address phase signals of the platform crossbar's slave AHB ports. From a platform topology viewpoint, the MPU module appears to be directly connected "downstream" from the crossbar switch with interfaces to the AHB slave ports.

24.2 Memory map and register description

The MPU module provides an IPS programming model mapped to an SPP-standard on-platform 16 KB space. The programming model is partitioned into three groups: control/status registers, the data structure containing the region descriptors and the alternate view of the region descriptor access control values.

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

Finally, the programming model allocates space for an MPU definition with 8 region descriptors and up to 3 AHB slave ports, like flash controller, system ram controller and IPS peripherals bus.

24.2.1 Memory map

The MPU programming model map is shown in Table 24-1.

Table 24-1. MPU memory map

Offset address	Register name	Register description	Size (bits)	Access	Location
0x0000	MPU_CESR	MPU Control/Error Status Register	32	R/ partial-W	on page 885
0x0004–0x000F	Reserved				
0x0010	MPU_EAR0	MPU Error Address Register, Slave Port 0	32	R-only	on page 886
0x0014	MPU_EDR0	MPU Error Detail Register, Slave Port 0	32	R-only	on page 887
0x0018	MPU_EAR1	MPU Error Address Register, Slave Port 1	32	R-only	on page 886
0x001C	MPU_EDR1	MPU Error Detail Register, Slave Port 1	32	R-only	on page 887
0x0020	MPU_EAR2	MPU Error Address Register, Slave Port 2	32	R-only	on page 886
0x0024	MPU_EDR2	MPU Error Detail Register, Slave Port 2	32	R-only	on page 887
0x0028	MPU_EAR3	MPU Error Address Register, Slave Port 3	32	R-only	on page 886
0x002C	MPU_EDR3	MPU Error Detail Register, Slave Port 3	32	R-only	on page 887
0x0030–0x03FF	Reserved				
0x0400	MPU_RGD0	MPU Region Descriptor 0	128	R/W	on page 888
0x0410	MPU_RGD1	MPU Region Descriptor 1	128	R/W	on page 888

Table 24-1. MPU memory map (continued)

Offset address	Register name	Register description	Size (bits)	Access	Location
0x0420	MPU_RGD2	MPU Region Descriptor 2	128	R/W	on page 888
0x0430	MPU_RGD3	MPU Region Descriptor 3	128	R/W	on page 888
0x0440	MPU_RGD4	MPU Region Descriptor 4	128	R/W	on page 888
0x0450	MPU_RGD5	MPU Region Descriptor 5	128	R/W	on page 888
0x0460	MPU_RGD6	MPU Region Descriptor 6	128	R/W	on page 888
0x0470	MPU_RGD7	MPU Region Descriptor 7	128	R/W	on page 888
0x0480	MPU_RGD8	MPU Region Descriptor 8	128	R/W	on page 888
0x0490	MPU_RGD9	MPU Region Descriptor 9	128	R/W	on page 888
0x04A0	MPU_RGD10	MPU Region Descriptor 10	128	R/W	on page 888
0x04B0	MPU_RGD11	MPU Region Descriptor 11	128	R/W	on page 888
0x04C0–0x07FF	Reserved				
0x0800	MPU_RGDAAC0	MPU RGD Alternate Access Control 0	32	R/W	on page 893
0x0804	MPU_RGDAAC1	MPU RGD Alternate Access Control 1	32	R/W	on page 893
0x0808	MPU_RGDAAC2	MPU RGD Alternate Access Control 2	32	R/W	on page 893
0x080C	MPU_RGDAAC3	MPU RGD Alternate Access Control 3	32	R/W	on page 893
0x0810	MPU_RGDAAC4	MPU RGD Alternate Access Control 4	32	R/W	on page 893
0x0814	MPU_RGDAAC5	MPU RGD Alternate Access Control 5	32	R/W	on page 893
0x0818	MPU_RGDAAC6	MPU RGD Alternate Access Control 6	32	R/W	on page 893
0x081C	MPU_RGDAAC7	MPU RGD Alternate Access Control 7	32	R/W	on page 893
0x0820	MPU_RGDAAC8	MPU RGD Alternate Access Control 8	32	R/W	on page 893
0x0824	MPU_RGDAAC9	MPU RGD Alternate Access Control 9	32	R/W	on page 893
0x0828	MPU_RGDAAC10	MPU RGD Alternate Access Control 10	32	R/W	on page 893
0x082C	MPU_RGDAAC11	MPU RGD Alternate Access Control 11	32	R/W	on page 893
0x0830–0x3FFF	Reserved				

24.2.2 Register description

The following sections detail the individual registers within the MPU's programming model.

24.2.2.1 MPU Control/Error Status Register (MPU_CESR)

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

Offset MPU_Base + 0x000 Access: Read/Partial Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SPERR								1	0	0	0	HRL				NSP				NRGD				0	0	0	0	0	0	0	VLD
W	w1c																															
Reset	0	0	0	0	0	0	0	0	1	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0

Figure 24-2. MPU Control/Error Status Register (MPU_CESR)

Table 24-2. MPU_CESR field descriptions

Field	Description
0–7 SPERR	Slave Port n Error, where the slave port number matches the bit number. Each bit in this field represents a flag maintained by the MPU for signaling the presence of a captured error contained in the MPU_EAR*n* and MPU_EDR*n* registers. The individual bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared when the corresponding bit is written as a logical one. If another error is captured at the exact same cycle as a write of a logical one, this flag remains set. A “find first one” instruction (or equivalent) can be used to detect the presence of a captured error. 0 The corresponding MPU_EAR*n*/MPU_EDR*n* registers do not contain a captured error. 1 The corresponding MPU_EAR*n*/MPU_EDR*n* registers do contain a captured error.
12–15 HRL	Hardware Revision Level. This 4-bit read-only field specifies the MPU’s hardware and definition revision level. It can be read by software to determine the functional definition of the module.
16–19 NSP	Number of Slave Ports. This 4-bit read-only field specifies the number of slave ports [1-8] connected to the MPU. This field contains values of 0b0001-0b1000, depending on the device configuration.
20–23 NRGD	Number of Region Descriptors. This 4-bit read-only field specifies the number of region descriptors implemented in the MPU. The defined encodings include: 0b00008 region descriptors 0b000112 region descriptors 0b001016 region descriptors
31 VLD	Valid. This bit provides a global enable/disable for the MPU. 0 The MPU is disabled. 1 The MPU is enabled. While the MPU is disabled, all accesses from all bus masters are allowed.

24.2.2.2 MPU Error Address Register, Slave Port *n* (MPU_EAR*n*)

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

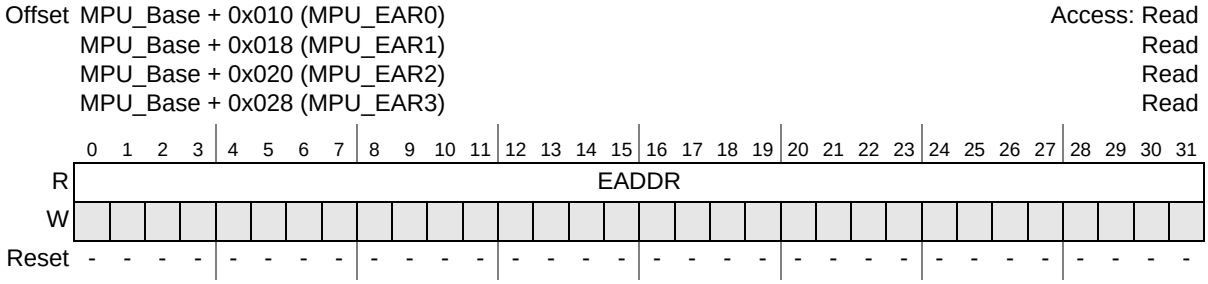


Figure 24-3. MPU Error Address Register, Slave Port *n* (MPU_EAR*n*)

Table 24-3. MPU_EAR*n* field descriptions

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

24.2.2.3 MPU Error Detail Register, Slave Port *n* (MPU_EDR*n*)

When the MPU detects an access error on slave port *n*, 32 bits of error detail are captured in this read-only register and the corresponding bit in the MPU_CESR[SPERR] field set. Information on the faulting address is captured in the corresponding MPU_EAR*n* register at the same time. Note this register and the corresponding MPU_EAR*n* register contain the most recent access error; there are no hardware interlocks with the MPU_CESR[SPERR] field as the error registers are always loaded upon the occurrence of each protection violation.

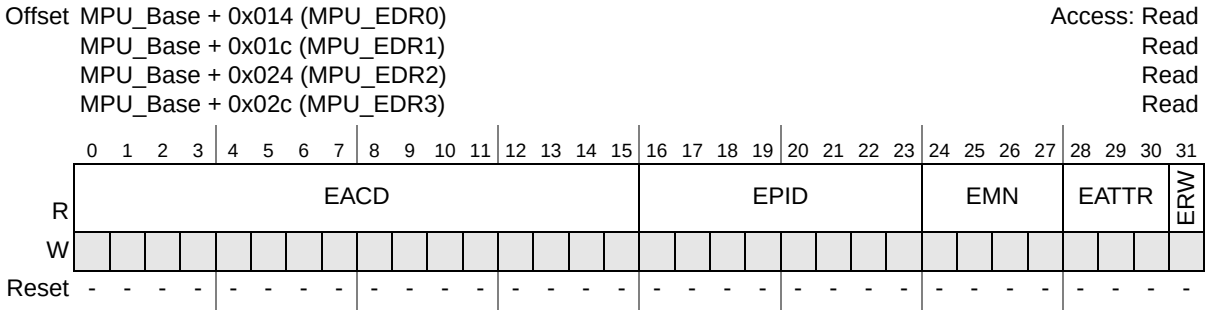


Figure 24-4. MPU Error Detail Register, Slave Port *n* (MPU_EDR*n*)

Table 24-4. MPU_EDR*n* field descriptions

Field	Description
0–15 EACD	Error Access Control Detail. This 16-bit read-only field implements one bit per region descriptor and is an indication of the region descriptor hit logically anded with the access error indication. The MPU performs a reference-by-reference evaluation to determine the presence/absence of an access error. When an error is detected, the hit-qualified access control vector is captured in this field. If the MPU_EDR*n* register contains a captured error and the EACD field is all zeroes, this signals an access that did not hit in any region descriptor. All non-zero EACD values signal references that hit in a region descriptor(s), but failed due to a protection error as defined by the specific set bits. If only a single EACD bit is set, then the protection error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, then the protection error was caused in an overlapping set of region descriptors.
16–23 EPID	Error Process Identification. This 8-bit read-only field records the process identifier of the faulting reference. The process identifier is typically driven only by processor cores; for other bus masters, this field is cleared.
24–27 EMN	Error Master Number. This 4-bit read-only field records the logical master number of the faulting reference. This field is used to determine the bus master that generated the access error.
28–30 EATTR	Error Attributes. This 3-bit read-only field records attribute information about the faulting reference. The supported encodings are defined as: 0b000User mode, instruction access 0b001User mode, data access 0b010Supervisor mode, instruction access 0b011Supervisor mode, data access All other encodings are reserved. For non-core bus masters, the access attribute information is typically wired to supervisor, data (0b011).
31 ERW	Error Read/Write. This 1-bit read-only field signals the access type (read, write) of the faulting reference. 0 Read 1 Write

24.2.2.4 MPU Region Descriptor *n* (MPU_RGD*n*)

Each 128-bit (16 byte) region descriptor specifies a given memory space and the access attributes associated with that space. The descriptor definition is the very essence of the operation of the Memory Protection Unit.

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

24.2.2.4.1 MPU Region Descriptor n, Word 0 (MPU_RGD*n*.Word0)

The first word of the MPU region descriptor defines the 0-modulo-32 byte start address of the memory region. Writes to this word clear the region descriptor's valid bit (see Section 24.2.2.4.4, MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3) for more information).

Offset MPU_Base + 0x400 + (16*n) + 0x0 (MPU_RGD*n*.Word0) Access:Read/write

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

Figure 24-5. MPU Region Descriptor, Word 0 Register (MPU_RGD*n*.Word0)

Table 24-5. MPU_RGD*n*.Word0 field descriptions

Field	Description
0–26 SRTADDR	Start Address. This field defines the most significant bits of the 0-modulo-32 byte start address of the memory region.

24.2.2.4.2 MPU Region Descriptor *n*, Word 1 (MPU_RGD*n*.Word1)

The second word of the MPU region descriptor defines the 31-modulo-32 byte end address of the memory region. Writes to this word clear the region descriptor's valid bit (see Section 24.2.2.4.4, MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3) for more information).

Offset MPU_Base + 0x400 + (16*n) + 0x4 (MPU_RGD*n*.Word1) Access:Read/write

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

Figure 24-6. MPU Region Descriptor, Word 1 Register (MPU_RGD*n*.Word1)

Table 24-6. MPU_RGD*n*.Word1 field descriptions

Field	Description
0–26 ENDADDR	End Address. This field defines the most significant bits of the 31-modulo-32 byte end address of the memory region. There are no hardware checks to verify that ENDADDR ≥ SRTADDR; it is software's responsibility to properly load these region descriptor fields.

24.2.2.4.3 MPU Region Descriptor *n*, Word 2 (MPU_RGD*n*.Word2)

The third word of the MPU region descriptor defines the access control rights of the memory region. The access control privileges are dependent on two broad classifications of bus masters. Bus masters 0-3 are typically reserved for processor cores and the corresponding access control is a 6-bit field defining separate privilege rights for user and supervisor mode accesses as well as the optional inclusion of a process identification field within the definition. Bus masters 4-7 are typically reserved for data movement engines and their capabilities are limited to separate read and write permissions. For these fields, the bus master number refers to the logical master number defined as the AHB hmaster[3:0] signal.

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

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch.

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

The evaluation logic defines the processor access type based on multiple AHB signals, as hwrite and hprot[1:0].

For non-processor data movement engines (bus masters 4-7), the evaluation logic simply uses hwrite to determine if the access is a read or write.

Writes to this word clear the region descriptor's valid bit (see Section 24.2.2.4.4, MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3) for more information). Since it is also expected that system software may adjust only the access controls within a region descriptor (MPU_RGD*n*.Word2) as different tasks execute, an alternate programming view of this 32-bit entity is provided. If only the access controls are being updated, this operation should be performed by writing to MPU_RGDAAC*n* (Alternate Access Control n) as stores to these locations do *not* affect the descriptor's valid bit.

Offset MPU_Base + 0x400 + (16*n) + 0x8 (MPU_RGD*n*.Word2) Access:Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M7RE	M7WE	M6RE	M6WE	M5RE	M5WE	M4RE	M4WE	M3PE	M3SM			M3UM				M2PE	M2SM			M2UM				M1PE	M1SM			M1UM				M0PE	M0SM			M0UM		
W													r	w	x						r	w	x						r	w	x	
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 24-7. MPU Region Descriptor, Word 2 Register (MPU_RGD*n*.Word2)

Table 24-7. MPU_RGD*n*.Word2 field descriptions

Field	Description
0 M7RE	Bus master 7 read enable. If set, this flag allows bus master 7 to perform read operations. If cleared, any attempted read by bus master 7 terminates with an access error and the read is not performed.
1 M7WE	Bus master 7 write enable. If set, this flag allows bus master 7 to perform write operations. If cleared, any attempted write by bus master 7 terminates with an access error and the write is not performed.
2 M6RE	Bus master 6 read enable. If set, this flag allows bus master 6 to perform read operations. If cleared, any attempted read by bus master 6 terminates with an access error and the read is not performed.
3 M6WE	Bus master 6 write enable. If set, this flag allows bus master 6 to perform write operations. If cleared, any attempted write by bus master 6 terminates with an access error and the write is not performed.
4 M5RE	Bus master 5 read enable. If set, this flag allows bus master 5 to perform read operations. If cleared, any attempted read by bus master 5 terminates with an access error and the read is not performed.
5 M5WE	Bus master 5 write enable. If set, this flag allows bus master 5 to perform write operations. If cleared, any attempted write by bus master 5 terminates with an access error and the write is not performed.
6 M4RE	Bus master 4 read enable. If set, this flag allows bus master 4 to perform read operations. If cleared, any attempted read by bus master 4 terminates with an access error and the read is not performed.
7 M4WE	Bus master 4 write enable. If set, this flag allows bus master 4 to perform write operations. If cleared, any attempted write by bus master 4 terminates with an access error and the write is not performed.

Table 24-7. MPU_RGD*n*.Word2 field descriptions

Field	Description
8 M3PE	Bus master 3 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
9–10 M3SM	Bus master 3 supervisor mode access control. This 2-bit field defines the access controls for bus master 3 when operating in supervisor mode. The M3SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M3UM for user mode
11–13 M3UM	Bus master 3 user mode access control. This 3-bit field defines the access controls for bus master 3 when operating in user mode. The M3UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
14 M2PE	Bus master 2 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
15–16 M2SM	Bus master 2 supervisor mode access control. This 2-bit field defines the access controls for bus master 2 when operating in supervisor mode. The M2SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M2UM for user mode
17–19 M2UM	Bus master 2 user mode access control. This 3-bit field defines the access controls for bus master 2 when operating in user mode. The M2UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
20 M1PE	Bus master 1 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
21–22 M1SM	Bus master 1 supervisor mode access control. This 2-bit field defines the access controls for bus master 1 when operating in supervisor mode. The M1SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M1UM for user mode
23–25 M1UM	Bus master 1 user mode access control. This 3-bit field defines the access controls for bus master 1 when operating in user mode. The M1UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
26 M0PE	Bus master 0 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.

Table 24-7. MPU_RGD*n*.Word2 field descriptions

Field	Description
27–28 M0SM	Bus master 0 supervisor mode access control. This 2-bit field defines the access controls for bus master 0 when operating in supervisor mode. The M0SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M0UM for user mode
29–31 M0UM	Bus master 0 user mode access control. This 3-bit field defines the access controls for bus master 0 when operating in user mode. The M0UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.

24.2.2.4.4 MPU Region Descriptor *n*, Word 3 (MPU_RGD*n*.Word3)

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

Since the region descriptor is a 128-bit entity, there are potential coherency issues as this structure is being updated since multiple writes are required to update the entire descriptor. Accordingly, the MPU hardware assists in the operation of the descriptor valid bit to prevent incoherent region descriptors from generating spurious access errors. In particular, it is expected that a complete update of a region descriptor is typically done with sequential writes to MPU_RGD*n*.Word0, then MPU_RGD*n*.Word1,... and finally MPU_RGD*n*.Word3. The MPU hardware automatically clears the valid bit on any writes to words {0,1,2} of the descriptor. Writes to this word set/clear the valid bit in a normal manner.

Since it is also expected that system software may adjust only the access controls within a region descriptor (MPU_RGD*n*.Word2) as different tasks execute, an alternate programming view of this 32-bit entity is provided. If only the access controls are being updated, this operation should be performed by writing to MPU_RGDAAC*n* (Alternate Access Control n) as stores to these locations do *not* affect the descriptor's valid bit.

Offset MPU_Base + 0x400 + (16*n) + 0xc (MPU_RGD*n*.Word3) Access:Read/write

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

Figure 24-8. MPU Region Descriptor, Word 3 Register (MPU_RGD*n*.Word3)

Table 24-8. MPU_RGD*n*.Word3 field descriptions

Field	Description
0–7 PID	Process Identifier. This 8-bit field specifies that the optional process identifier is to be included in the determination of whether the current access hits in the region descriptor. This field is combined with the PIDMASK and included in the region hit determination if MPU_RGDn.Word2[MxPE] is set.
8–15 PIDMASK	Process Identifier Mask. This 8-bit field provides a masking capability so that multiple process identifiers can be included as part of the region hit determination. If a bit in the PIDMASK is set, then the corresponding bit of the PID is ignored in the comparison. This field is combined with the PID and included in the region hit determination if MPU_RGD*n*.Word2[M*x*PE] is set. For more information on the handling of the PID and PIDMASK, see Section 24.3.1.1, Access evaluation—hit determination.
31 VLD	Valid. This bit signals the region descriptor is valid. Any write to MPU_RGD*n*.Word{0,1,2} clears this bit, while a write to MPU_RGD*n*.Word3 sets or clears this bit depending on bit 31 of the write operand. 0 Region descriptor is invalid 1 Region descriptor is valid

24.2.2.5 MPU Region Descriptor Alternate Access Control *n* (MPU_RGDAAC*n*)

As noted in Section 24.2.2.4.3, MPU Region Descriptor n, Word 2 (MPU_RGDn.Word2), it is expected that since system software may adjust only the access controls within a region descriptor (MPU_RGD*n*.Word2) as different tasks execute, an alternate programming view of this 32-bit entity is desired. If only the access controls are being updated, this operation should be performed by writing to MPU_RGDAAC*n* (Alternate Access Control *n*) as stores to these locations do not affect the descriptor's valid bit.

The memory address therefore provides an alternate location for updating MPU_RGD*n*.Word2.

Offset MPU_Base + 0x800 + (4*n) (MPU_RGDAACn) Access:Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M7RE	M7WE	M6RE	M6WE	M5RE	M5WE	M4RE	M4WE	M3PE	M3SM			M3UM			M2PE	M2SM				M1PE	M1SM			M1UM			M0PE	M0SM			M0UM	
W													r	w	x										r	w	x					r	w	x
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 24-9. MPU RGD Alternate Access Control *n* (MPU_RGDAAC*n*)

Since the MPU_RGDAAC*n* register is simply another memory mapping for MPU_RGD*n*.Word2, the field definitions shown in Table 24-9 are identical to those presented in Table 24-7.

Table 24-9. MPU_RGDAAC*n* field descriptions

Field	Description
0 M7RE	Bus master 7 read enable. If set, this flag allows bus master 7 to perform read operations. If cleared, any attempted read by bus master 7 terminates with an access error and the read is not performed.
1 M7WE	Bus master 7 write enable. If set, this flag allows bus master 7 to perform write operations. If cleared, any attempted write by bus master 7 terminates with an access error and the write is not performed.
2 M6RE	Bus master 6 read enable. If set, this flag allows bus master 6 to perform read operations. If cleared, any attempted read by bus master 6 terminates with an access error and the read is not performed.

Table 24-9. MPU_RGDAAC*n* field descriptions (continued)

Field	Description
3 M6WE	Bus master 6 write enable. If set, this flag allows bus master 6 to perform write operations. If cleared, any attempted write by bus master 6 terminates with an access error and the write is not performed.
4 M5RE	Bus master 5 read enable. If set, this flag allows bus master 5 to perform read operations. If cleared, any attempted read by bus master 5 terminates with an access error and the read is not performed.
5 M5WE	Bus master 5 write enable. If set, this flag allows bus master 5 to perform write operations. If cleared, any attempted write by bus master 5 terminates with an access error and the write is not performed.
6 M4RE	Bus master 4 read enable. If set, this flag allows bus master 4 to perform read operations. If cleared, any attempted read by bus master 4 terminates with an access error and the read is not performed.
7 M4WE	Bus master 4 write enable. If set, this flag allows bus master 4 to perform write operations. If cleared, any attempted write by bus master 4 terminates with an access error and the write is not performed.
8 M3PE	Bus master 3 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
9–10 M3SM	Bus master 3 supervisor mode access control. This 2-bit field defines the access controls for bus master 3 when operating in supervisor mode. The M3SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M3UM for user mode
11–13 M3UM	Bus master 3 user mode access control. This 3-bit field defines the access controls for bus master 3 when operating in user mode. The M3UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
14 M2PE	Bus master 2 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
15–16 M2SM	Bus master 2 supervisor mode access control. This 2-bit field defines the access controls for bus master 2 when operating in supervisor mode. The M2SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M2UM for user mode
17–19 M2UM	Bus master 2 user mode access control. This 3-bit field defines the access controls for bus master 2 when operating in user mode. The M2UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
20 M1PE	Bus master 1 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.

Table 24-9. MPU_RGDAAC*n* field descriptions (continued)

Field	Description
21–22 M1SM	Bus master 1 supervisor mode access control. This 2-bit field defines the access controls for bus master 1 when operating in supervisor mode. The M1SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M1UM for user mode
23–25 M1UM	Bus master 1 user mode access control. This 3-bit field defines the access controls for bus master 1 when operating in user mode. The M1UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
26 M0PE	Bus master 0 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGD*n*.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
27–28 M0SM	Bus master 0 supervisor mode access control. This 2-bit field defines the access controls for bus master 0 when operating in supervisor mode. The M0SM field is defined as: 0b00 r, w, x = read, write and execute allowed 0b01 r, –, x = read and execute allowed, but no write 0b10 r, w, – = read and write allowed, but no execute 0b11 Same access controls as that defined by M0UM for user mode
29–31 M0UM	Bus master 0 user mode access control. This 3-bit field defines the access controls for bus master 0 when operating in user mode. The M0UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.

24.3 Functional description

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

24.3.1 Access evaluation macro

As previously discussed, the basic operation of the MPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in Figure 24-10, the access evaluation macro inputs the AHB system bus address phase signals (AHB_ap) and the contents of a region descriptor (RGD*n*) and performs two major functions: region hit determination (hit_b) and detection of an access protection violation (error).

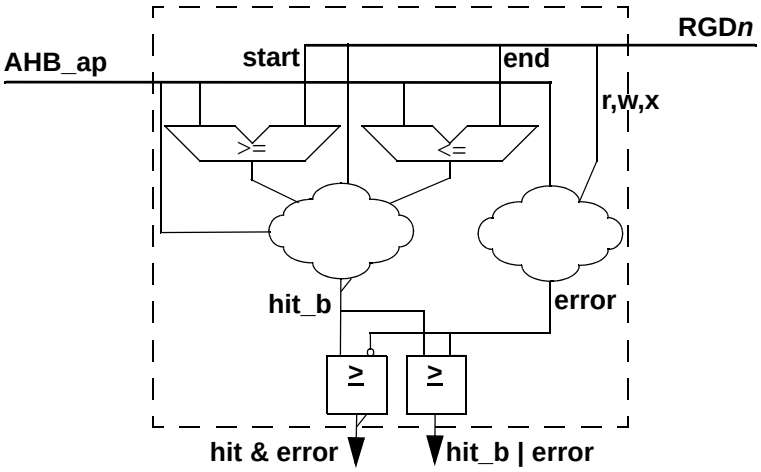


Figure 24-10. MPU access evaluation macro

Figure 24-10 is not intended to be a schematic of the actual access evaluation macro, but rather a generalized block diagram showing the major functions included in this logic block.

24.3.1.1 Access evaluation—hit determination

To evaluate the region hit determination, the MPU uses two magnitude comparators in conjunction with the contents of a region descriptor: the current access must be included between the region's “start” and “end” addresses and simultaneously the region's valid bit must be active.

Recall there are no hardware checks to verify that region's “end” address is greater then region's “start” address, and it is software’s responsibility to properly load appropriate values into these fields of the region descriptor.

In addition to this, the optional process identifier is examined against the region descriptor’s PID and PIDMASK fields. In order to generate the pid_hit indication: the current PID with its PIDMASK must be equal to the region's PID with its PIDMASK. Also the process identifier enable is take into account in this comparison so that the MPU forces the pid_hit term to be asserted in the case of AHB bus master does not provide its process identifier.

24.3.1.2 Access evaluation—privilege violation determination

While the access evaluation macro is making the region hit determination, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown in Table 24-10.

Table 24-10. Protection Violation Definition

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

As shown in Figure 24-10, the output of the protection violation logic is the error signal.

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

24.3.2 Putting It All Together and AHB Error Terminations

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

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

The third condition reflects that priority is given to permission granting over access denying for overlapping regions as this approach provides more flexibility to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Section 24.5, Application information.

In event of a protection error, the MPU requires two distinct actions:

1. intercepts the error during the AHB address phase (first cycle out of two) and cancels the transaction before it is seen by the slave device.
2. performs the required logic functions to force the standard 2-cycle AHB error response to properly terminate the bus transaction and then provides the right values to the crossbar switch to commit the AHB transaction to other portions of the platform.

If instead the access is allowed, then the MPU simply passes all “original” AHB signals to the slave device. In this case, from functionality point of view, the MPU is fully transparent.

24.4 Initialization Information

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

Typically the appropriate number of region descriptors (MPU_RGD*n*) are loaded at system startup, including the setting of the MPU_RGD*n*.Word3[VLD] bits, before MPU_CESR[VLD] is set, enabling the module. This approach allows all the loaded region descriptors to be enabled simultaneously. Recall if a memory reference does not hit in any region descriptor, the attempted access is terminated with an error.

24.5 Application information

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

1. Creation of a new memory region requires loading the appropriate region descriptor into an available register location. When a new descriptor is loaded into a RGD*n*, it would typically be performed using four 32-bit word writes. As discussed in Section 24.2.2.4.4, MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3), the hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues associated with the multi-cycle descriptor writes. Deletion/removal of an existing memory region is performed simply by clearing MPU_RGD*n*.Word3[VLD].
2. If only the access rights for an existing region descriptor need to change, a 32-bit write to the alternate version of the access control word (MPU_RGDAAC*n*) would typically be performed. Recall writes to the region descriptor using this alternate access control location do not affect the valid bit, so there are, by definition, no coherency issues involved with the update. The access rights associated with the memory region switch instantaneously to the new value as the IPS write completes.
3. If the region's start and end addresses are to be changed, this would typically be performed by writing a minimum of three words of the region descriptor: MPU_RGD*n*.Word{0,1,3}, where the writes to Word0 and Word1 redefine the start and end addresses respectively and the write to Word3 re-enables the region descriptor valid bit. In many situations, all four words of the region descriptor would be rewritten.
4. Typically, references to the MPU's programming model would be restricted to supervisor mode accesses from a specific processor(s), so a region descriptor would be specifically allocated for this purpose with attempted accesses from other masters or while in user mode terminated with an error.
5. When the MPU detects an access error, the current AHB bus cycle is terminated with an error response and information on the faulting reference captured in the MPU_EAR*n* and MPU_EDR*n* registers. The error-terminated AHB bus cycle typically initiates some type of error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. In any event, the processor can retrieve the captured error address and detail information simply be reading the MPU_E{A,D}R*n* registers. Information on which error registers contain captured fault data is signaled by MPU_CESR[SPERR].

Chapter 25
Mode Entry Module (MC_ME)

25.1 Introduction

25.1.1 Overview

The MC_ME controls the device mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

Figure 25-1 depicts the MC_ME block diagram.

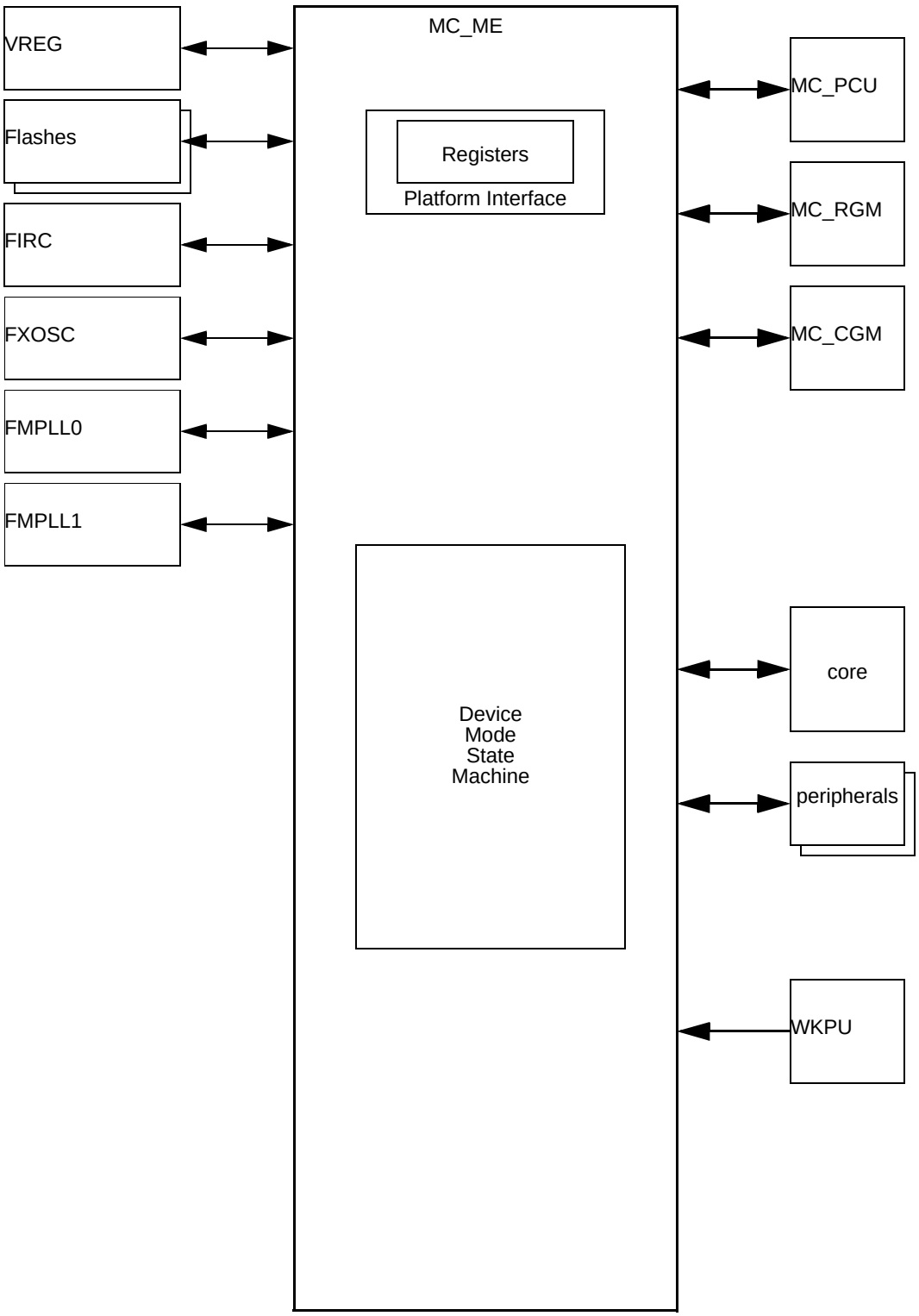


Figure 25-1. MC_ME block diagram

25.1.2 Features

The MC_ME includes the following features:

- Control of the available modes by the ME_ME register
- Definition of various device mode configurations by the ME_<*mode*>_MC registers
- Control of the actual device mode by the ME_MCTL register
- Capture of the current mode and various resource status within the contents of the ME_GS register
- Optional generation of various mode transition interrupts
- Status bits for each cause of invalid mode transitions
- Peripheral clock gating control based on the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTL0…143 registers
- Capture of current peripheral clock gated/enabled status

25.1.3 Modes of operation

The MC_ME is based on several device modes corresponding to different usage models of the device. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as Reset, DRUN, Safe, and Test. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as Run0…3, Halt, Stop, and Standby which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, Safe, Test, and Run0…3 are the device software running modes.

Table 25-1 describes the MC_ME modes.

Table 25-1. MC_ME mode descriptions

Name	Description	Entry	Exit
Reset	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the device. It manages hardware initialization of chip configuration, voltage regulators, oscillators, PLLs, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter USER modes. BAM when present is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from Safe, Test and Run0…3, wakeup request from Standby	system reset assertion, Run0…3, Test, Standby via software, Safe via software or hardware failure.
Safe	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, Test, and Run0…3	system reset assertion, DRUN via software

Table 25-1. MC_ME mode descriptions (continued)

Name	Description	Entry	Exit
Test	This is a chip-wide service mode which is intended to provide a control environment for device self-test. It may enable the application to run its own self-test like flash checksum, memory BIST etc.	software request from DRUN	system reset assertion, DRUN via software
Run0…3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock and power configurations of the system with respect to each other.	software request from DRUN, interrupt event from Halt, interrupt or wakeup event from Stop	system reset assertion, Safe via software or hardware failure, other Run0…3 modes, Halt, Stop, Standby via software
Halt	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals like PLL, flash, main regulator etc. for efficient power management at the cost of higher wakeup latency.	software request from Run0…3	system reset assertion, Safe on hardware failure, Run0…3 on interrupt event
Stop	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including oscillator for efficient power management at the cost of higher wakeup latency.	software request from Run0…3	system reset assertion, Safe on hardware failure, Run0…3 on interrupt event or wakeup event
Standby	This is a reduced-leakage low-power mode during which power supply is cut off from most of the device. Wakeup from this mode takes a relatively long time, and content is lost or must be restored from backup.	software request from Run0…3, DRUN modes	system reset assertion, DRUN on wakeup event

25.2 External signal description

The MC_ME has no connections to any external pins.

25.3 Memory map and register definition

The MC_ME contains registers for:

- Mode selection and status reporting
- Mode configuration
- Mode transition interrupts status and mask control
- Scalable number of peripheral sub-mode selection and status reporting

25.3.1 Memory map

Table 25-2. MC_ME register description

Address	Name	Description	Size	Access
0xC3FD_C000	ME_GS	Global Status	word	read
0xC3FD_C004	ME_MCTL	Mode Control	word	read/write
0xC3FD_C008	ME_ME	Mode Enable	word	read/write
0xC3FD_C00C	ME_IS	Interrupt Status	word	read/write
0xC3FD_C010	ME_IM	Interrupt Mask	word	read/write
0xC3FD_C014	ME_IMTS	Invalid Mode Transition Status	word	read/write
0xC3FD_C018	ME_DMTS	Debug Mode Transition Status	word	read
0xC3FD_C020	ME_RESET_MC	Reset Mode Configuration	word	read
0xC3FD_C024	ME_TEST_MC	Test Mode Configuration	word	read/write
0xC3FD_C028	ME_SAFE_MC	Safe Mode Configuration	word	read/write
0xC3FD_C02C	ME_DRUN_MC	DRUN Mode Configuration	word	read/write
0xC3FD_C030	ME_RUN0_MC	Run0 Mode Configuration	word	read/write
0xC3FD_C034	ME_RUN1_MC	Run1 Mode Configuration	word	read/write
0xC3FD_C038	ME_RUN2_MC	Run2 Mode Configuration	word	read/write
0xC3FD_C03C	ME_RUN3_MC	Run3 Mode Configuration	word	read/write
0xC3FD_C040	ME_HALT_MC	Halt Mode Configuration	word	read/write
0xC3FD_C048	ME_STOP_MC	Stop Mode Configuration	word	read/write
0xC3FD_C054	ME_STANDBY_MC	Standby Mode Configuration	word	read/write
0xC3FD_C060	ME_PS0	Peripheral Status 0	word	read
0xC3FD_C064	ME_PS1	Peripheral Status 1	word	read
0xC3FD_C068	ME_PS2	Peripheral Status 2	word	read
0xC3FD_C06C	ME_PS3	Peripheral Status 3	word	read
0xC3FD_C080	ME_RUN_PC0	Run Peripheral Configuration 0	word	read/write
0xC3FD_C084	ME_RUN_PC1	Run Peripheral Configuration 1	word	read/write
...				
0xC3FD_C09C	ME_RUN_PC7	Run Peripheral Configuration 7	word	read/write
0xC3FD_C0A0	ME_LP_PC0	Low-Power Peripheral Configuration 0	word	read/write
0xC3FD_C0A4	ME_LP_PC1	Low-Power Peripheral Configuration 1	word	read/write
...				

Table 25-2. MC_ME register description (continued)

Address	Name	Description	Size	Access
0xC3FD_C0BC	ME_LP_PC7	Low-Power Peripheral Configuration 7	word	read/write
0xC3FD_C0C4	ME_PCTL4	DSPI0 Control	byte	read/write
0xC3FD_C0C5	ME_PCTL5	DSPI1 Control	byte	read/write
0xC3FD_C0CA	ME_PCTL10	QuadSPI Control	byte	read/write
0xC3FD_C0D0	ME_PCTL16	FlexCAN0 Control	byte	read/write
0xC3FD_C0D1	ME_PCTL17	FlexCAN1 Control	byte	read/write
0xC3FD_C0D7	ME_PCTL23	DMA_CH_MUX Control	byte	read/write
0xC3FD_C0E0	ME_PCTL32	ADC0 Control	byte	read/write
0xC3FD_C0EC	ME_PCTL44	I2C_DMA0 Control	byte	read/write
0xC3FD_C0ED	ME_PCTL45	I2C_DMA1 Control	byte	read/write
0xC3FD_C0EE	ME_PCTL46	I2C_DMA2 Control	byte	read/write
0xC3FD_C0EF	ME_PCTL47	I2C_DMA3 Control	byte	read/write
0xC3FD_C0F0	ME_PCTL48	LIN_FLEX0 Control	byte	read/write
0xC3FD_C0F1	ME_PCTL49	LIN_FLEX1 Control	byte	read/write
0xC3FD_C0F8	ME_PCTL56	GaugeDriver Control	byte	read/write
0xC3FD_C0FC	ME_PCTL60	CANSampler Control	byte	read/write
0xC3FD_C0FD	ME_PCTL61	LCD Control	byte	read/write
0xC3FD_C0FE	ME_PCTL62	SGL Control	byte	read/write
0xC3FD_C0FF	ME_PCTL63	DCU Control	byte	read/write
0xC3FD_C104	ME_PCTL68	SIUL Control	byte	read/write
0xC3FD_C108	ME_PCTL72	eMIOS0 Control	byte	read/write
0xC3FD_C109	ME_PCTL73	eMIOS1 Control	byte	read/write
0xC3FD_C11B	ME_PCTL91	RTC/API Control	byte	read/write
0xC3FD_C11C	ME_PCTL92	PIT_RTI Control	byte	read/write
0xC3FD_C128	ME_PCTL104	CMU0 Control	byte	read/write

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- Not change register content
- Cause a transfer error

Table 25-3. MC_ME memory map

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FD_C000	ME_GS	R	S_CURRENT_MODE				S_MTRANS	0	0	0	S_PDO	0	0	S_MVR	S_DFLA		S_CFLA	
		W																
		R	0	0	0	0	0	0	0	0	S_FMPLL1	S_FMPLL0	S_FXOSC	S_FIRC	S_SYSCLK			
		W																
0xC3FD_C004	ME_MCTL	R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
		W	KEY															
0xC3FD_C008	ME_ME	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	STANDBY	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
		W																
0xC3FD_C00C	ME_IS	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	I_ICONF	I_IMODE	I_SAFE	I_MTC
		W													w1c	w1c	w1c	w1c
0xC3FD_C010	ME_IM	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	M_ICONF	M_IMODE	M_SAFE	M_MTC
		W																
0xC3FD_C014	ME_IMTS	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
		W												w1c	w1c	w1c	w1c	w1c

Table 25-3. MC_ME memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FD_C018	ME_DMTS	R	0	0	0	0	0	0	0	0	MPH_BUSY	0	0	PMC_PROG	CORE_DBG	0	0	SMR
		W																
		R	0	FMPLL0_SC	FXOSC_SC	FIRC_SC	SSCLK_SC	SYSCLK_SW	DFLASH_SC	CFLASH_SC	CDP_PRPH_0_143	0	0	0	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
		W																
0xC3FD_C01C	Reserved																	
0xC3FD_C020	ME_RESET_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
		W																
0xC3FD_C024	ME_TEST_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
		W																
0xC3FD_C028	ME_SAFE_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
		W																

Table 25-3. MC_ME memory map (continued)

Address	Name		0 16	1 17	2 18	3 19	4 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
0xC3FD_C02C	ME_DRUN_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
		W																
0xC3FD_C030 … 0xC3FD_C03C	ME_RUN0…3_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
		W																
0xC3FD_C040	ME_HALT_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
		W																
0xC3FD_C044	Reserved																	
0xC3FD_C048	ME_STOP_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYS CLK	0	0	0
		W																
0xC3FD_C04C … 0xC3FD_C050	Reserved																	

Table 25-3. MC_ME memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FD_C054	ME_STANDBY_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
		W																
		R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
		W																
0xC3FD_C058 … 0xC3FD_C05C	Reserved																	
0xC3FD_C060	ME_PS0	R	S_BAM	0	0	0	0	0	0	0	S_DMA_CH_MUX	0	0	0	0	0	S_FlexCAN1	S_FlexCAN0
		W																
		R	0	0	0	0	0	S_QUADSPI	0	0	0	S_QUADSPI	S_DSPI1	S_DSPI0	0	0	0	0
		W																
0xC3FD_C064	ME_PS1	R	S_DCU	S_SGL	S_LCD	S_CANSampler	0	0	0	S_GAUGEDRIVER	0	0	0	0	0	0	S_LIN_FLEX1	S_LIN_FLEX0
		W																
		R	S_I2C_DMA3	S_I2C_DMA2	S_I2C_DMA1	S_I2C_DMA0	0	0	0	0	0	0	0	0	0	0	0	S_ADC0
		W																

Table 25-3. MC_ME memory map (continued)

Address	Name		0 / 16	1 / 17	2 / 18	3 / 19	4 / 20	5 / 21	6 / 22	7 / 23	8 / 24	9 / 25	10 / 26	11 / 27	12 / 28	13 / 29	14 / 30	15 / 31
0xC3FD_C068	ME_PS2	R	0	0	0	S_PIT_RTI	S_RTC_API	S_MC_PCU	S_MC_RGM	S_MC_CGM	S_MC_ME	S_SSCM	0	0	0	0	0	0
		W																
		R	0	0	0	S_CFLASH1	0	0	S_eMIOS1	S_eMIOS0	0	0	S_WKPU	S_SIUL	S_DFLASH0	S_CFLASH0	0	0
		W																
0xC3FD_C06C	ME_PS3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	S_CMU0	0	0	0	0	0	0	0
		W																
0xC3FD_C070	Reserved																	
0xC3FD_C074 … 0xC3FD_C07C	Reserved																	
0xC3FD_C080 … 0xC3FD_C09C	ME_RUN_PC0…7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
		W																
0xC3FD_C0A0 … 0xC3FD_C0BC	ME_LP_PC0…7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	STANDBY	0	0	STOP	0	HALT	0	0	0	0	0	0	0	0
		W																

Table 25-3. MC_ME memory map (continued)

Address	Name		0 16	1 17	2 18	3 19	4 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
0xC3FD_C0C0 … 0xC3FD_C14C	ME_PCTL0…143	R	0	DBG_F	LP_CFG			RUN_CFG			0	DBG_F	LP_CFG			RUN_CFG		
		W																
		R	0	DBG_F	LP_CFG			RUN_CFG			0	DBG_F	LP_CFG			RUN_CFG		
		W																
0xC3FD_C150 … 0xC3FD_FFFC	Reserved																	

25.3.2 Register description

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0xC3FD_C080, as a half-word at address 0xC3FD_C082, or as a byte at address 0xC3FD_C083.

25.3.2.1 Global Status Register (ME_GS)

Address 0xC3FD_C000 Access: Supervisor read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	0	0	0	S_PDO	0	0	S_MVR	S_DFLA		S_CFLA	
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	S_FMPLL1	S_FMPLL0	S_FXOSC	S_FIRC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-2. Global Status Register (ME_GS)

This register contains global mode status.

Table 25-4. Global Status Register (ME_GS) field descriptions

Field	Description
S_CURRENT_MODE	Current device mode status 0000 Reset 0001 Test 0010 Safe 0011 DRUN 0100 Run0 0101 Run1 0110 Run2 0111 Run3 1000 Halt 1001 Reserved 1010 Stop 1011 Reserved 1100 Reserved 1101 Standby 1110 Reserved 1111 Reserved
S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
S_PDO	Output power-down status — This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In Safe/Test modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In Stop mode, only pad power sequence driver is disabled but the state of the output is kept. In Standby mode, the power sequence driver and all pads except those mapped on wakeup lines are not powered and therefore high impedance. Wakeup lines configuration remains unchanged
S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
S_DFLA	Data flash availability status 00 Data flash is not available 01 Data flash is in power-down mode 10 Data flash is in low-power mode 11 Data flash is in normal mode and available for use
S_CFLA	Code flash availability status 00 Code flash is not available 01 Code flash is in power-down mode 10 Code flash is in low-power mode 11 Code flash is in normal mode and available for use
S_SSCLK1	Secondary system clock source 1 status 0 Secondary system clock source 1 is not stable 1 Secondary system clock source 1 is providing a stable clock
S_FMPLL1	secondary frequency modulated phase locked loop status 0 secondary frequency modulated phase locked loop is not stable 1 secondary frequency modulated phase locked loop is providing a stable clock
S_FMPLL0	primary frequency modulated phase locked loop status 0 primary frequency modulated phase locked loop is not stable 1 primary frequency modulated phase locked loop is providing a stable clock

Table 25-4. Global Status Register (ME_GS) field descriptions (continued)

Field	Description
S_FXOSC	fast external crystal oscillator (4-16MHz) status 0 fast external crystal oscillator (4-16MHz) is not stable 1 fast external crystal oscillator (4-16MHz) is providing a stable clock
S_FIRC	fast internal RC oscillator (16MHz) status 0 fast internal RC oscillator (16MHz) is not stable 1 fast internal RC oscillator (16MHz) is providing a stable clock
S_SYSCLK	System clock switch status — These bits specify the system clock currently used by the system. 0000 16MHz int. RC osc. 0001 div. 16MHz int. RC osc. 0010 reserved 0011 div. 4-16MHz ext. osc. 0100 primary freq. mod. PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

25.3.2.2 Mode Control Register (ME_MCTL)

Address 0xC3FD_C004 Access: Supervisor read/write

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

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

Figure 25-3. Mode Control Register (ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<*mode*>_MC registers must respect this for successful mode requests.

NOTE

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

Table 25-5. Mode Control Register (ME_MCTL) field descriptions

Field	Description
TARGET_MODE	Target device mode — These bits provide the target device mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering Safe on hardware request. Also, while exiting from the Halt and Stop modes on hardware exit events, these are updated with the appropriate Run0…3 mode value. 0000 Reset 0001 Test 0010 Safe 0011 DRUN 0100 Run0 0101 Run1 0110 Run2 0111 Run3 1000 Halt 1001 Reserved 1010 Stop 1011 Reserved 1100 Reserved 1101 Standby 1110 Reserved 1111 Reserved
KEY	Control key — These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101101011110000 (0x5AF0) INVERTED KEY: 1010010100001111 (0xA50F)

25.3.2.3 Mode Enable Register (ME_ME)

Address 0xC3FD_C008 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STANDBY	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 25-4. Mode Enable Register (ME_ME)

This register allows a way to disable the device modes which are not required for a given device. Reset, Safe, DRUN, and Run0 modes are always enabled.

Table 25-6. Mode Enable Register (ME_ME) field descriptions

Field	Description
STANDBY	Standby mode enable 0 Standby mode is disabled 1 Standby mode is enabled
STOP	Stop mode enable 0 Stop mode is disabled 1 Stop mode is enabled
HALT	Halt mode enable 0 Halt mode is disabled 1 Halt mode is enabled
RUN3	Run3 mode enable 0 Run3 mode is disabled 1 Run3 mode is enabled
RUN2	Run2 mode enable 0 Run2 mode is disabled 1 Run2 mode is enabled
RUN1	Run1 mode enable 0 Run1 mode is disabled 1 Run1 mode is enabled
RUN0	Run0 mode enable 0 Run0 mode is disabled 1 Run0 mode is enabled
DRUN	DRUN mode enable 0 DRUN mode is disabled 1 DRUN mode is enabled
SAFE	Safe mode enable 0 Safe mode is disabled 1 Safe mode is enabled
TEST	Test mode enable 0 Test mode is disabled 1 Test mode is enabled
RESET	Reset mode enable 0 Reset mode is disabled 1 Reset mode is enabled

25.3.2.4 Interrupt Status Register (ME_IS)

Address 0xC3FD_C00C Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	I_ICONF	I_IMODE	I_SAFE	I_MTC
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-5. Interrupt Status Register (ME_IS)

This register provides the current interrupt status.

Table 25-7. Interrupt Status Register (ME_IS) field descriptions

Field	Description
I_ICONF	Invalid mode configuration interrupt — This bit is set whenever a write operation to ME_<*mode*>_MC registers with invalid mode configuration is attempted. It is cleared by writing a 1 to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending **Note:** The I_ICONF bit will not detect that modes that select a PLL to be active also have the FXOSC enabled. Therefore, always ensure that any mode that selects PLL as the system clock also has the associated FXOSC bit set.
I_IMODE	Invalid mode interrupt — This bit is set whenever an invalid mode transition is requested. It is cleared by writing a 1 to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
I_SAFE	Safe mode interrupt — This bit is set whenever the device enters Safe mode on hardware requests generated in the system. It is cleared by writing a 1 to this bit. 0 No Safe mode interrupt occurred 1 Safe mode interrupt is pending
I_MTC	Mode transition complete interrupt — This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a 1 to this bit. This mode transition interrupt bit will not be set while entering low-power modes Halt, Stop, or Standby. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

25.3.2.5 Interrupt Mask Register (ME_IM)

Address 0xC3FD_C010 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-6. Interrupt Mask Register (ME_IM)

This register controls whether an event generates an interrupt or not.

Table 25-8. Interrupt Mask Register (ME_IM) field descriptions

Field	Description
M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_SAFE	Safe mode interrupt mask 0 Safe mode interrupt is masked 1 Safe mode interrupt is enabled
M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

25.3.2.6 Invalid Mode Transition Status Register (ME_IMTS)

Address 0xC3FD_C014 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-7. Invalid Mode Transition Status Register (ME_IMTS)

This register provides the status bits for each cause of invalid mode interrupt.

Table 25-9. Invalid Mode Transition Status Register (ME_IMTS) field descriptions

Field	Description
S_MTI	Mode Transition Illegal status — This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is 1). Please refer to Section 25.4.5, Mode transition interrupts for the exceptions to this behavior. It is cleared by writing a 1 to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
S_MRI	Mode Request Illegal status — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a 1 to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
S_DMA	Disabled Mode Access status — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a 1 to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
S_NMA	Non-existing Mode Access status — This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a 1 to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
S_SEA	Safe Event Active status — This bit is set whenever the device is in Safe mode, Safe event bit is pending and a new mode requested other than Reset/Safe modes. It is cleared by writing a 1 to this bit. 0 No new mode requested other than Reset/Safe while Safe event is pending 1 New mode requested other than Reset/Safe while Safe event is pending

25.3.2.7 Debug Mode Transition Status Register (ME_DMTS)

Address 0xC3FD_C018 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	MPH_BUSY	0	0	PMC_PROG	CORE_DBG	0	0	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	FMPLL0_SC	FXOSC_SC	FIRC_SC	SSCLK_SC	SYSCLK_SW	DFLASH_SC	CFLASH_SC	CDP_PRPH_0_143	0	0	0	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-8. Debug Mode Transition Status Register (ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected.

NOTE

The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Table 25-10. Debug Mode Transition Status Register (ME_DMTS) field descriptions

Field	Description
MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator — This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet responded. It is cleared when the MC_PCU has responded. 0 Handshake is not busy 1 Handshake is busy
PMC_PROG	MC_PCU Mode Change in Progress indicator — This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all startup/shutdown processes have completed. 0 Startup/shutdown transition is not in progress 1 Startup/shutdown transition is in progress
CORE_DBG	Processor is in Debug mode indicator — This bit is set while the processor is in debug mode. 0 The processor is not in debug mode 1 The processor is in debug mode

Table 25-10. Debug Mode Transition Status Register (ME_DMTS) field descriptions (continued)

Field	Description
SMR	Safe mode request from MC_RGM is active indicator — This bit is set if a hardware Safe mode request has been triggered. It is cleared when the hardware Safe mode request has been cleared. 0 A Safe mode request is not active 1 A Safe mode request is active
FMPLL0_SC	FMPLL0 State Change during mode transition indicator — This bit is set when the primary frequency modulated phase locked loop is requested to change its startup/shutdown state. It is cleared when the primary frequency modulated phase locked loop has completed its state change. 0 No state change is taking place 1 A state change is taking place
FXOSC_SC	FXOSC State Change during mode transition indicator — This bit is set when the fast external crystal oscillator (4-16MHz) is requested to change its startup/shutdown state. It is cleared when the fast external crystal oscillator (4-16MHz) has completed its state change. 0 No state change is taking place 1 A state change is taking place
FIRC_SC	FIRC State Change during mode transition indicator — This bit is set when the fast internal RC oscillator (16MHz) is requested to change its startup/shutdown state. It is cleared when the fast internal RC oscillator (16MHz) has completed its state change. 0 No state change is taking place 1 A state change is taking place
SSCLK_SC	Secondary System Clock Sources State Change during mode transition indicator — This bit is set when a secondary system clock source is requested to change its startup/shutdown state. It is cleared when all secondary system clock sources have completed their state changes. (A secondary system clock source is a system clock source other than FIRC, FXOSC, or FMPLL0.) 0 No state change is taking place 1 A state change is taking place
SYSCLK_SW	System Clock Switching pending status — 0 No system clock source switching is pending 1 A system clock source switching is pending
DFLASH_SC	DFLASH State Change during mode transition indicator — This bit is set when the DFLASH is requested to change its startup/shutdown state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
CFLASH_SC	CFLASH State Change during mode transition indicator — This bit is set when the CFLASH is requested to change its startup/shutdown state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
CDP_PRPH_0_143	Clock Disable Process Pending status for Peripherals 0…143 — This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_96_127	Clock Disable Process Pending status for Peripherals 96…127 — This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table 25-10. Debug Mode Transition Status Register (ME_DMTS) field descriptions (continued)

Field	Description
CDP_PRPH_64_95	Clock Disable Process Pending status for Peripherals 64…95 — This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_32_63	Clock Disable Process Pending status for Peripherals 32…63 — This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_0_31	Clock Disable Process Pending status for Peripherals 0…31 — This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

25.3.2.8 Reset Mode Configuration Register (ME_RESET_MC)

Address 0xC3FD_C020 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-9. Invalid Mode Transition Status Register (ME_IMTS)

This register configures system behavior during Reset mode. Please refer to Table 25-11 for details.

25.3.2.9 Test Mode Configuration Register (ME_TEST_MC)

Address 0xC3FD_C024 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-10. Test Mode Configuration Register (ME_TEST_MC)

This register configures system behavior during Test mode. Please refer to Table 25-11 for details.

NOTE

Byte and half-word write accesses are not allowed to this register.

25.3.2.10 Safe Mode Configuration Register (ME_SAFE_MC)

Address 0xC3FD_C028 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-11. Safe Mode Configuration Register (ME_SAFE_MC)

This register configures system behavior during Safe mode. Please refer to Table 25-11 for details.

NOTE

Byte and half-word write accesses are not allowed to this register.

25.3.2.11 DRUN Mode Configuration Register (ME_DRUN_MC)

Address 0xC3FD_C02C Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-12. DRUN Mode Configuration Register (ME_DRUN_MC)

This register configures system behavior during DRUN mode. Please refer to Table 25-11 for details.

NOTE

Byte and half-word write accesses are not allowed to this register.

NOTE

The values of FXOSCON, FMPLL1ON, CFLAON and DFLAON are retained through Standby mode.

25.3.2.12 Run0…3 Mode Configuration Registers (ME_RUN0…3_MC)

Address 0xC3FD_C030–0xC3FD_C03C Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-13. Run0…3 Mode Configuration Registers (ME_RUN0…3_MC)

This register configures system behavior during Run0…3 modes. Please refer to Table 25-11 for details.

NOTE

Byte and half-word write accesses are not allowed to this register.

25.3.2.13 Halt Mode Configuration Register (ME_HALT_MC)

Address 0xC3FD_C040 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-14. Halt Mode Configuration Register (ME_HALT_MC)

This register configures system behavior during Halt mode. Please refer to Table 25-11 for details.

NOTE

Byte and half-word write accesses are not allowed to this register.

25.3.2.14 Stop Mode Configuration Register (ME_STOP_MC)

Address 0xC3FD_C048 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 25-15. Stop Mode Configuration Register (ME_STOP_MC)

This register configures system behavior during Stop mode. Please refer to Table 25-11 for details.

NOTE

Byte and half-word write accesses are not allowed to this register.

25.3.2.15 Standby Mode Configuration Register (ME_STANDBY_MC)

Address 0xC3FD_C054 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FMPLL1ON	FMPLL0ON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 25-16. Standby Mode Configuration Register (ME_STANDBY_MC)

This register configures system behavior during Standby mode. Please refer to Table 25-11 for details.

NOTE

Byte and half-word write accesses are not allowed to this register.

Table 25-11. Mode Configuration Registers (ME_<*mode*>_MC) field descriptions

Field	Description
PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In Safe/Test modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In Stop mode, only pad power sequence driver is disabled but the state of the output is kept. In Standby mode, power sequence driver and all pads except those mapped on wakeup lines are not powered and therefore high impedance. Wakeup line configuration remains unchanged.
MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 0 Main voltage regulator is switched off 1 Main voltage regulator is switched on
DFLAON	Data flash power-down control — This bit specifies the operating mode of the data flash after entering this mode. 00 Reserved 01 Data flash is in power-down mode 10 Data flash is in low-power mode 11 Data flash is in normal mode
CFLAON	Code flash power-down control — This bit specifies the operating mode of the program flash after entering this mode. 00 Reserved 01 Code flash is in power-down mode 10 Code flash is in low-power mode 11 Code flash is in normal mode
FMPLL1ON	secondary frequency modulated phase locked loop control 0 secondary frequency modulated phase locked loop is switched off 1 secondary frequency modulated phase locked loop is switched on
FMPLL0ON	primary frequency modulated phase locked loop control 0 primary frequency modulated phase locked loop is switched off 1 primary frequency modulated phase locked loop is switched on
FXOSCON	fast external crystal oscillator (4-16MHz) control 0 fast external crystal oscillator (4-16MHz) is switched off 1 fast external crystal oscillator (4-16MHz) is switched on
FIRCON	fast internal RC oscillator (16MHz) control 0 fast internal RC oscillator (16MHz) is switched off 1 fast internal RC oscillator (16MHz) is switched on

Table 25-11. Mode Configuration Registers (ME_<*mode*>_MC) field descriptions (continued)

Field	Description
SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16MHz int. RC osc. 0001 div. 16MHz int. RC osc. 0010 reserved 0011 div. 4-16MHz ext. osc. 0100 primary freq. mod. PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

25.3.2.16 Peripheral Status Register 0 (ME_PS0)

Address 0xC3FD_C060 Access: Supervisor read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_BAM	0	0	0	0	0	0	0	S_DMA_CH_MUX	0	0	0	0	0	S_FlexCAN1	S_FlexCAN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

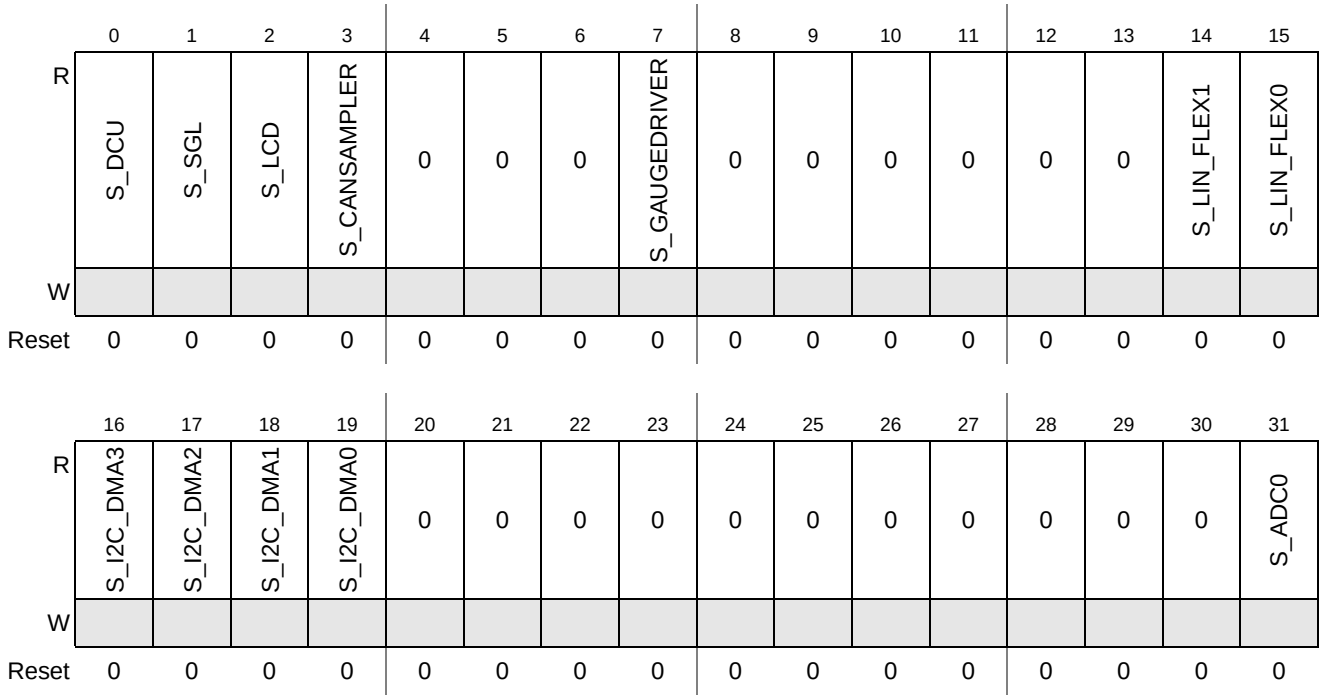
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_QUADSPI	0	0	0	S_QUADSPI	S_DSPI1	S_DSPI0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-17. Peripheral Status Register 0 (ME_PS0)

This register provides the status of the peripherals. Please refer to Table 25-12 for details.

25.3.2.17 Peripheral Status Register 1 (ME_PS1)

Address 0xC3FD_C064 Access: Supervisor read



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_DCU	S_SGL	S_LCD	S_CANSAMPLER	0	0	0	S_GAUGEDRIVER	0	0	0	0	0	0	S_LIN_FLEX1	S_LIN_FLEX0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_I2C_DMA3	S_I2C_DMA2	S_I2C_DMA1	S_I2C_DMA0	0	0	0	0	0	0	0	0	0	0	0	S_ADC0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-18. Peripheral Status Register 1 (ME_PS1)

This register provides the status of the peripherals. Please refer to Table 25-12 for details.

25.3.2.18 Peripheral Status Register 2 (ME_PS2)

Address 0xC3FD_C068 Access: Supervisor read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_PIT_RTI	S_RTC_API	S_MC_PCU	S_MC_RGM	S_MC_CGM	S_MC_ME	S_SSCM	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	S_CFLASH1	0	0	S_eMIOS1	S_eMIOS0	0	0	S_WKPU	S_SIUL	S_DFLASH0	S_CFLASH0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-19. Peripheral Status Register 2 (ME_PS2)

This register provides the status of the peripherals. Please refer to Table 25-12 for details.

25.3.2.19 Peripheral Status Register 3 (ME_PS3)

Address 0xC3FD_C06C Access: Supervisor read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	S_CMU0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-20. Peripheral Status Register 3 (ME_PS3)

This register provides the status of the peripherals. Please refer to Table 25-12 for details.

Table 25-12. Peripheral Status Registers 0…3 (ME_PS0…3) field descriptions

Field	Description
S_<*periph*>	Peripheral status — These bits specify the current status of the peripherals in the system. If no peripheral is mapped on a particular position, the corresponding bit is always read as 0. 0 Peripheral is frozen 1 Peripheral is active

25.3.2.20 Run Peripheral Configuration Registers (ME_RUN_PC0…7)

Address 0xC3FD_C080–0xC3FD_C09C Access: Supervisor read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-21. Run Peripheral Configuration Registers (ME_RUN_PC0…7)

These registers configure eight different types of peripheral behavior during run modes.

Table 25-13. Run Peripheral Configuration Registers (ME_RUN_PC0…7) field descriptions

Field	Description
RUN3	Peripheral control during Run3 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN2	Peripheral control during Run2 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN1	Peripheral control during Run1 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN0	Peripheral control during Run0 0 Peripheral is frozen with clock gated 1 Peripheral is active
DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active

Table 25-13. Run Peripheral Configuration Registers (ME_RUN_PC0…7) field descriptions (continued)

Field	Description
SAFE	Peripheral control during Safe 0 Peripheral is frozen with clock gated 1 Peripheral is active
TEST	Peripheral control during Test 0 Peripheral is frozen with clock gated 1 Peripheral is active
RESET	Peripheral control during Reset 0 Peripheral is frozen with clock gated 1 Peripheral is active

25.3.2.21 Low-Power Peripheral Configuration Registers (ME_LP_PC0…7)

Address 0xC3FD_C0A0–0xC3FD_C0BC Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STANDBY	0	0	STOP	0	HALT	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-22. Low-Power Peripheral Configuration Registers (ME_LP_PC0…7)

These registers configure eight different types of peripheral behavior during non-run modes.

Table 25-14. Low-Power Peripheral Configuration Registers (ME_LP_PC0…7) field descriptions

Field	Description
STANDBY	Peripheral control during Standby 0 Peripheral is frozen with clock gated 1 Peripheral is active
STOP	Peripheral control during Stop 0 Peripheral is frozen with clock gated 1 Peripheral is active
HALT	Peripheral control during Halt 0 Peripheral is frozen with clock gated 1 Peripheral is active

25.3.2.22 Peripheral Control Registers (ME_PCTL0…143)

Address 0xC3FD_C0C0–0xC3FD_C14F Access: Supervisor read/write

	0	1	2	3	4	5	6	7
R	0	DBG_F	LP_CFG			RUN_CFG		
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-23. Peripheral Control Registers (ME_PCTL0…143)

These registers select the configurations during run and non-run modes for each peripheral.

Table 25-15. Peripheral Control Registers (ME_PCTL0…143) field descriptions

Field	Description
DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the device mode. 1 Peripheral is frozen if not already frozen in device modes. **Note:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses.
LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

25.4 Functional description

25.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control ME_MCTL register. But in case of special events, mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- The first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- And the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<*mode*>_MC register. The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to Section 25.4.5, Mode transition interrupts.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as Run0…3 → Run0…3, DRUN → DRUN, Safe → Safe, and Test → Test are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set until the status in the ME_GS register matches the configuration programmed in the respective ME_<*mode*>_MC register.

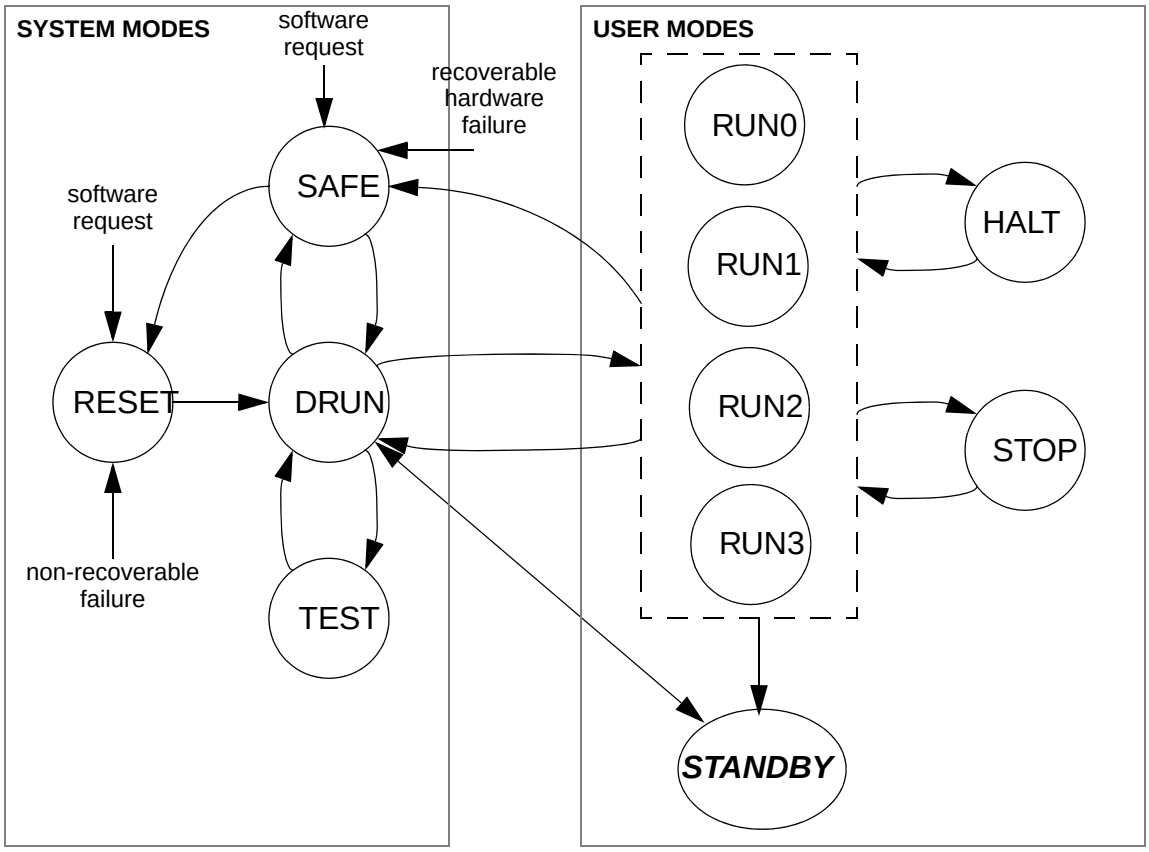


Figure 25-24. MC_ME mode diagram

25.4.2 Mode details

25.4.2.1 Reset mode

The device enters this mode whenever the system reset is asserted by the MC_RGM. Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. All power domains are made active during this mode.

25.4.2.2 DRUN mode

The device enters this mode on the following events.

- Automatically from Reset mode after completion of the reset sequence
- From Run0…3, Safe, or Test mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0011”
- From the Standby mode after an external wakeup event or internal wakeup alarm

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flashes, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16MHz int. RC osc. as the system clock.

This mode is intended to be used by software

- To initialize all registers as per the system needs
- To execute small routines in a ‘ping-pong’ with the Standby mode

When this mode is entered from Standby after a wakeup event, the ME_DRUN_MC register content is restored to its pre-Standby values, and the mode starts in that configuration.

All power domains are active when this mode is entered due to a system reset sequence initiated by a destructive reset event. In other cases of entry, such as the exit from Standby after a wakeup event, a functional reset event like an external reset or a software request from Run0…3, Safe, or Test mode, active power domains are determined by the power configuration register PCU_PCONF2 of the MC_PCU. All power domains except power domains #0 and #1 are configurable in this mode (see the MC_PCU chapter for details).

NOTE

As flashes can be configured in low-power or power-down state in this mode, software must ensure that the code executes from RAM before changing to this mode.

25.4.2.3 Safe mode

The device enters this mode on the following events:

- From DRUN, Run0…3, or Test mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0010”

- From any mode except Reset due to a Safe mode request generated by the MC_RGM because of some hardware failure in the system (see Chapter 31, Reset Generation Module (MC_RGM), for details)

As soon as any of the above events has occurred, a Safe mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16MHz int. RC osc. is selected as the system clock. All power domains are made active in this mode.

If the Safe mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the Safe mode regardless of other pending requests. In this case, the new mode request is not interpreted as an invalid request.

NOTE

If software requests to change to the Safe mode and then requests to change back to the parent mode before the mode transition is completed, the device's final mode after mode transition will be the parent mode. However, this is not recommended software behavior. It is recommended for software to wait until the S_MTRANS bit is cleared after requesting a change to Safe before requesting another mode change.

As long as a Safe event is active, the system remains in the Safe mode and no write access is allowed to the ME_MCTL register.

This mode is intended to be used by software

- To assess the severity of the cause of failure and then to either
  - — Re-initialize the device via the DRUN mode, or
  - — Completely reset the device via the Reset mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the PDO bit of the ME_SAFE_MC register should be set. In this case, the pads' power sequence driver cell is also disabled. The input levels remain unchanged.

NOTE

Peripherals that reside in auxiliary clock domains may be in an unknown state after exiting the Safe mode to enter the DRUN mode. Therefore execute a software reset while in the Safe mode if one or more peripherals present in auxiliary clock domains is required for further operation.

25.4.2.4 Test mode

The device enters this mode on the following events:

- From the DRUN mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0001"

As soon as any of the above events has occurred, a Test mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole

system can be stopped by programming the SYSCLK bit field to “1111”, and in this case, the only way to exit this mode is via a device reset.

This mode is intended to be used by software

- To execute on-chip test routines

All power domains except power domains #0 and #1 are configurable in this mode. Active power domains are determined by the power configuration register PCU_PCONF2 of the MC_PCU.

NOTE

As flash modules can be configured to a low-power or power-down state in these modes, software must ensure that the code will execute from RAM before it changes to this mode.

25.4.2.5 Run0…3 modes

The device enters one of these modes on the following events:

- From the DRUN another Run0…3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0100…0111”
- From the Halt mode by an interrupt event
- From the Stop mode by an interrupt or wakeup event

As soon as any of the above events occur, a Run0…3 mode transition request is generated. The mode configuration information for these modes is provided by ME_RUN0…3_MC registers. In these modes, the flashes, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- To execute application routines

All power domains except power domains #0 and #1 are configurable in these modes in order to reduce leakage consumption. Active power domains are determined by the power configuration register PCU_PCONF2 of the MC_PCU.

NOTE

As flash modules can be configured to a low-power or power-down state in these modes, software must ensure that the code will execute from RAM before it changes to this mode.

25.4.2.6 Halt mode

The device enters this mode on the following events:

- From one of the Run0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1000”.

As soon as any of the above events occur, a Halt mode transition request is generated. The mode configuration information for this mode is provided by ME_HALT_MC register. This mode is quite configurable, and the ME_HALT_MC register should be programmed according to the system needs. The

flashes can be put in power-down mode as needed. If there is a Halt mode request while an interrupt request is active, the device mode does not change, and an invalid mode interrupt is not generated.

This mode is intended as a first level low-power mode with

- The core clock frozen
- Only a few peripherals running

and to be used by software

- To wait until it is required to do something and then to react quickly (i.e. within a few system clock cycles of an interrupt event)

All power domains except power domains #0 and #1 are configurable in this mode in order to reduce leakage consumption. Active power domains are determined by the power configuration register PCU_PCONF2 of the MC_PCU.

NOTE

If Halt is configured with ME_HALT_MC[MVRON] = 0, ME_HALT_MC[FIRCON] = 0, and ME_HALT_MC[SYSCLK] = 0010/0011, the Main VREG will nevertheless remain enabled during the Halt mode if the previous Run[0..3] mode is configured with ME_RUN[0..3]_MC[FXOSCON] = 1.

25.4.2.7 Stop mode

The device enters this mode on the following events:

- From one of the Run0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1010”.

As soon as any of the above events occur, a Stop mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP_MC register. This mode is fully configurable, and the ME_STOP_MC register should be programmed according to the system needs. The FMPLL0 is switched off in this mode. The flashes can be put in power-down mode as needed. If there is a Stop mode request while any interrupt or wakeup event is active, the device mode does not change, and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- The core clock frozen
- Almost all peripherals stopped

and to be used by software

- To wait until it is required to do something with no need to react quickly (e.g. allow for system clock source to be re-started)

If the pads’ power sequence driver cell needs to be disabled while entering this mode, the PDO bit of the ME_STOP_MC register should be set. The state of the outputs is kept.

This mode can be used to stop all clock sources, thus preserving the device status. When exiting the Stop mode, the fast internal RC oscillator (16MHz) clock is selected as the system clock until the target clock is available.

All power domains except power domains #0 and #1 are configurable in this mode in order to reduce leakage consumption. Active power domains are determined by the power configuration register PCU_PCONF2 of the MC_PCU.

NOTE

If Halt is configured with ME_STOP_MC[MVRON] = 0, ME_STOP_MC[FIRCON] = 0, and ME_STOP_MC[SYSCLK] = 0010/0011, the Main VREG will nevertheless remain enabled during the Stop mode if the previous Run[0..3] mode is configured with ME_RUN[0..3]_MC[FXOSCON] = 1.

25.4.2.8 Standby mode

The device enters this mode on the following events:

- From the DRUN or one of the Run0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1101”.

As soon as any of the above events occur, a Standby mode transition request is generated. The mode configuration information for this mode is provided by the ME_STANDBY_MC register. In this mode, the power supply is turned off for most of the device. By default the only parts of the device that are still powered during this mode are pads mapped on wakeup lines and power domain #0 which contains the MC_RGM, MC_PCU, WKPU, 8K RAM, RTC_API, CANSampler, SIRC, FIRC, LCD, and device and user option bits. The FIRC can be optionally switched off. This is the lowest power consumption mode possible on the device.

This mode is intended as an extreme low-power mode with

- The core, the flashes, and almost all peripherals and memories powered down

and to be used by software

- To wait until it is required to do something with no need to react quickly (i.e. allow for system startup and system clock source to be re-started)

The exit sequence of this mode is similar to the reset sequence. If there is a Standby mode request while any wakeup event is active, the device mode does not change.

All power domains except power domain #0 are configurable in this mode in order to reduce leakage consumption. Active power domains are determined by the power configuration register PCU_PCONF2 of the MC_PCU.

25.4.3 Mode transition process

The process of mode transition follows the following steps in a pre-defined manner depending on the current device mode and the requested target mode. In many cases of mode transition, not all steps need

to be executed based on the mode control information, and some steps may not be valid according to the mode definition itself.

25.4.3.1 Target mode request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to Section 25.4.5, Mode transition interrupts, for details.

In the case of mode transitions occurring because of hardware events such as a reset, a Safe mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A Reset mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The Reset mode request has the highest priority, and the MC_ME is kept in the Reset mode during the entire reset sequence.

The Safe mode request has the next highest priority after reset which can be generated by software via the ME_MCTL register from all software running modes including DRUN, Run0…3, and Test or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

25.4.3.2 Target mode configuration loading

On completion of the Target mode request, the target mode configuration from the ME_<*target mode*>_MC register is loaded to start the resources (voltage sources, clock sources, flashes, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in Table 25-16. A '√' indicates that a given resource is configurable for a given mode.

Table 25-16. MC_ME resource control overview

Resource	Mode							
	Reset	Test	Safe	DRUN	Run0…3	Halt	Stop	Standby
FIRC	on	√ always on	on	on	on	√ always on	√ always on	√ on
FXOSC	off	√ off	off	√ off	√ off	√ off	√ off	off by default, but also writable
FMPLL0	off	√ off	off	√ off	√ off	√ off	off	off

Table 25-16. MC_ME resource control overview (continued)

Resource	Mode							
	Reset	Test	Safe	DRUN	Run0…3	Halt	Stop	Standby
FMPLL1	off	√ off	off	√ off	√ off	√ off	√ off	off
CFLASH	normal	√ normal	normal	√ normal	√ normal	√ low-power	√ power-down	power-down
DFLASH	normal	√ normal	normal	√ normal	√ normal	√ low-power	√ power-down	power-down
MVREG	on	on	on	on	on	√ on	√ on	off
PDO	off	√ off	√ on	off	off	off	√ off	on

25.4.3.3 Peripheral Clocks Disable

On completion of the Target mode request, the MC_ME requests each peripheral to enter its stop mode when:

- The peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0…7 and ME_LP_PC0…7, and the peripheral control registers ME_PCTL0…143

WARNING

The MC_ME does not automatically request peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. Therefore, it is software's responsibility to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a Safe mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The Safe mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to Section 25.4.6, Peripheral clock gating for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the device enters the Safe mode.

25.4.3.4 Processor Low-Power mode entry

If, on completion of the Peripheral Clocks Disable, the mode transition is to the Halt mode, the MC_ME requests the processor to enter its halted state. The processor acknowledges its halt state request after completing all outstanding bus transactions.

If, on completion of the Peripheral Clocks Disable, the mode transition is to the Stop or Standby mode, the MC_ME requests the processor to enter its stopped state. The processor acknowledges its stop state request after completing all outstanding bus transactions.

25.4.3.5 Processor and system memory clock disable

If, on completion of the Processor Low-Power mode entry, the mode transition is to the Halt, Stop, or Standby mode and the processor is in its appropriate halted or stopped state, the MC_ME disables the processor and system memory clocks to achieve further power saving.

The clocks to the processor and system memories are unaffected for all transitions between software running modes including DRUN, Run0…3, and Safe.

WARNING

Clocks to the whole device including the processor and system memories can be disabled in Test mode.

25.4.3.6 Clock sources switch-on

On completion of the Processor Low-Power mode entry, the MC_ME controls all clock sources that affect the system clock based on the <*clock source*>ON bits of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers. The following system clock sources are controlled at this step:

- The fast internal RC oscillator (16MHz)
- The fast external crystal oscillator (4-16MHz)
- The secondary frequency modulated phase locked loop

NOTE

The primary frequency modulated phase locked loop, which needs the main voltage regulator to be stable, is not controlled by this step.

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these system clocks is updated in the S_<*clock source*> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

25.4.3.7 Main Voltage Regulator Switch-On

On completion of the Target mode request, if the main voltage regulator needs to be switched on from its off state based on the MVRON bit of the ME_<*current mode*>_MC and ME_<*target mode*>_MC

registers, the MC_ME requests the MC_PCU to turn on the regulator and waits for the output voltage stable status in order to update the S_MVR bit of the ME_GS register.

This step is required only during the exit of the low-power modes Halt and Stop. In this step, the fast internal RC oscillator (16MHz) is switched on regardless of the target mode configuration, as the main voltage regulator requires the 16MHz int. RC osc. during startup in order to generate the voltage status.

During the Standby exit sequence, the MC_PCU alone manages the startup of the main voltage regulator, and the MC_ME is kept in Reset or shut off (depending on the power domain #1 status).

25.4.3.8 Flash Modules Switch-On

On completion of the Main Voltage Regulator Switch-On, if a flash module needs to be switched to normal mode from its low-power or power-down mode based on the CFLAON and DFLAON bit fields of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, the MC_ME requests the flash to exit from its low-power/power-down mode. When the flash modules are available for access, the S_CFLA and S_DFLA bit fields of the ME_GS register are updated to “11” by hardware.

If the main regulator is also off in device low-power modes, then during the exit sequence, the flash is kept in its low-power state and is switched on only when the Main Voltage Regulator Switch-On process has completed.

WARNING

It is illegal to switch the flashes from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

25.4.3.9 FMPLL0 Switch-On

On completion of the Clock sources switch-on and Main Voltage Regulator Switch-On, if the FMPLL0 is to be switched on from the off state based on the FMPLL0ON bit of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, the MC_ME requests the FMPLL0 digital interface to start the phase locking process and waits for the FMPLL0 to enter into the locked state. When the FMPLL0 enters the locked state and starts providing a stable output clock, the S_FMPLL0 bit of ME_GS register is set.

25.4.3.10 Power Domain #2 Switch-On

On completion of the Main Voltage Regulator Switch-On, the MC_ME indicates a mode change to the MC_PCU. The MC_PCU then determines whether a startup sequence is required for power domain #2. Only after the MC_PCU has executed all required startups does the MC_ME complete the mode transition.

25.4.3.11 Pad Outputs-On

On completion of the Main Voltage Regulator Switch-On, if the PDO bit of the ME_<*target mode*>_MC register is cleared, then

- All pad outputs are enabled to return to their previous state
- The I/O pads power sequence driver is switched on

25.4.3.12 Peripheral Clocks Enable

Based on the current and target device modes, the peripheral configuration registers ME_RUN_PC0…7, ME_LP_PC0…7, and the peripheral control registers ME_PCTL0…143, the MC_ME enables the clocks for selected modules as required. This step is executed only after the Main Voltage Regulator Switch-On process is completed.

Also if a mode change translates to a startup of one or more power domains, the MC_PCU indicates the MC_ME after completing the startup sequence upon which the MC_ME may assert the peripheral clock enables of the peripherals residing in those power domains.

25.4.3.13 Processor and memory clock enable

If the mode transition is from any of the low-power modes Halt or Stop to Run0…3, the clocks to the processor and system memories are enabled. The process of enabling these clocks is executed only after the Flash Modules Switch-On process is completed.

25.4.3.14 Processor Low-Power mode exit

If the mode transition is from any of the low-power modes Halt, Stop, or Standby to Run0…3, the MC_ME requests the processor to exit from its halted or stopped state. This step is executed only after the Processor and memory clock enable process is completed.

25.4.3.15 System clock switching

Based on the SYSCLK bit field of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16MHz int. RC osc. is effective only when the S_FIRC bit of the ME_GS register is set by hardware (i.e. the fast internal RC oscillator (16MHz) has stabilized).
- The target clock configuration for the div. 16MHz int. RC osc. is effective only when the S_FIRC bit of the ME_GS register is set by hardware (i.e. the fast internal RC oscillator (16MHz) has stabilized).
- The target clock configuration for the div. 4-16MHz ext. osc. is effective only when the S_FXOSC bit of the ME_GS register is set by hardware (i.e the fast external crystal oscillator (4-16MHz) has stabilized).
- The target clock configuration for the primary freq. mod. PLL is effective only when the S_FMPLL0 bit of the ME_GS register is set by hardware (i.e. the primary frequency modulated phase locked loop has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with “1111”. This is possible only in the Stop and Test modes. In the Standby mode, the clock configuration is fixed, and the system clock is automatically forced to 0.

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- The Clock sources switch-on process has completed if the target system clock source needs to be switched on
- The FMPLL0 Switch-On process has completed if the target system clock is the primary freq. mod. PLL
- The Peripheral Clocks Disable process is completed in order not to change the system clock frequency before peripherals close their internal activities

An overview of system clock source selection possibilities for each mode is shown in Table 25-17. A '√' indicates that a given clock source is selectable for a given mode.

Table 25-17. MC_ME system clock selection overview

System Clock Source	Mode							
	Reset	Test	Safe	DRUN	Run0…3	Halt	Stop	Standby
16MHz int. RC osc.	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	
div. 16MHz int. RC osc.		√		√	√	√	√	
div. 4-16MHz ext. osc.		√		√	√	√	√	
primary freq. mod. PLL		√		√	√	√		
system clock is disabled		√[1]					√	√ (default)

[1] Disabling the system clock during Test mode will require a reset in order to exit Test mode

25.4.3.16 Power Domain #2 Switch-Off

Based on the device mode and the MC_PCU's power configuration register PCU_PCONF2, the power domain #2 is controlled by the MC_PCU.

If a mode change translates to a power-down of the power domain, then the MC_PCU starts the power-down sequence. The MC_PCU acknowledges the completion of the power-down sequence with respect to the new mode, and the MC_ME uses this information to update the mode transition status. This step is executed only after the Peripheral Clocks Disable process has completed.

25.4.3.17 Pad Switch-Off

If the PDO bit of the ME_<*target mode*>_MC register is 1 then

- The outputs of the pads are forced to the high impedance state if the target mode is Safe or Test

- I/O pads power sequence driver is switched off if the target mode is one of Safe, Test, or Stop modes

In Standby mode, the power sequence driver and all pads except the external reset and those mapped on wakeup lines are not powered and therefore high impedance. The wakeup line configuration remains unchanged.

This step is executed only after the Peripheral Clocks Disable process is completed.

25.4.3.18 FMPLL0 Switch-Off

Based on the FMPLL0ON bit of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, if FMPLL0 is to be switched off, the MC_ME requests the FMPLL0 to power down and updates its availability status bit S_FMPLL0 of the ME_GS register to 0. This step is executed only after the System clock switching process is completed.

25.4.3.19 Clock Sources Switch-Off

Based on the device mode and the <*clock source*>ON bits of the ME_<*mode*>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<*clock source*> of the ME_GS register to 0.

This step is executed only after

- System clock switching process is completed in order not to lose the current system clock during mode transition.
- FMPLL0 Switch-Off as the input reference clock of the FMPLL0 can be among these clock sources. This is needed to prevent an unwanted lock transition when the FMPLL0 is switched on.

25.4.3.20 Flash Switch-Off

Based on the CFLAON and DFLAON bit fields of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, if any of the flash modules is to be put in a low-power state, the MC_ME requests the flash to enter the corresponding low-power state and waits for the deassertion of flash ready status signal. The exact low-power mode status of the flash modules is updated in the S_CFLA and S_DFLA bit fields of the ME_GS register. This step is executed only when Processor and system memory clock disable process is completed.

25.4.3.21 Main Voltage Regulator Switch-Off

Based on the MVRON bit of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, if the main voltage regulator is to be switched off, the MC_ME requests it to power down and clears the availability status bit S_MVR of the ME_GS register.

This step is required only during the entry of low-power modes like Halt and Stop. This step is executed only after completing the following processes:

- FMPLL0 Switch-Off
- Flash Switch-Off

- Power Domain #2 Switch-Off
- Power Domain #2 Switch-On
- The device consumption is less than the pre-defined threshold value (i.e. the S_DC bit of the ME_GS register is 0).

If the target mode is Standby, the main voltage regulator is not switched off by the MC_ME and the Standby request is asserted after the above processes have completed upon which the MC_PCU takes control of the main regulator. As the MC_PCU needs the 16MHz int. RC osc., the fast internal RC oscillator (16MHz) remains active until all the Standby steps are executed by the MC_PCU after which it may be switched off depending on the FIRCON bit of the ME_STANDBY_MC register.

25.4.3.22 Current mode update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when:

- All the updated status bits in the ME_GS register match the configuration specified in the ME_<*target mode*>_MC register
- Power sequences are done
- Clock disable/enable process is finished
- Processor low-power mode (halt/stop) entry and exit processes are finished

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

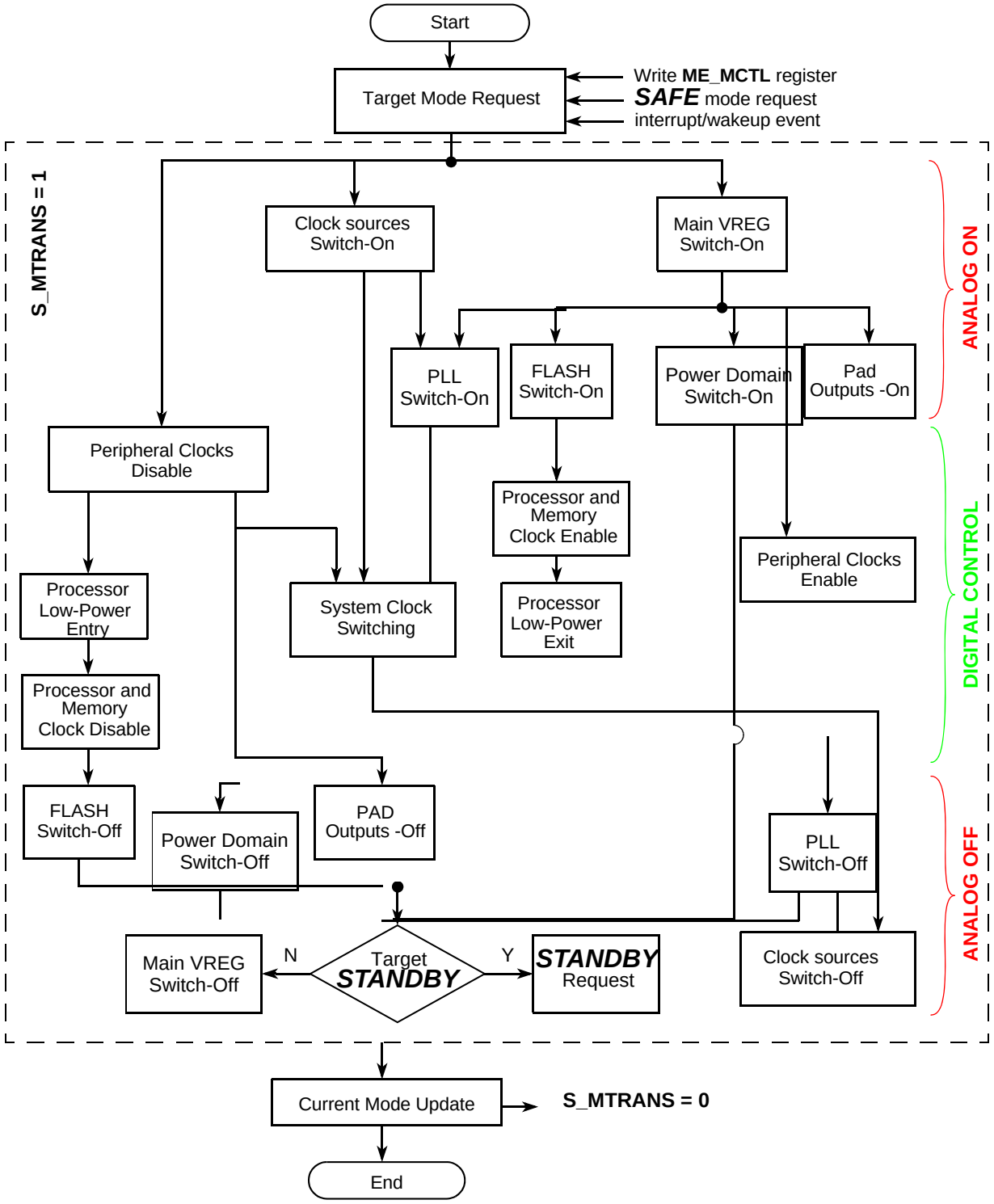


Figure 25-25. MC_ME Transition Diagram

25.4.4 Protection of mode configuration registers

While programming the mode configuration registers ME_<*mode*>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- FIRC must be on if the system clock is one of the following:
  - 16MHz int. RC osc.
  - div. 16MHz int. RC osc.
- FXOSC must be on if the system clock is one of the following:
  - div. 4-16MHz ext. osc.

NOTE

Software must ensure to switch on the clock source that provides the input reference clock to the FMPLL0. There is no automatic protection mechanism to check this in the MC_ME.

- FMPLL0 must be on if the system clock is the primary freq. mod. PLL.
- Configuration “00” for the CFLAON and DFLAON bit fields are reserved.
- MVREG must be on if any of the following is active:
  - FMPLL0
  - CFLASH
  - DFLASH
- System clock configurations marked as reserved may not be selected.
- Configuration “1111” for the SYSCLK bit field is allowed only for the Stop and Test modes, and only in this case may all system clock sources be turned off.

WARNING

If the system clock is stopped during Test mode, the device can exit only via a system reset.

25.4.5 Mode transition interrupts

The following are the three interrupts related to mode transition implemented in the MC_ME.

25.4.5.1 Invalid mode configuration interrupt

Whenever a write operation is attempted to the ME_<*mode*>_MC registers violating the protection rules mentioned in the Section 25.4.4, Protection of mode configuration registers, the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of ME_IM register is 1.

25.4.5.2 Invalid mode transition interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the Safe mode and the Safe mode request from MC_RGM is active, and if the target mode requested is other than Reset or Safe, then this new mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.
- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (i.e. a non existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the device modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition is in progress (the S_MTRANS bit of the ME_GS register is 1), the mode transition illegal status bit S_MTI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

NOTE

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the Reset or Safe mode irrespective of the mode transition status.
- As the exit of Halt and Stop modes depends on the interrupts of the system which can occur at any instant, these requests to return to Run0…3 modes are always valid.
- In order to avoid any unwanted lockup of the device modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined ‘reasonable’ amount of time for whatever reason. The parent mode is the device mode before a valid mode request was made.
- Self-transition requests (e.g. Run0 → Run0) are not considered as invalid even when the mode transition process is active (i.e. S_MTRANS is 1). During the low-power mode exit process, if the system is not able to enter the respective Run0…3 mode properly (i.e. all status bits of the ME_GS register match with configuration bits in the ME_<*mode*>_MC register), then software can only request the Safe or Reset mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE is ME_IM register is 1.

25.4.5.3 Safe mode transition interrupt

Whenever the system enters the Safe mode as a result of a Safe mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is 1.

The Safe mode interrupt pending bit can be cleared only when the Safe mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a Safe mode request). If the system is already in Safe mode, any new Safe mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the Safe mode interrupt pending bit is not set when the Safe mode is entered by a software request (i.e. programming of ME_MCTL register).

25.4.5.4 Mode transition complete interrupt

Whenever the system completes a mode transition fully (i.e. the S_MTRANS bit of ME_GS register transits from 1 to 0), the interrupt pending bit I_MTC of the ME_IS register is set, and interrupt request is generated if the mask bit M_MTC of the ME_IM register is 1. The interrupt bit I_MTC is not set when entering low-power modes Halt and Stop in order to avoid the same event requesting the exit of these low-power modes.

25.4.6 Peripheral clock gating

During all device modes, certain peripherals can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0…7 are chosen only during the software running modes DRUN, Test, Safe, and Run0…3. All configurations are programmable by software according to the needs of application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTL0…143 registers.

The low-power peripheral configuration registers ME_LP_PC0…7 are chosen only during the low-power modes Halt, Stop, and Standby. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTL0…143 registers.

Any modifications to the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTL0…143 registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the processor enters a debug session during any mode, the following occurs for the peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTL0…143 register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits. Any further modifications of the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTL0…143 registers during a debug session will take effect immediately without requiring any new mode request.

25.4.7 Application Example

Figure 25-26 shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

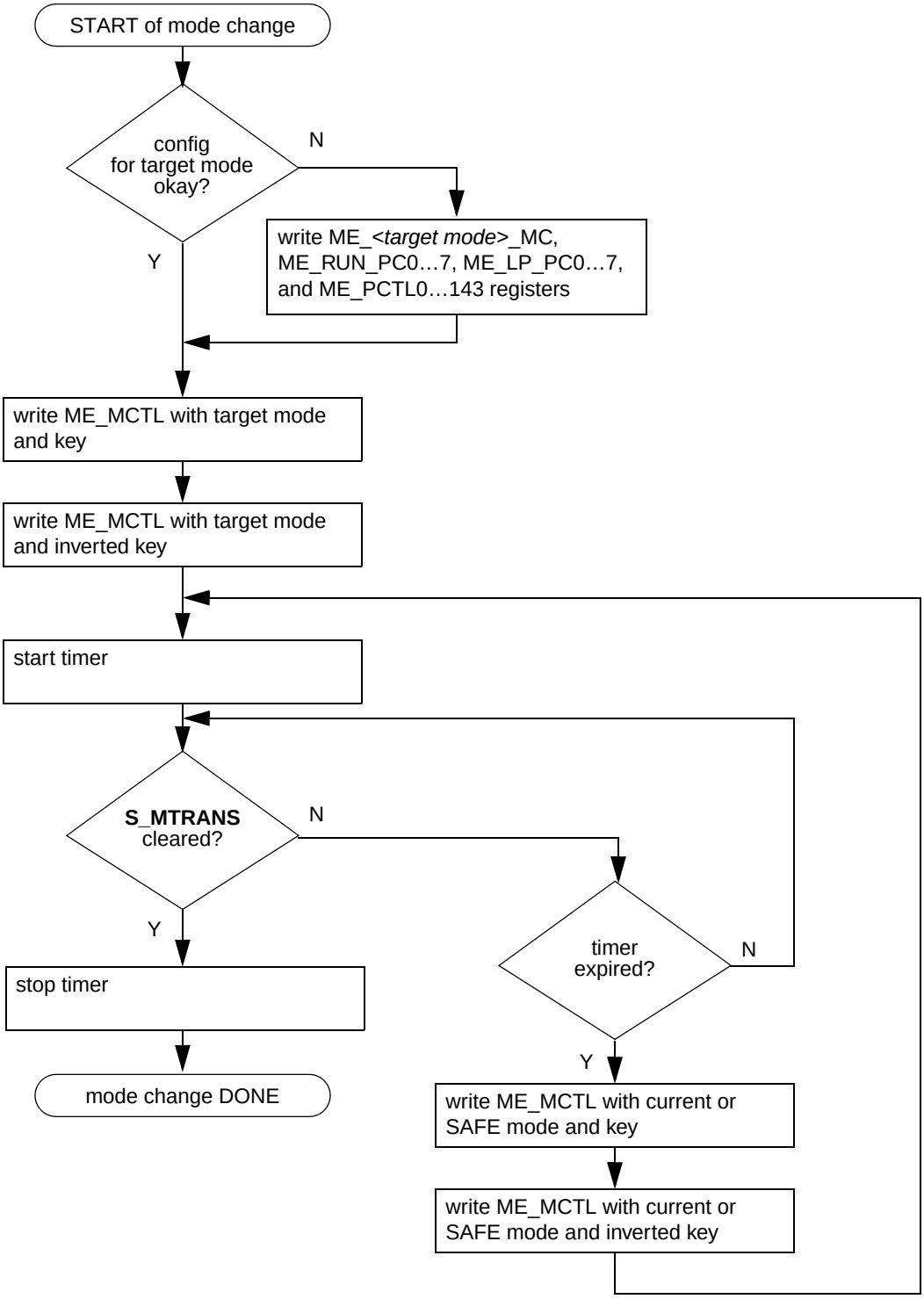


Figure 25-26. MC_ME Application Example Flow Diagram

Chapter 26
Nexus Development Interface (NDI)

26.1 Introduction

The Nexus Development Interface (NDI) block provides real-time development support capabilities for the MPC5606S MCU in compliance with the IEEE-ISTO 5001-2003 standard. This development support is supplied for MCUs without requiring external address and data pins for internal visibility.

The NDI block is an integration of several individual Nexus blocks that are selected to provide the development support interface for MPC5606S.

The NDI block interfaces to the e200z0 and internal buses to provide development support as per the IEEE-ISTO 5001-2003 standard. The development support provided includes program trace, watchpoint messaging, ownership trace, watchpoint triggering, processor overrun control, run-time access to the MCU's internal memory map, and access to the e200z0 internal registers during halt, all via the JTAG port.

26.2 Block diagram

Figure 26-1 shows a functional block diagram of the NDI.

A simplified block diagram of the NDI illustrates the functionality and interdependence of major blocks (see Figure 26-2) and how the individual Nexus blocks are combined to form the NDI.

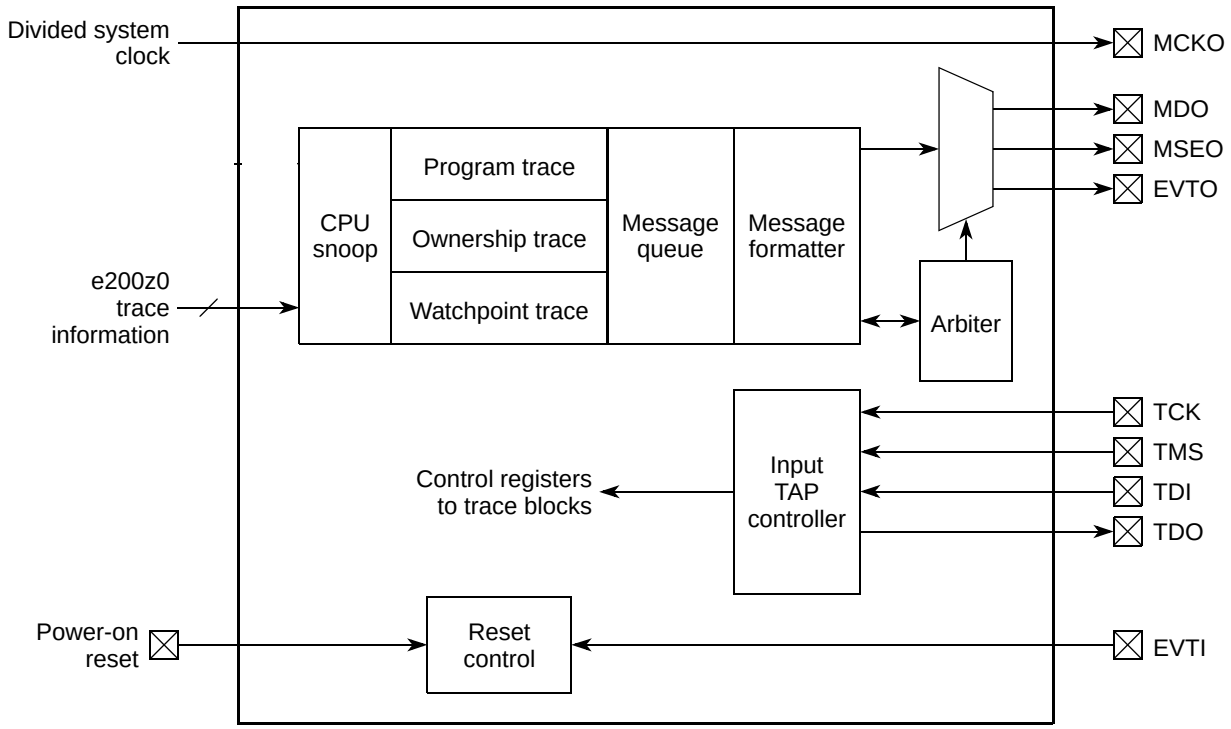


Figure 26-1. NDI functional block diagram

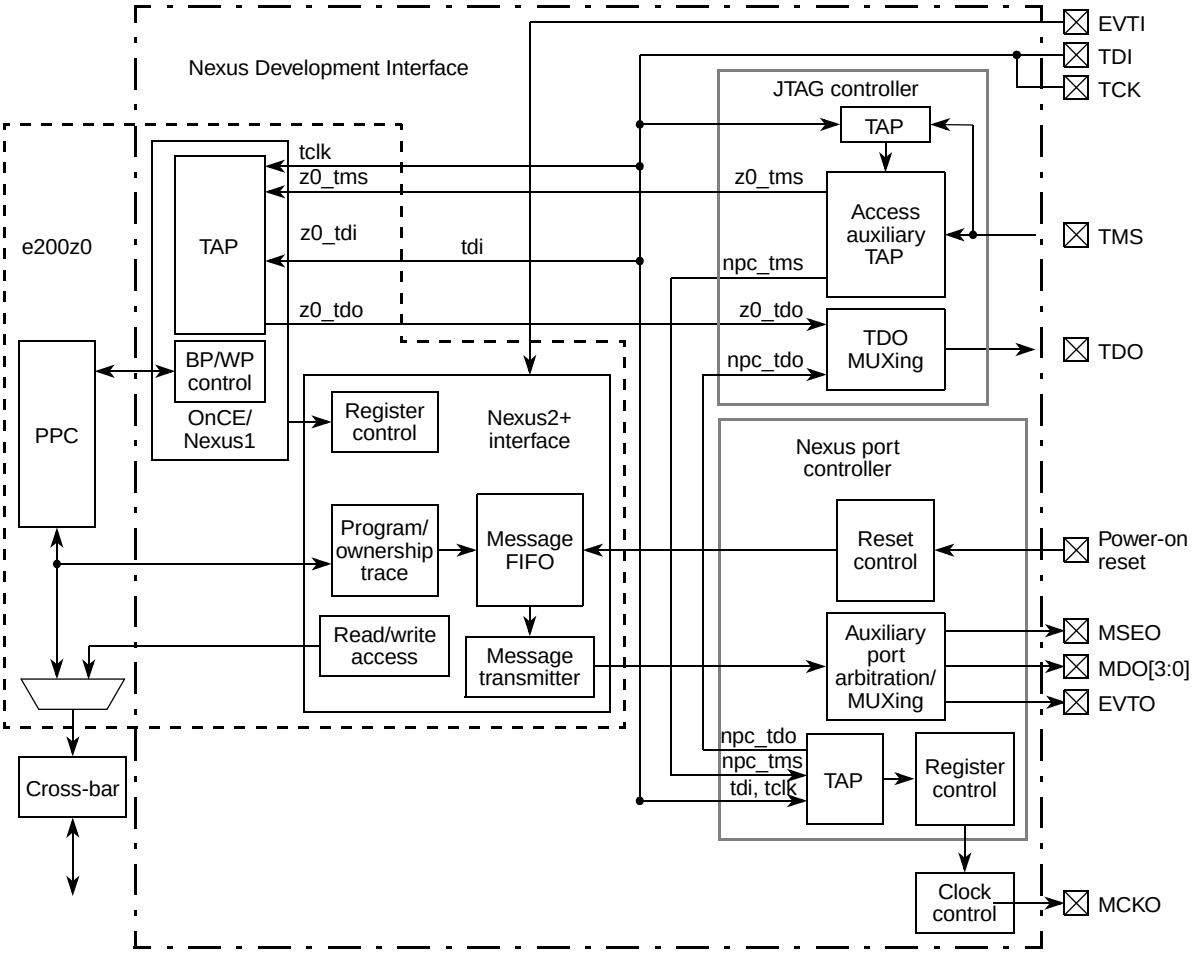


Figure 26-2. NDI implementation block diagram

26.3 Features

The NDI module of the MPC5606S is compliant with Class 2 of the IEEE-ISTO 5001-2003 standard, with additional Class 3 and Class 4 features available.The following features are implemented:

- Program trace via branch trace messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Ownership trace via ownership trace messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An ownership trace message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Watchpoint messaging via the auxiliary pins.
- Watchpoint trigger enable of program trace messaging.
- Auxiliary interface for higher data input/output.
  - — Four message data out pins.

  - One message start/end out pins (MSEO).
  - One watchpoint event pin (EVTO).
  - One event-in pin (EVTI).
  - One message clock out pin (MCKO).
  - Four pin JTAG port (TDI, TDO, TMS, and TCK).
- Registers for program trace, ownership trace, and watchpoint trigger.
- All features controllable and configurable via the JTAG port.
- Run-time access to the on-chip memory map via the Nexus read/write access protocol. This allows for enhanced download/upload capabilities.
- All features are independently configurable and controllable via the IEEE 1149.1 I/O port.
- The NDI block reset is controlled with JCOMP, power-on reset, and the TAP state machine. All these sources are independent of system reset.
- Support for internal censorship mode to prevent external access to flash memory contents when censorship is enabled.

NOTE

If the e200z0 core has executed a wait instruction, then the Nexus2+ controller clocks are gated off. While the core is in this state, it is not possible to perform Nexus read/write operations.

26.4 Modes of operation

The NDI block is in reset when the TAP controller state machine is in the Test-Logic-Reset state. The Test-Logic-Reset state is entered on the assertion of the power-on reset signal or through state machine transitions controlled by TMS. Ownership of the TAP is achieved by loading the appropriate enable instruction for the desired Nexus client in the JTAGC controller (JTAGC) block.

The NPC transitions out of the reset state immediately following negation of power-on reset.

26.4.1 Nexus Reset

In Nexus reset mode, the following actions occur:

- Register values default back to their reset values.
- The message queues are marked as empty.
- The auxiliary output port pins are negated if the NDI controls the pads.
- The TDO output buffer is disabled if the NDI has control of the TAP.
- The TDI, TMS, and TCK inputs are ignored.
- The NDI block indicates to the MCU that it is not using the auxiliary output port. This indication can be used to tristate the output pins or use them for another function.

26.4.2 Operating mode

In Full-Port mode, all available MDO pins are used to transmit messages. All trace features are enabled or can be enabled by writing the configuration registers via the JTAG port. Four MDO pins are available in Full-Port mode.

26.4.2.1 Disabled-Port mode

In Disabled-Port mode, message transmission is disabled. Any debug feature that generates messages cannot be used. The primary features available are class 1 features and read/write access.

26.4.2.2 Censored mode

The NDI supports internal flash censorship mode by preventing the transmission of trace messages and Nexus access to memory-mapped resources when censorship is enabled.

26.4.2.3 Stop mode

Stop mode logic is implemented in the Nexus port controller (NPC). When a request is made to enter Stop mode, the NDI block completes monitoring of any pending bus transaction, transmits all messages already queued, and acknowledges the stop request. After the acknowledgment, the system clock inputs are shut off by the clock driver on the device. While the clocks are shut off, the development tool cannot access NDI registers via the JTAG port.

26.5 External signal description

All the signals are available in the 208BGA without any multiplexing scheme. Refer to Chapter 3, Signal Description, for details.

26.5.1 Nexus signal reset states

Table 26-1. NDI signal reset state

Name	Function	Nexus reset state	Pull
EVTI	Event-in pin	—	Up
EVTO	Event-out pin	0b1	—
MCKO	Message clock out pin	0b0	—
MDO[3:0]	Message data out pins	0	—
MSEO	Message start/end out pin	0b1	—

26.6 Memory map and register description

The NDI block contains no memory-mapped registers. Nexus registers are accessed by a development tool via the JTAG port using a client-select value and a register index. OnCE registers are accessed by loading the appropriate value in the RS[0:6] field of the OnCE command register (OCMD) via the JTAG port.

26.6.1 Nexus Debug Interface (NDI) registers

Table 26-2 shows the NDI registers by client select and index values. OnCE register addressing is documented in Chapter 19, IEEE 1149.1 Test Access Port Controller (JTAGC).

Table 26-2. NDI registers

Client Select	Index	Register
Client-independent registers		
0bxxxx	0	Device ID (DID)[1]
0bxxxx	127	Port configuration register (PCR)[1]
e200z0 control/status registers		
0b0000	2	e200z0 development control1 (DC1)
0b0000	3	e200z0 development control2 (DC2)
0b0000	4	e200z0 development status (DS)
0b0000	7	Read/write access control/status (RWCS)
0b0000	9	Read/write access address (RWA)
0b0000	10	Read/write access data (RWD)
0b0000	11	e200z0 watchpoint trigger (PPC_WT)

1 Implemented in NPC block. All other registers implemented in e200z0 Nexus2+ block.

26.6.2 Register description

This section lists the NDI registers and describes the registers and their bit fields.

26.6.2.1 Nexus Device ID Register (DID)

The NPC device identification register, shown in Figure 26-3, allows the part revision number, design center, part identification number, and manufacturer identity code of the device to be determined through the auxiliary output port, and serially through TDO. This register is read-only.

Reg Index: 0 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Part Revision Number[1]				Design Center						Part Identification Number					
W																
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Part Identification Number (*continued*)				Manufacturer Identity Code											1
W																
Reset	?	?	?	?	0	0	0	0	0	1	0	0	0	0	0	1

Figure 26-3. Nexus Device ID Register (DID)

[1] Part Revision Number default value is 0x0 for the device's initial mask set and changes for each mask set revision.

Table 26-3. DID field descriptions

Field	Description
0–3 PRN	Part Revision Number. Contains the revision number of the part. This field changes with each revision of the device or module.
4–9 DC	Design center.
10–19 PIN	Part identification number. Contains the part number of the device.
20–30 MIC	Manufacturer identity code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID: 0x20.
31	Fixed Per JTAG 1149.1. Always set to 1.

26.6.2.2 Port Configuration Register (PCR)

The PCR is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NDI is enabled.

The PCR register may be rewritten by the debug tool subsequent to the enabling of the NPC for low-power debug support. In this case, the debug tool may set and clear the LP_DBG_EN, SLEEP_SYNC, and STOP_SYNC bits, but must preserve the original state of the remaining bits in the register.

NOTE

The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Reg Index: 127 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FPM	MCKO_GT	MCKO_EN	MCKO_DIV			EVT_EN	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LP_DBG_EN	0	0	0	0	0	SLEEP_SYNC	STOP_SYNC	0	0	0	0	0	0	0	PSTAT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-4. Port Configuration Register (PCR)

Table 26-4. PCR field descriptions

Field	Description
FPM	Full Port Mode. The value of the FPM bit determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages. 0 A subset of MDO pins is used to transmit messages. 1 All MDO pins are used to transmit messages.
MCKO_GT	MCKO Clock Gating Control. This bit is used to enable or disable MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted. 0 MCKO gating is disabled. 1 MCKO gating is enabled.
MCKO_EN	MCKO Enable. This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field. 0 MCKO clock is driven to zero. 1 MCKO clock is enabled.

Table 26-4. PCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>MCKO_DIV</td><td>MCKO Division Factor. The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. In this table, SYS_CLK represents the system clock frequency.
<table>
<tr><th>MCKO_DIV[2:0]</th><th>MCKO Frequency</th></tr>
<tr><td>0b000</td><td>SYSCLK÷1</td></tr>
<tr><td>0b001</td><td>SYSCLK÷2</td></tr>
<tr><td>0b010</td><td>Reserved</td></tr>
<tr><td>0b011</td><td>SYS_CLK÷4</td></tr>
<tr><td>0b100</td><td>Reserved</td></tr>
<tr><td>0b101</td><td>Reserved</td></tr>
<tr><td>0b110</td><td>Reserved</td></tr>
<tr><td>0b111</td><td>SYS_CLK÷8</td></tr>
</table>
Note: If the Nexus clock divider (NPC_PCR[MCKO_DIV]) is set to 8 and the Nexus clock gating control (NPC_PCR[MCKO_GT]) is enabled, the Nexus clock (MCKO) will be disabled prior to the completion of transmission of the Nexus message data. Do not enable the automatic clock gating mode when the Nexus clock divider is set to 8.</td></tr>
<tr><td>EVT_EN</td><td>EVTO/EVTI Enable. This bit enables the EVTO/EVTI port functions.
0 EVTO/EVTI port disabled.
1 EVTO/EVTI port enabled.</td></tr>
<tr><td>LP_DBG_EN</td><td>Low-Power Debug Enable. The LP_DBG_EN bit enables debug functionality to support entry and exit from low-power sleep and Stop modes.
0 Low-power debug disabled.
1 Low-power debug enabled.</td></tr>
<tr><td>SLEEP_SYNC</td><td>Sleep Mode Synchronization. The SLEEP_SYNC bit is used to synchronize the entry into sleep mode between the device and the debug tool. The device sets this bit before a pending entry into sleep mode. After reading SLEEP_SYNC as set, the debug tool then clears SLEEP_SYNC to acknowledge to the device that it may enter sleep mode.
0 Sleep mode entry acknowledge.
1 Sleep mode entry pending.</td></tr>
<tr><td>STOP_SYNC</td><td>Stop Mode Synchronization. The STOP_SYNC bit is used to synchronize the entry into Stop mode between the device and the debug tool. The device sets this bit before a pending entry into Stop mode. After reading STOP_SYNC as set, the debug tool then clears STOP_SYNC to acknowledge to the device that it may enter Stop mode.
0 Stop mode entry acknowledge.
1 Stop mode entry pending</td></tr>
<tr><td>PSTAT_EN</td><td>Processor Status Mode Enable.</td></tr>
</table>

26.6.2.3 Development Control Register 1, 2 (DC1, DC2)

The development control registers are used to control the basic development features of the Nexus module. Figure 26-5 shows development control register 1 and Table 26-5 describes the register's fields.

Nexus Reg: 0x0002 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OPC	MCK_DIV			EOC		0	PTM	WEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									OVC			EIC	TM			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-5. Development Control Register 1 (DC1)

Table 26-5. DC1 field descriptions

Field	Description
0 OPC[1]	Output Port Mode Control. 0 Reduced-Port mode configuration (2 MDO pins). 1 Full-Port mode configuration (4 MDO pins).
1–2 MCK_DIV[1:0][1]	MCKO Clock Divide Ratio (see note 1). 00 MCKO is 1× processor clock frequency. 01 MCKO is 1/2× processor clock frequency. 10 MCKO is 1/4× processor clock frequency. 11 MCKO is 1/8× processor clock frequency.
3–4 EOC[1:0]	EVTO Control. 00 EVTO upon occurrence of watchpoints (configured in DC2). 01 EVTO upon entry into debug mode. 10 EVTO upon timestamping event. 11 Reserved.
5	Reserved
6 PTM	Program Trace Method. 0 Program trace uses traditional branch messages. 1 Program trace uses branch history messages.
7 WEN	Watchpoint Trace Enable. 0 Watchpoint messaging disabled. 1 Watchpoint messaging enabled.
8–23	Reserved.
24–26 OVC[2:0]	Overrun Control. 000 Generate overrun messages. 001–010 Reserved. 011 Delay processor for BTM / DTM / OTM overruns. 1XX Reserved.

Table 26-5. DC1 field descriptions (continued)

Field	Description
27–28 EIC[1:0]	EVTI Control. 00 EVTI is used for synchronization (program trace/data trace). 01 EVTI is used for debug request. 1X Reserved.
29–31 TM[2:0]	Trace Mode. Any or all of the TM bits may set, enabling one or more traces. 000 No trace. 1XX Program trace enabled. X1X Data trace enabled (not supported mode) XX1 Ownership trace enabled.

[1] The output port mode control bit (OPC) and MCKO divide bits (MCK_DIV) are shown for clarity. These functions are controlled globally by the NPC port control register (PCR). These bits are writable in the PCR but have no effect.

Development control register 2 is shown in Figure 26-6 and its fields are described in Table 26-6.

Nexus Reg: 0x0003 — Access: User read/write

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

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

Figure 26-6. Development Control Register 2 (DC2)

Table 26-6. DC2 field descriptions

Field	Description
0–7 EWC[7:0]	EVTO Watchpoint Configuration. Any or all of the bits in EWC may be set to configure the EVTO watchpoint. 00000000 No Watchpoints trigger EVTO 1XXXXXXX Watchpoint #0 (IAC1 from Nexus1) triggers EVTO. X1XXXXXX Watchpoint #1 (IAC2 from Nexus1) triggers EVTO. XX1XXXXX Watchpoint #2 (IAC3 from Nexus1) triggers EVTO. XXX1XXXX Watchpoint #3 (IAC4 from Nexus1) triggers EVTO. XXXX1XXX Watchpoint #4 (DAC1 from Nexus1) triggers EVTO. XXXXX1XX Watchpoint #5 (DAC2 from Nexus1) triggers EVTO. XXXXXX1X Watchpoint #6 (DCNT1 from Nexus1) triggers EVTO. XXXXXXX1 Watchpoint #7 (DCNT2 from Nexus1) triggers EVTO.
8–31	Reserved.

NOTE

For the EWC bits to have any effect, the EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence.

26.6.2.4 Development Status Register (DS)

The development status register is used to report system debug status. When debug mode is entered or exited, or a core-defined low-power mode is entered, a debug status message is transmitted with DS[31:24]. The external tool can read this register at any time.

Nexus Reg: 0x0004 — Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBG	LPS			LPC		CHK	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 26-7. Development Status Register (DS)

Table 26-7. DS field descriptions

Field	Description
0 DBG	CPU Debug Mode Status. 0 CPU not in debug mode. 1 CPU in debug mode.
1–3 LPS	System Low-Power Status 000 Normal (run) mode xx1 Reserved, do not use x1x Nap mode 1xx Sleep mode
4–5 LPC[1:0]	CPU Low-Power Mode Status. 00 Normal (run) mode. 01 CPU in halted state. 10 CPU in stopped state. 11 Reserved.
6 CHK	CPU Checkstop Status. 0 CPU not in checkstop state. 1 CPU in checkstop state.
7–31	Reserved.

26.6.2.5 Read/Write Access Control/Status (RWCS)

The read/write access control/status register provides control for read/write access. Read/write access provides DMA-like access to memory-mapped resources on the system bus while the processor is halted or during runtime. Writes to this register do not begin until one JTAG clock (TCK) cycle after leaving the JTAG Update-DR state. The RWCS register also provides read/write access status information as shown in Table 26-9.

Nexus Reg: 0x0007 — Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AC	RW	SZ				MAP		PR		0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT														ERR	DV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-8. Read/Write Access Control/Status Register (RWCS)

Table 26-8. RWCS field description

Field	Description
0 AC	Access Control. 0 End access. 1 Start access.
1 RW	Read/Write Select. 0 Read access. 1 Write access.
2–4 SZ[2:0]	Word Size. 000 8-bit (byte.) 001 16-bit (halfword). 010 32-bit (word). 011 64-bit (doubleword—only in burst mode). 100–111 Reserved (default to word).
5–7 MAP[2:0]	MAP Select. 000 Primary memory map. 001–111 Reserved.
8–9 PR[1:0]	Read/Write Access Priority. 00 Lowest access priority. 01 Reserved (default to lowest priority). 10 Reserved (default to lowest priority). 11 Highest access priority.
10–15	Reserved.

Table 26-8. RWCS field description (continued)

Field	Description
16–31 CNT[13:0]	Access Control Count. Number of accesses of word size SZ.
30 ERR	Read/Write Access Error. See Table 26-9.
31 DV	Read/Write Access Data Valid. See Table 26-9.

Table 26-9 details the status bit encodings.

Table 26-9. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read access has not completed	Write access completed without error	0	0
Read access error has occurred	Write access error has occurred	1	0
Read access completed without error	Write access has not completed	0	1
Not allowed	Not allowed	1	1

26.6.2.6 Read/Write Access Address (RWA)

The read/write access address register provides the system bus address to be accessed when initiating a read or a write access.

Nexus Reg: 0x0009 — Access: User read/write

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

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

Figure 26-9. Read/Write Access Address Register (RWA)

26.6.2.7 Read/Write Access Data (RWD)

The read/write access data register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Nexus Reg: 0x000A — Access: User read/write

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

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

Figure 26-10. Read/Write Access Data Register (RWD)

26.6.2.8 Watchpoint Trigger Register (WT)

The watchpoint trigger register allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control program and/or data trace enable and disable. The WT bits can be used to produce an address-related window for triggering trace messages.

Nexus Reg: 0x000B — Access: User read/write

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

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

Figure 26-11. Watchpoint Trigger Register (WT)

Table 26-10 details the watchpoint trigger register fields.

Table 26-10. WT field descriptions

Field	Description
0–2 PTS[2:0]	Program Trace Start Control. 000 Trigger disabled. 001 Use watchpoint #0 (IAC1 from Nexus1). 010 Use watchpoint #1 (IAC2 from Nexus1). 011 Use watchpoint #2 (IAC3 from Nexus1). 100 Use watchpoint #3 (IAC4 from Nexus1). 101 Use watchpoint #4 (DAC1 from Nexus1). 110 Use watchpoint #5 (DAC2 from Nexus1). 111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1).
3–5 PTE[2:0]	Program Trace End Control. 000 Trigger disabled. 001 Use watchpoint #0 (IAC1 from Nexus1). 010 Use watchpoint #1 (IAC2 from Nexus1). 011 Use watchpoint #2 (IAC3 from Nexus1). 100 Use watchpoint #3 (IAC4 from Nexus1). 101 Use watchpoint #4 (DAC1 from Nexus1). 110 Use watchpoint #5 (DAC2 from Nexus1). 111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1).
12–31	Reserved.

26.7 Functional description

The NDI block is implemented by integrating the following blocks on this device:

- Nexus e200z0 development interface (OnCE and Nexus2p subblocks)
- Nexus port controller (NPC) Block
- NPC_HNDSHK module

26.7.1 NPC_HNDSHK module

This module enables debug entry/exit across low-power modes (Stop, Halt, Standby).

The NPC_HNDSHK supports:

- Setting and clearing of the NPC PCR sync bit on low-power mode entry and exit
- Putting the core into debug mode on low-power mode exit
- Generating a falling edge on the JTAG TDO pad on low-power mode exit

On Halt0, Stop0, or Standby0 mode entry, the MC_ME asserts the lp_mode_entry_req input after the clock disable process has completed and before the processor enters its halted or stopped state. The mode transition will then not proceed until the lp_mode_entry_ack output has been asserted. The notification to the debugger of a low-power mode entry consists of setting the low-power mode handshake bit in the port control register (read by the debugger) via the lp_sync_in output. The debugger acknowledges that the transition into a low-power mode may proceed by clearing the low-power mode handshake bit in the port control register (written by the debugger), which results in the deassertion of the lp_sync_out input.

In anticipation of the low-power mode exit notification, the TDO pad is driven to 1.

On Halt0 or Stop0 mode exit, the MC_ME asserts the lp_mode_exit_req input after ensuring that the regulator and memories are in normal mode and before the processor exits its halted or stopped state. The mode transition will then not proceed until the lp_mode_exit_ack output has been asserted. The MC_RGM asserts the exit_from_standby input when executing a reset sequence due to a Standby0 exit. The reset sequence will then not complete until the lp_mode_exit_ack output has been asserted.

The notification to the debugger of a low-power mode exit consists of driving the TDO pad to 0. The debugger acknowledges that the transition from a low-power mode can continue by setting the low-power mode sync bit in the port control register (written by debugger), which results in the assertion of the lp_sync_out input.

NOTE

The debugger clock multiplexer may not guarantee glitch-free switching. Therefore, TCK should be disabled from when the debugger clears the sync bit in ENTRY_CLR until the debugger senses the falling edge of TDO in TDO_SET.

26.7.2 Enabling Nexus clients for TAP access

After the conditions have been met to bring the NDI out of the reset state, the loading of a specific instruction in the JTAG controller (JTAGC) block is required to grant the NDI ownership of the TAP. Each Nexus client has its own JTAGC instruction opcode for ownership of the TAP, granting that client the means to read/write its registers. The JTAGC instruction opcode for each Nexus client is shown in Table 26-11. After the JTAGC opcode for a client has been loaded, the client is enabled by loading its NEXUS-ENABLE instruction. The NEXUS-ENABLE instruction opcode for each Nexus client is listed in Table 26-12. Opcodes for all other instructions supported by Nexus clients can be found in the relevant sections of this chapter.

Table 26-11. JTAGC Instruction opcodes to enable Nexus clients

JTAGC instruction	Opcode	Description
ACCESS_AUX_TAP_NPC	10000	Enables access to the NPC TAP controller.
ACCESS_AUX_TAP_ONCE	10001	Enables access to the e200z0 TAP controller.

Table 26-12. Nexus client JTAG instructions

Instruction	Description	Opcode
NPC JTAG instruction opcodes		
NEXUS_ENABLE	Opcode for NPC Nexus ENABLE instruction (4-bits)	0x0
BYPASS	Opcode for the NPC BYPASS instruction (4-bits)	0xF
e200z0 OnCE JTAG instruction opcodes[1]		
NEXUS2_ACCESS	Opcode for e200z0 OnCE Nexus ENABLE instruction (10-bits)	0x7C
BYPASS	Opcode for the e200z0 OnCE BYPASS instruction (10-bits)	0x7F

1 Refer to the e200z0 reference manual for a complete list of available OnCE instructions.

26.7.3 Configuring the NDI for Nexus messaging

The NDI is placed in disabled mode upon exit of reset. If message transmission via the auxiliary port is desired, a write to the port configuration register (PCR) located in the NPC is then required to enable the NDI and select the mode of operation. Asserting MCKO_EN in the PCR places the NDI in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects Full-Port or Reduced-Port mode, respectively. When writing to the PCR, the PCR LSB must be written to a logic zero. Setting the LSB of the PCR enables factory debug mode and prevents the transmission of Nexus messages.

Table 26-13 describes the NDI configuration options.

Table 26-13. NDI configuration options

JCOMP Asserted	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	Configuration
No	X	X	Reset
Yes	0	X	Disabled
Yes	1	1	Full-Port mode
Yes	1	0	Reduced-Port mode

26.7.4 Programmable MCKO frequency

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

Figure 26-14 shows the MCKO_DIV encodings. In this table, SYS_CLK represents the system clock frequency. The default value selected if a reserved encoding is programmed is SYS_CLK÷2.

NOTE

On MPC5606S, the pad type used for the Nexus2+ signals will not support the default SYSCLK÷2 and SYSCLK÷1 setting, so the user must change the MCKO frequency to be not faster than SYSCLK÷4.

Table 26-14. MCKO_DIV Values

MCKO_DIV[2:0]	MCKO frequency
0b000	SYSCLK÷1
0b001	SYSCLK÷2
0b010	Reserved
0b011	SYS_CLK÷4
0b100	Reserved
0b101	Reserved

Table 26-14. MCKO_DIV Values (continued)

MCKO_DIV[2:0]	MCKO frequency
0b110	Reserved
0b111	SYS_CLK÷8

26.7.5 Nexus messaging

Most of the messages transmitted by the NDI include an SRC field. This field is used to identify which source generated the message. Table 26-15 shows the values used for the SRC field by the different clients on the MPC5606S. These values are specific to the MPC5606S. The size of the SRC field in transmitted messages is 4 bits. This value is also specific to the MPC5606S.

Table 26-15. SRC packet encodings

SRC[3:0]	MPC5606S client
0b0000	e200z0
All other combinations	Reserved

26.7.6 EVTO sharing

The NPC block controls sharing of the EVTO output between all Nexus clients that generate an EVTO signal. The sharing mechanism is a logical AND of all incoming EVTO signals from Nexus blocks, thereby asserting EVTO whenever any block drives its EVTO. When there is no active MCKO, such as in disabled mode, the NPC drives EVTO for two system clock periods. EVTO sharing is active as long as the NDI is not in reset.

26.7.7 Debug mode control

On MPC5606S, program breaks can be requested either by using the EVTI pin as a break request, or when a Nexus event is triggered.

26.7.7.1 EVTI generated break request

To use the EVTI pin as a debug request, the EIC field in the e200z0 Nexus2+ Development Control Register 1 (DC1[4:3]) must be set to configure the EVTI input as a debug request.

26.7.8 Nexus reset control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus blocks. The single bit reset signal functions much like the IEEE 1149.1-2001 defined $\overline{\text{TRST}}$ signal but has a default value of disabled (JCOMP is pulled low during reset). The IEEE 1149.1-2001 defines $\overline{\text{TRST}}$ to be pulled up (enabled) by default.

Chapter 27
Periodic Interrupt Timer (PIT)

27.1 Introduction

27.1.1 Overview

The PIT is an array of timers that can be used to raise interrupts and trigger DMA channels.

This device has one PIT module with four Timer Channels (PIT channels 0 through 3). These are connected to the Trigger input 0 through 3 of the DMA MUX.

Figure 27-1 shows the PIT block diagram.

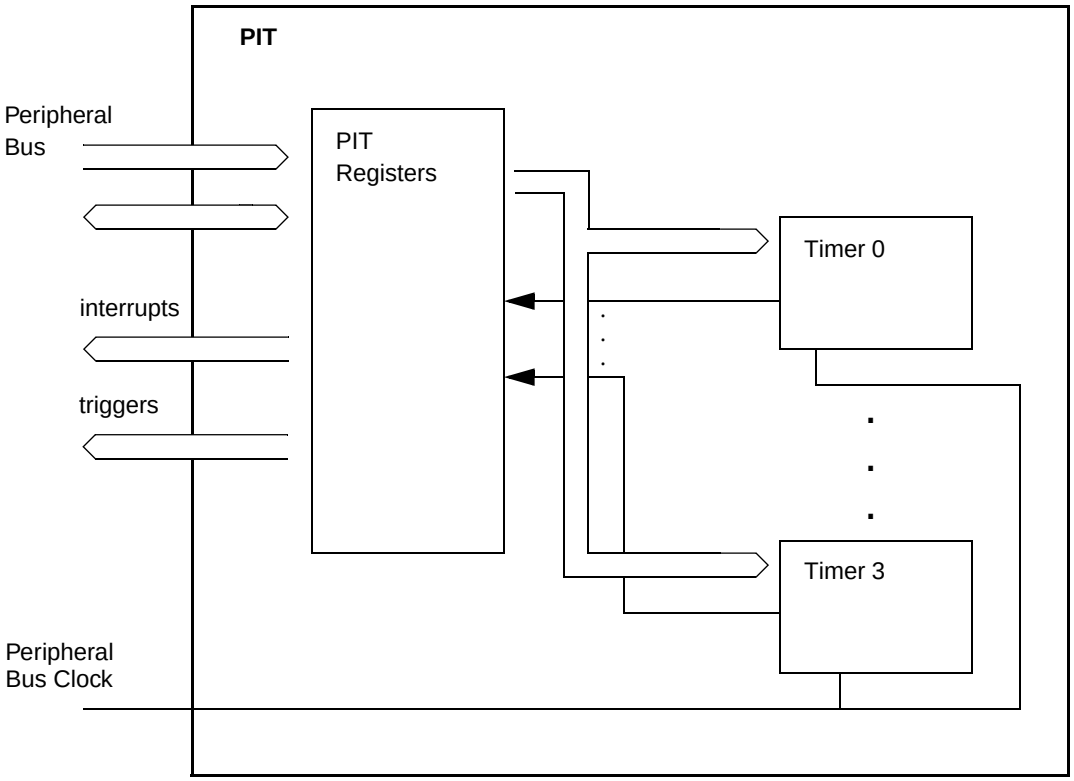


Figure 27-1. PIT block diagram

27.1.2 Features

The main features of this block are:

- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable

- Independent timeout periods for each timer

27.2 Signal description

The PIT module has no external pins.

27.3 Memory map and register description

This section provides a detailed description of all registers accessible in the PIT module.

27.3.1 Memory map

Table 27-1 gives an overview on all PIT registers.

Table 27-1. PIT memory map

Address offset	Use	Access
0x000	PIT Module Control Register	R/W
0x004–0x0FC	Reserved	R
0x100–0x10C	Timer Channel 0	[1]
0x110–0x11C	Timer Channel 1	[1]
0x120–0x12C	Timer Channel 2	[1]
0x130–0x13C	Timer Channel 3	[1]
0x140–0x1FC	Reserved	R

[1] See Table 27-2

Table 27-2. Timer Channel n

Address offset	Use	Access
Channel + 0x00	Timer Load Value Register	R/W
Channel + 0x04	Current Timer Value Register	R
Channel + 0x08	Timer Control Register	R/W
Channel + 0x0C	Timer Flag Register	R/W

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

NOTE

Reserved registers will read as 0, writes will have no effect.

27.3.2 Register descriptions

This section describes in address order all the PIT registers and their individual bits.

27.3.2.1 PIT Module Control Register (PITMCR)

The PIT Module Control Register (PITMCR) controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Offset: Base + 0x000 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MDIS	FRZ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 27-2. PIT Module Control Register (PITMCR)

Table 27-3. PITMCR field descriptions

Field	Description
MDIS	Module Disable. This is used to disable the module clock. This bit should be enabled before any other setup is done. 0 Clock for PIT Timers is enabled 1 Clock for PIT Timers is disabled (default)
FRZ	Freeze. Allows the timers to be stopped when the device enters debug mode. 0 Timers continue to run in debug mode. 1 Timers are stopped in debug mode.

27.3.2.2 Timer Load Value (LDVAL) register

The Timer Load Value (LDVAL) register selects the timeout period for the timer interrupts.

Offset: Channel_base + 0x00 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSV 15	TSV 14	TSV 13	TSV 12	TSV 11	TSV 10	TSV9	TSV8	TSV7	TSV6	TSV5	TSV4	TSV3	TSV2	TSV1	TSV0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-3. Timer Load Value (LDVAL) register

Table 27-4. LDVAL field descriptions

Field	Description
TSV*n*	Time Start Value Bits. These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again (see Figure 27-8).

27.3.2.3 Current Timer Value (CVAL) register

The Current Timer Value (CVAL) register indicates the current timer position.

Offset: Channel_base + 0x04 Access: Read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL 15	TVL 14	TVL 13	TVL 12	TVL 11	TVL 10	TVL9	TVL8	TVL7	TVL6	TVL5	TVL4	TVL3	TVL2	TVL1	TVL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-4. Current Timer Value (CVAL) register

Table 27-5. CVAL field descriptions

Field	Description
TVL*n*	Current Timer Value. These bits represent the current timer value. Note that the timer uses a downcounter. NOTE: The timer values will be frozen in Debug mode if the FRZ bit is set in the PIT Module Control Register (see Figure 27-2)

27.3.2.4 Timer Control (TCTRL) register

The Timer Control (TCTRL) register contains the control bits for each timer.

Offset: Channel_base + 0x08 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-5. Timer Control (TCTRL) register

Table 27-6. TCTRL field descriptions

Field	Description
TIE	Timer Interrupt Enable Bit. 0 Interrupt requests from Timer x are disabled 1 Interrupt will be requested whenever TIF is set When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
TEN	Timer Enable Bit. 0 Timer will be disabled 1 Timer will be active

27.3.2.5 Timer Flag (TFLG) register

The Timer Flag (TFLG) register holds the PIT interrupt flags.

Offset: Channel_base + 0x0C Access: Read/Write

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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-6. Timer Flag (TFLG) register

Table 27-7. TFLG field descriptions

Field	Description
TIF	Time Interrupt Flag. TIF is set to 1 at the end of the timer period.This flag can be cleared only by writing it with a 1. Writing a 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred 1 Timeout has occurred

27.4 Functional description

27.4.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts, each interrupt will be available on a separate interrupt line.

27.4.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse, and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see Figure 27-7).

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see Figure 27-8).

It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (see Figure 27-9).

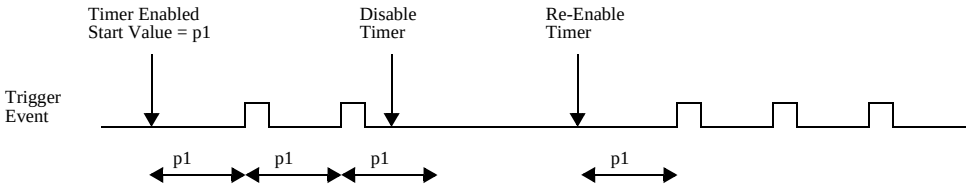


Figure 27-7. Stopping and Starting a Timer

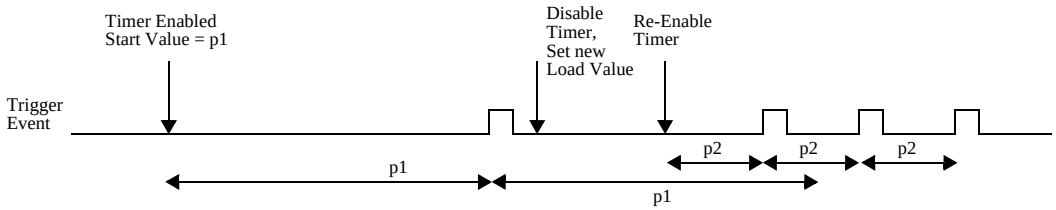


Figure 27-8. Modifying Running Timer Period

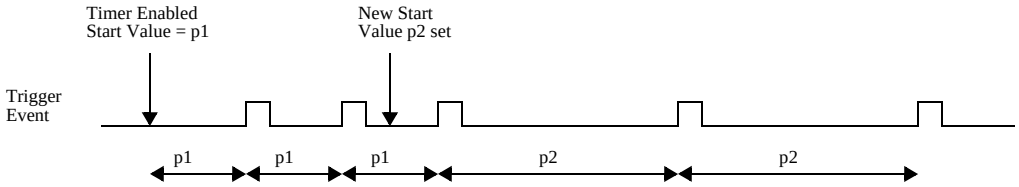


Figure 27-9. Dynamically Setting a New Load Value

27.4.1.2 Debug mode

In Debug mode the timers will be frozen. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (for example, the timer values) and then continue the operation.

27.4.2 Interrupts

All of the timers support interrupt generation. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be disabled by setting the TIE bits to zero. The timer interrupt flags (TIF) are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to that TIF bit.

27.5 Initialization and application information

27.5.1 Example configuration

In the example configuration:

- the PIT clock has a frequency of 50 MHz
- timer 1 shall create an interrupt every 5.12 ms
- timer 3 shall create a trigger event every 30 ms

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the PITCTRL register.

The 50 MHz clock frequency equates to a clock period of 20 ns. Timer 1 needs to trigger every 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles. The value for the LDVAL register trigger would be calculated as (period / clock period) – 1.

This means that LDVAL1 with 0003E7FF hex and LDVAL3 with 0016E35F hex.

The interrupt for Timer 1 is enabled by setting TIE in the TCTRL1 register. The timer is started by writing a 1 to bit TEN in the TCTRL1 register.

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a 1 to bit TEN in the TCTRL3 register, bit TIE stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_CTRL = 0x00;

// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN; // start RTI

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3for 1500000 cycles
PIT_TCTRL3 = TEN; // start timer 3
```

Chapter 28
Peripheral Bridge (PBRIDGE)

28.1 Introduction

The PBRIDGE is the interface between the system bus and on-chip peripherals. It differs from that used on Power Architecture products in the fact that it has a hard-wired configuration and cannot be configured in software.

28.1.1 Overview

MPC5606S devices have one PBRIDGE, which provides an interface between the system bus and all lower bandwidth peripherals. Accesses that fall within the address space of the PBRIDGE are decoded to provide individual module selects for peripheral devices on the slave bus interface.

28.1.2 Features

The following list summarizes the key features of the PBRIDGE.

- Supports the slave interface signals. This interface is only meant for slave peripherals.
- Supports 32-bit slave peripherals. (Byte, halfword, and word reads and writes are supported to each.)

28.1.3 Modes of operation

The PBRIDGE has only one operating mode.

28.2 Functional description

The PBRIDGE serves as an interface between a system bus and the peripheral (slave) bus. It functions as a protocol translator. Accesses that fall within the address space of the PBRIDGE are decoded to provide individual module selects for peripheral devices on the slave bus interface.

28.2.1 Access support

Aligned 32-bit word accesses, halfword accesses, and byte accesses are supported for the peripherals. Peripheral registers must not be misaligned, although no explicit checking is performed by the PBRIDGE.

NOTE

Data accesses that cross a 32-bit boundary are not supported.

28.2.1.1 Peripheral write buffering

Buffered writes are not supported by the MPC5606S PBRIDGE.

28.2.1.2 Read Cycles

Two-clock read accesses are possible with the PBRIDGE when the requested access size is 32-bits or smaller, and is not misaligned across a 32-bit boundary.

28.2.1.3 Write Cycles

Three-clock write accesses are possible with the PBRIDGE when the requested access size is 32-bits or smaller. Misaligned writes that cross a 32-bit boundary are not supported.

28.2.2 General Operation

Slave peripherals are modules that contain readable/writable control and status registers. The system bus master reads and writes these registers through the PBRIDGE. The PBRIDGE generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the slave peripherals. The PBRIDGE captures read data from the slave interface and drives it on the system bus.

The PBRIDGE occupies a 64 MB portion of the address space. The register maps of the slave peripherals are located on 16-KB boundaries. Each slave peripheral is allocated one 16-KB block of the memory map, and is activated by one of the module enables from the PBRIDGE.

The PBRIDGE is responsible for indicating to slave peripherals if an access is in supervisor or user mode.

Chapter 29
Power Control Unit (MC_PCU)

29.1 Introduction

29.1.1 Overview

The power control unit (MC_PCU) is used to reduce the overall SoC power consumption. Power can be saved by disconnecting parts of the SoC from the power supply via a power switching device. The SoC is grouped into multiple parts having this capability, which are called power domains.

When a power domain is disconnected from the supply, the power consumption is reduced to zero in that domain. Any status information of such a power domain is lost. When re-connecting a power domain to the supply voltage, the domain draws an increased current until the power domain reaches its operational voltage.

Power domains are controlled on a device mode basis. For each mode, software can configure whether a power domain is connected to the supply voltage (startup state) or disconnected (power-down state). Maximum power saving is reached by entering the Standby mode.

On each mode change request, the MC_PCU evaluates the power domain settings in the power domain configuration registers and initiates a power-down or a startup sequence for each individual power domain. The startup/shutdown sequences are handled by finite state machines to ensure a smooth and safe transition from one power state to the other.

Exiting the Standby mode can only be done via a system wakeup event as all power domains other than power domain #0 are in the power-down state.

In addition, the MC_PCU acts as a bridge for mapping the VREG peripheral to the MC_PCU address space.

Figure 29-1 depicts the MC_PCU block diagram.

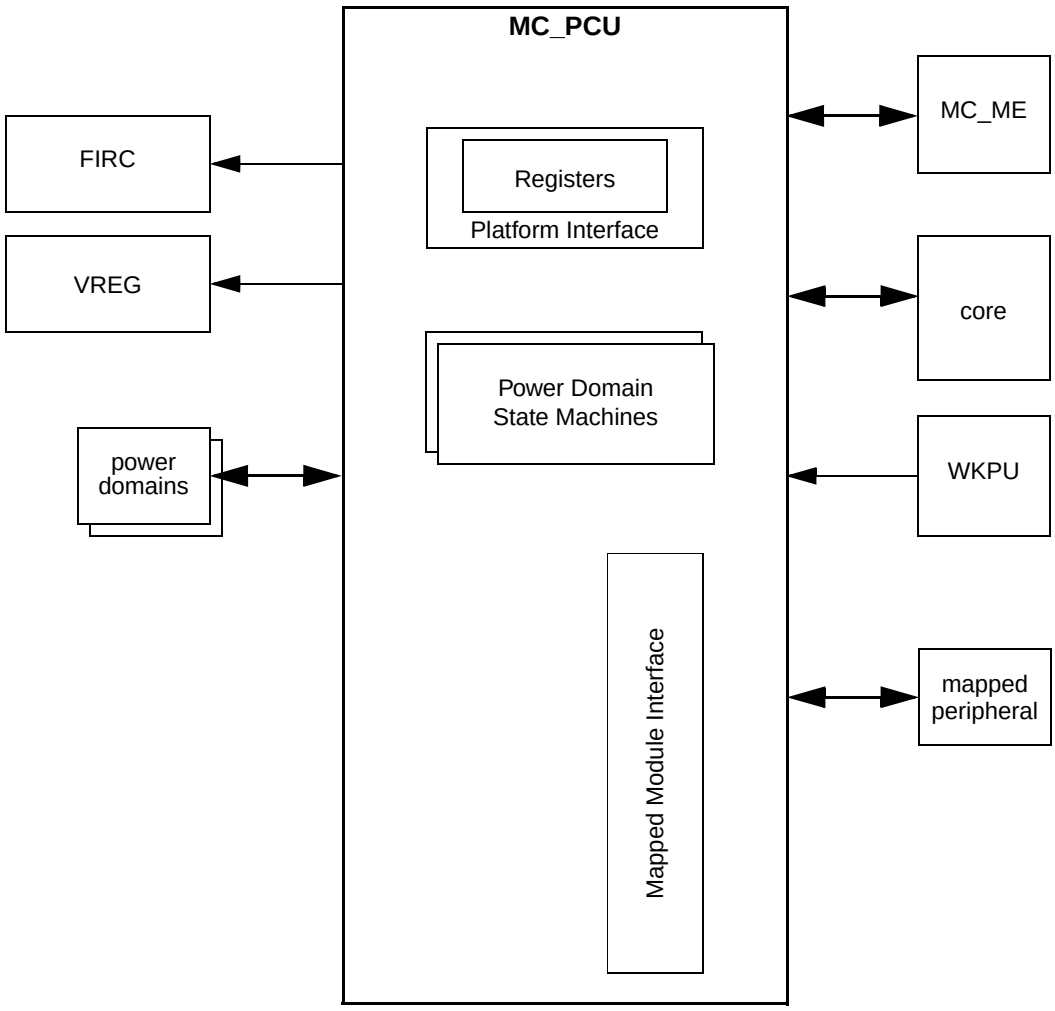


Figure 29-1. MC_PCU block diagram

29.1.2 Features

The MC_PCU includes the following features:

- Support for 3 power domains
- Support for device modes Reset, DRUN, Safe, Test, Run0…3, Halt, Stop, and Standby (for further mode details, please see Chapter 25, Mode Entry Module (MC_ME))
- Power states updating on each mode change and on system wakeup
- A handshake mechanism for power state changes thus guaranteeing operable voltage
- Maps the VREG registers to the MC_PCU address space

29.1.3 Modes of operation

The MC_PCU is available in all device modes.

29.2 External signal description

The MC_PCU has no connections to any external pins.

29.3 Memory map and register definition

29.3.1 Memory map

Table 29-1. MC_PCU register description

Address	Name	Description	Size	Access
0xC3FE_8000	PCU_PCONF0	Power Domain #0 Configuration	word	read
0xC3FE_8004	PCU_PCONF1	Power Domain #1 Configuration	word	read
0xC3FE_8008	PCU_PCONF2	Power Domain #2 Configuration	word	read/write
0xC3FE_8040	PCU_PSTAT	Power Domain Status Register	word	read

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- Not change register content
- Cause a transfer error

Table 29-2. MC_PCU memory map

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FE_8000	PCU_PCONF0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
		W																
0xC3FE_8004	PCU_PCONF1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
		W																
0xC3FE_8008	PCU_PCONF2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
		W																

Table 29-2. MC_PCU memory map (continued)

Address	Name		0 16	1 17	2 18	3 19	27 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
0xC3FE_800C … 0xC3FE_803C	Reserved																	
0xC3FE_8040	PCU_PSTAT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R														PD2	PD1	PD0
		W																
0x044 … 0x07C	Reserved																	
0xC3FE_8080 … 0xC3FE_80FC	VREG registers																	
0xC3FE_8100 … 0xC3FE_BFFC	Reserved																	

29.3.2 Register descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at address 0xC3FE_8040, as a half-word at address 0xC3FE_8042, or as a byte at address 0xC3FE_8043.

29.3.2.1 Power Domain #0 Configuration Register (PCU_PCONF0)

Address 0xC3FE_8000 Access: Supervisor read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 29-2. Power Domain #0 Configuration Register (PCU_PCONF0)

This register defines for power domain #0 whether it is on or off in each device mode. As power domain #0 is the always-on power domain (and includes the MC_PCU), none of its bits are programmable. This register is available for completeness reasons.

Table 29-3. Power Domain Configuration Register field descriptions

Field	Description
RST	Power domain control during Reset mode 0 Power domain off 1 Power domain on
TEST	Power domain control during Test mode 0 Power domain off 1 Power domain on
SAFE	Power domain control during Safe mode 0 Power domain off 1 Power domain on
DRUN	Power domain control during DRUN mode 0 Power domain off 1 Power domain on
RUN0	Power domain control during Run0 mode 0 Power domain off 1 Power domain on
RUN1	Power domain control during Run1 mode 0 Power domain off 1 Power domain on
RUN2	Power domain control during Run2 mode 0 Power domain off 1 Power domain on
RUN3	Power domain control during Run3 mode 0 Power domain off 1 Power domain on
HALT	Power domain control during Halt mode 0 Power domain off 1 Power domain on
STOP	Power domain control during Stop mode 0 Power domain off 1 Power domain on
STBY0	Power domain control during Standby mode 0 Power domain off 1 Power domain on

29.3.2.2 Power Domain #1 Configuration Register (PCU_PCONF1)

Address 0xC3FE_8004 Access: Supervisor read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 29-3. Power Domain #1 Configuration Register (PCU_PCONF1)

This register defines for power domain #1 whether it is on or off in each device mode. The bit field description is the same as in Table 29-3. As the platform, clock generation, and mode control reside in power domain #1, this power domain is only powered down during the Standby mode. Therefore, none of the bits is programmable. This register is available for completeness reasons.

The difference between PCU_PCONF0 and PCU_PCONF1 is the reset value of the STBY0 bit: During the Standby mode, power domain #1 is disconnected from the power supply, and therefore PCU_PCONF1.STBY0 is always 0. Power domain #0 is always on, and therefore PCU_PCONF0.STBY0 is 1.

For further details about Standby mode, please refer to Section 29.4.4.2, Standby mode transition.

29.3.2.3 Power Domain #2 Configuration Register (PCU_PCONF2)

Address 0xC3FE_8008 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 29-4. Power Domain #2 Configuration Register (PCU_PCONF2)

This register defines for power domain #2 whether it is on or off in each device mode. The bit field description is the same as in Table 29-3.

29.3.2.4 Power Domain Status Register (PCU_PSTAT)

Address 0xC3FE_8040 Access: Supervisor read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R														PD2	PD1	PD0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 29-5. Power Domain Status Register (PCU_PSTAT)

This register reflects the power status of all available power domains.

Table 29-4. Power Domain Status Register (PCU_PSTAT) field descriptions

Field	Description
PD*n*	Power status for power domain #*n* 0 Power domain is inoperable 1 Power domain is operable

29.4 Functional description

29.4.1 General

The MC_PCU controls all available power domains on a device mode basis. The PCU_PCONF*n* registers specify during which system/user modes a power domain is powered up. The power state for each individual power domain is reflected by the bits in the PCU_PSTAT register.

On a mode change, the MC_PCU evaluates which power domain(s) must change power state. The power state is controlled by a state machine (FSM) for each individual power domain (see Figure 3-1) which ensures a clean and safe state transition.

29.4.2 Reset / Power-On Reset

After any reset, the SoC will transition to the Reset mode during which all power domains are powered up (see the MC_ME chapter). Once the reset sequence has been completed, the DRUN mode is entered and software can begin the MC_PCU configuration.

29.4.3 MC_PCU Configuration

Per default, all power domains are powered in all modes other than Standby. Software can change the configuration for each power domain on a mode basis by programming the PCU_PCONF*n* registers.

Each power domain which is powered down is held in a reset state. Read/write accesses to peripherals in those power domains will result in a transfer error.

29.4.4 Mode transitions

On a mode change requested by the MC_ME, the MC_PCU evaluates the power configurations for all power domains. It compares the settings in the PCU_PCONF*n* registers for the new mode with the settings for the current mode. If the configuration for a power domain differs between the modes, a power state change request is generated. These requests are handled by a finite state machine to ensure a smooth and safe transition from one power state to another.

29.4.4.1 DRUN, Safe, Test, Run0…3, Halt, and Stop mode transition

The DRUN, Safe, Test, Run0…3, Halt, and Stop modes allow an increased power saving. The level of power saving is software-controllable via the settings in the PCU_PCONF*n* registers for power domain #2 onwards. The settings for power domains #0 and #1 cannot be changed. Therefore, power domains #0 and #1 remain connected to the power supply for all modes beside Standby.

Figure 29-6 shows an example for a mode transition from Run0 to Halt and back, which will result in power domain #2 being powered down during the Halt mode. In this case, PCU_PCONF2.HALT is programmed to be 0.

When the MC_PCU receives the mode change request to Halt mode, it starts its power-down phase. During the power-down phase, clocks are disabled and the reset is asserted resulting in a loss of all information for this power domain.

Then the power domain is disconnected from the power supply (power-down state).

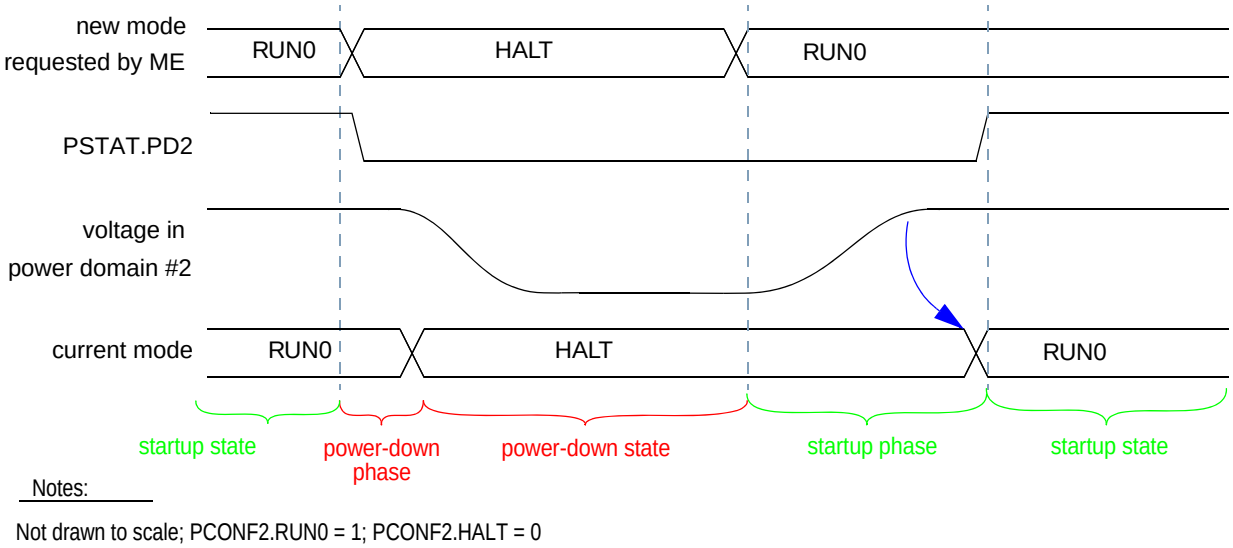


Figure 29-6. MC_PCU Events During Power Sequences (non-Standby mode)

When the MC_PCU receives a mode change request to Run0, it starts its startup phase if PCU_PCONF2.RUN0 is 1. The power domain is re-connected to the power supply, and the voltage in

power domain #2 will increase slowly. Once the voltage of power domain #2 is within an operable range, its clocks are enabled, and its resets are deasserted (startup state).

NOTE

It is possible that, due to a mode change, startup is requested before a power domain completed its power-down sequence. In this case, the information in that power domain is lost.

29.4.4.2 Standby mode transition

Standby mode offers the maximum power saving. The level of power saving is software-controllable via the settings in the PCU_PCONF*n* registers for power domain #2 onwards. Power domain #0 stays connected to the power supply while power domain #1 is disconnected from the power supply. Amongst others power domain #1 contains the platform and the MC_ME. Therefore this mode differs from all other user/system modes.

Once Standby is entered it can only be left via a system wakeup. On exiting the Standby mode, all power domains are powered up according to the settings in the PCU_PCONF*n* registers, and the DRUN mode is entered. In DRUN mode, at least power domains #0 and #1 are powered.

Figure 29-7 shows an example for a mode transition from Run0 to Standby to DRUN. All power domains which have PCU_PCONF*n*.STBY0 cleared will enter power-down phase. In this example only power domain #1 will be disabled during Standby mode.

When the MC_PCU receives the mode change request to Standby mode it starts the power down phase for power domain #1. During the power down phase, clocks are disabled and reset is asserted resulting in a loss of all information for this power domain. Then the power domain is disconnected from the power supply (power-down state).

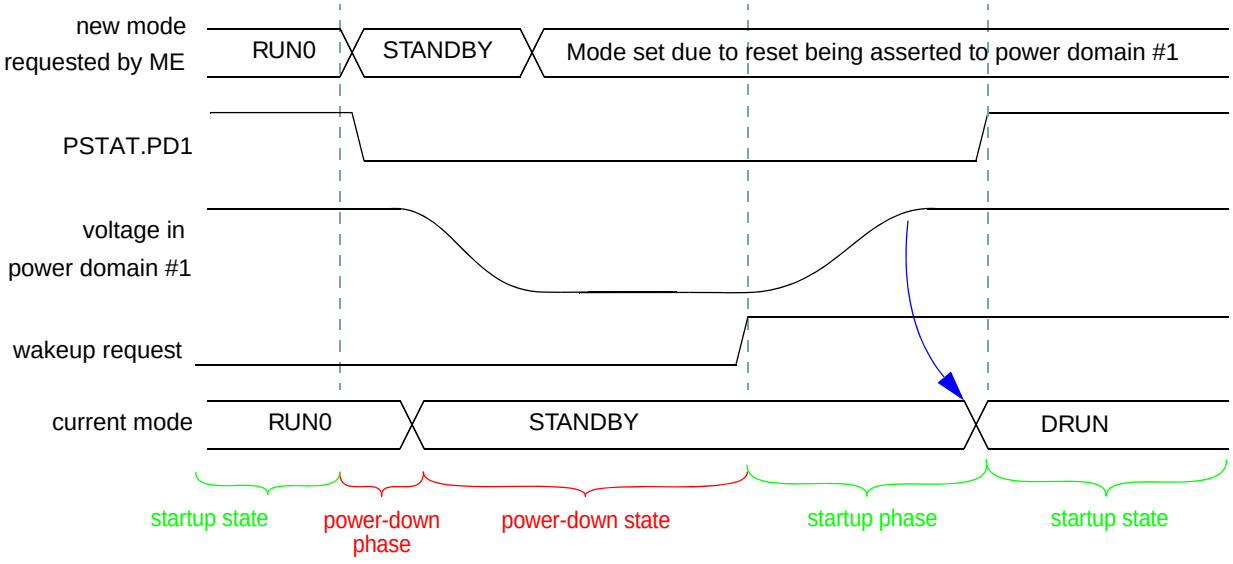


Figure 29-7. MC_PCU Events During Power Sequences (Standby mode)

When the MC_PCU receives a system wakeup request, it starts the startup phase. The power domain is re-connected to the power supply and the voltage in power domain #1 will increase slowly. Once the voltage is in an operable range, clocks are enabled and the reset is be deasserted (startup state).

NOTE

It is possible that due to a wakeup request, startup is requested before a power domain completed its power-down sequence. In this case, the information in that power domain is lost.

29.4.4.3 Power saving for memories during Standby mode

All memories which are not powered down during Standby mode automatically enter a power saving state. No software configuration is required to enable this power saving state. While a memory is residing in this state an increased power saving is achieved. Data in the memories is retained.

29.5 Initialization information

To initialize the MC_PCU, the registers PCU_PCONF2… should be programmed. After programming is done, those registers should no longer be changed.

29.6 Application information

29.6.1 Standby mode considerations

Standby offers maximum power saving possibility. But power is only saved during the time a power domain is disconnected from the supply. Increased power is required when a power domain is re-connected to the power supply. Additional power is required during restoring the information (e.g. in the platform). Care should be taken that the time during which the SoC is operating in Standby mode is significantly longer than the required time for restoring the information.

Chapter 30
Quad Serial Peripheral Interface (QuadSPI)

30.1 Preface

30.1.1 Conventions

Table 30-1 contains conventions used in this document.

Table 30-1. Conventions

Terms	Description
ACTIVE_HIGH	Names for signals that are active high are shown in uppercase text without an overbar. Signals that are active high are referred to as asserted when they are logic 1 and negated when they are logic 0.
ACTIVE_LOW	A bar over a signal name indicates that the signal is active low. Active-low signals are referred to as asserted when they are logic 0 and negated when they are logic 1.
0x0F	Hexadecimal numbers
0b0011	Binary numbers
x	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded 0bx001, the state of the first bit is a don't care.
REG[BIT]	Denotes the bitfields BIT in the register REG

30.1.2 Acronyms and Abbreviations

Table 30-2 contains acronyms and abbreviations used in this document.

Table 30-2. Acronyms and Abbreviations

Terms	Description
AHB	Advanced High-performance Bus, version of AMBA
AMBA	Advanced Microcontroller Bus Architecture
CS	Chip Select.
DMA	Direct Memory Access.
EOQ	End of Queue
LSB	Least Significant Bit
MSB	Most Significant Bit
PCS	Peripheral Chip Select
QSPI, QuadSPI	Quad Serial Peripheral Interface
SCK	Serial Communications Clock
SPI	Serial Peripheral Interface

Table 30-2. Acronyms and Abbreviations

Terms	Description
SS	Slave Select. Signal from the SPI master to the SPI slave indicating which SPI slave device the Master want to communicate with.
w1c	Write 1 to clear, writing a 1 to this field resets the flag

30.1.3 Glossary for QuadSPI module

Table 30-3. Glossary

Term	Definition
AHB Command	An AHB Command is a SFM Command triggered by a read access to the address range belonging to the memory-mapped access defined in Table 30-35. Refer to Section 30.6.6.2, AHB bus related commands, for details.
Asserted	A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.
Baud Rate	Rate of data transmission in bits per second.
Clear	To clear a bit or bits means to establish logic level zero on the bit or bits.
Clock Phase	Determines when the data should be sampled relative to the active edge of SCK
Clock Polarity	Determines the idle state of the SCK signal.
Deserialize	To convert data from a serial format to a parallel format.
Drain	To remove entries from a FIFO by software or hardware.
Field	Two or more register bits grouped together.
FIFO entry	FIFO entries and FIFO registers are used interchangeably.
Fill	To add entries to a FIFO by software or hardware.
Frame	The data content of a serial transmission. Also referred to as QuadSPI Data.
Host	Refers to another functional block in the device containing the QuadSPI module
Instruction Code	8 bits defining the type of command to be executed.
IP Command	A IP Command is a SFM Command triggered by writing into the QSPI_MCR[IC] field.
Logic level one	The voltage that corresponds to Boolean true (1) state.
Logic level zero	The voltage that corresponds to Boolean false (0) state.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
RX FIFO	First-In-First-Out buffer for received data
Serialize	To convert data from a parallel format to a serial format.
Set	To set a bit or bits means to establish logic level one on the bit or bits.

Table 30-3. Glossary (continued)

Term	Definition
SFM Command	Applicable in SFM mode only: A SFM command consists of an instruction code and all other parameters (e.g. size or mode bytes) needed for that specific instruction code. Triggering a command either initiates a transaction on the external serial flash or results in an error. Refer to Table 30-50 for details.
SFM mode	The QuadSPI is set up to for an external serial flash device.
SPI Command	Applicable in SPI Master mode only. A SPI Command is part of each TX FIFO entry specifying the parameters for the transmission of that specific entry.
SPI Master mode	The QuadSPI is set up as SPI master to communicate with external SPI slave devices.
SPI modes	SPI Slave mode or SPI Master mode.
SPI Slave mode	The QuadSPI is set up as SPI slave to communicate with an external SPI master device.
Transaction	A transaction consists of all flags, data and signals in either direction to execute a command for an attached serial flash device. It is a combination of chip select, sclk, instruction code, address, mode- and/or dummy bytes, transmit and/or receive data.
Transfer Format	The combination of SCK polarity, SCK phase, data MSB/LSB first, and associated CS signal timing during a serial transmission
TX FIFO	First-In-First-Out buffer for transmit data

30.2 Introduction

Figure 30-1 is a block diagram of the Quad Serial Peripheral Interface (QuadSPI) module.

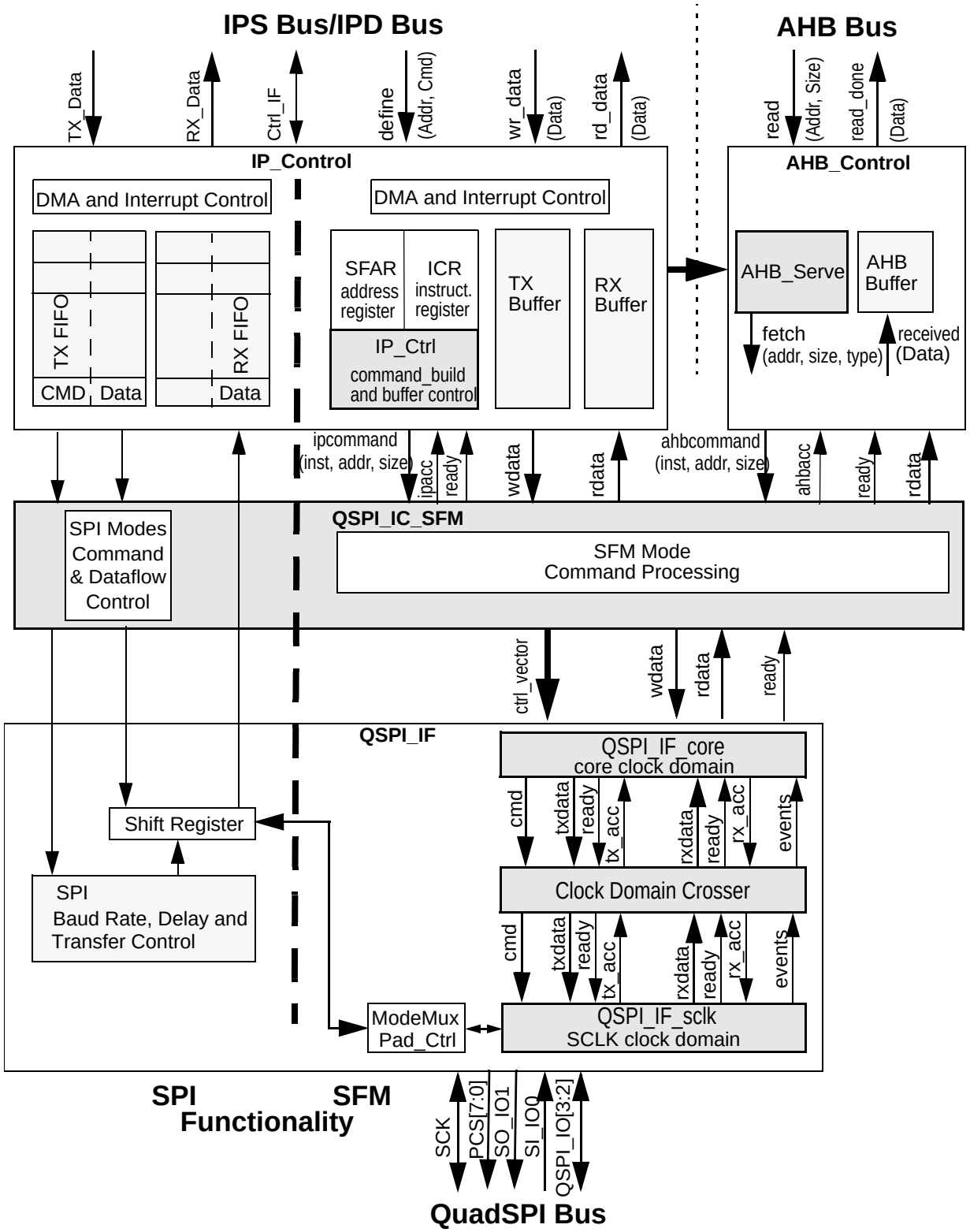


Figure 30-1. QuadSPI block diagram

30.2.1 Overview

Basically the Quad Serial Peripheral Interface (QuadSPI) block serves for two different interfacing purposes: When acting as a Serial Peripheral Interface it provides a synchronous serial bus for communication with an external peripheral device. Alternatively the QuadSPI block can act as an interface to a SPI serial flash device. Refer to Section 30.2.3, QuadSPI modes of operation, for a description of the different modes.

30.2.2 Features

When acting as SPI the QuadSPI supports the following features

- Full-duplex, three-wire synchronous transfers
- Master and Slave mode
- Buffered transmit operation using the TX FIFO with parameterized depth of 15
- Buffered receive operation using the RX FIFO with parameterized depth of 15
- Programmable transfer attributes on a per-frame basis:
  - Parameterized number of transfer attribute registers (from two to eight)
  - Serial clock with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size of 4 to 16 bits, expandable by software control
  - Continuously held chip select capability
- 3 Peripheral Chip Selects
- DMA support for TX/RX path mutually exclusive with register interface. Available status information:
  - TX FIFO is not full
  - RX FIFO is not empty
- 6 Interrupt conditions:
  - End of queue reached
  - TX FIFO is not full
  - Transfer of current frame complete
  - Attempt to transmit with an empty Transmit FIFO
  - RX FIFO is not empty
  - Frame received while RX FIFO is full
- The 6 interrupt conditions are mapped to 5 interrupt lines
- Continuous serial communications clock

When acting as interface to a serial flash device the QuadSPI supports the following features:

- Compatible with Winbond™ SPI Serial Flash supporting single, dual and quad mode of operation.
- One chip select signal dedicated for usage with a serial flash device.
- DMA support to read RX Buffer data via AMBA AHB bus.

- 9 Interrupt conditions
- The 9 interrupt conditions are mapped to 5 different interrupt lines
- Memory-mapped read access to flash memory content in separate address range:
  - Supports flash devices of various sizes.
  - Appropriate command sequence for flash read triggered automatically by read access.
- Automatic divide-by-2 of the serial flash device clock for commands not supporting the full frequency range.

Additionally the following feature for power saving purposes is available independent from the external device the QuadSPI is interfacing with:

- Support for global signal Stop mode

30.2.3 QuadSPI modes of operation

30.2.3.1 SPI Master mode

In the SPI Master mode the QuadSPI can initiate transmission and reception of serial data to/from the external SPI device.Refer to Section 30.5.2.2, Master mode, for a detailed description. In this mode the QuadSPI uses the system clock as its timing reference.

30.2.3.2 SPI Slave mode

The Slave mode allows the QuadSPI to communicate with an external SPI bus master. Refer to Section 30.5.2.3, Slave mode, for a detailed description.

30.2.3.3 Serial Flash mode

In this mode an external serial flash memory device can be accessed. Further details about this mode of operation can be found in Section 30.5.3, SFM (Serial Flash) mode. In this mode the QuadSPI uses the auxiliary clock as its timing reference.

30.2.3.4 Module Disable mode

The Module Disable mode is used for power management of the device containing the QuadSPI module, it is controlled by signals external to the QuadSPI. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in the Module Disable mode. See Section 30.5.4.1, Module Disable mode.

30.2.3.5 Stop mode

The Stop mode is also used for power management. When a request is made to enter Stop mode, the QuadSPI block completes the action currently processed. Then the request is acknowledged.

30.2.3.6 Debug mode (SPI modes only)

The Debug mode is used for system development and debugging, it is controlled by additional logic outside of the module. When the MCU is stopped by a debugger and the QSPI_MCR[FRZ] bit is set QuadSPI stops all serial transfers.

30.3 External signal description

30.3.1 Overview

Table 30-4 lists the signals of the external signals belonging to the QuadSPI module in conjunction with the different modes of operation.

Table 30-4. Signal properties

Signal name	Function and direction					
	SPI Master mode		SPI Slave mode		SPI Slave mode	
PCS0_2	Peripheral Chip Select 0	Output	Slave Select	Input	Peripheral Chip Select	Output
PCS2_2, PCS1_2	Peripheral Chip Select 1–2	Output	Unused		Unused	
SIN_2	Serial Data In	Input	Serial Data In	Input	Serial I/O 0	Bidir
SOUT_2	Serial Data Out	Output	Serial Data Out	Tristate[1]	Serial I/O 1	Bidir
QSPI_IO2	Unused		Unused		Serial I/O 2	Bidir
QSPI_IO3	Unused		Unused		Serial I/O 3	Bidir
SCK_2	Serial Clock	Output	Serial Clock	Input	Serial Clock	Output

1 Driven only when the module is selected by the SPI master. HiZ otherwise.

Table 30-5 shows how the signals are connected on the MPC5606S in SPI and serial flash modes. Serial flash mode is selected by setting QSPI_MCR[QMODE]. Using a quad flash memory as an SPI device on the fly (that is, on-the-fly change of mode from SFM to SPI or vice versa) may not be possible for all flash memories. Check I/O compatibility before using this.

Table 30-5. Connectivity of signals on this device

Chip signal	SPI mode[1]	Serial flash mode[2]
PF[10]	CS_0 (Chip Select)	PCS (Chip Select Out)
PF[11]	CS_1 (Chip Select)	IO2 (Bidir)
PF[12]	CS_2	IO3 (Bidir)
PF[13]	SIN (Serial Data In)	IO0
PF[14]	SOUT (Serial Data Out)	IO1
PF[15]	SCK (Serial Clock)	Clock Out

1 When connecting to an external SPI device, these definitions should be used as reference.
2 When connection to a Quad Serial Flash, these definitions should be used as reference.

NOTE

When using single/dual SPI commands, the HOLD(IO3) and WP(IO2) pins are driven with high impedance. Some external memories, depending on their input characteristics, require their HOLD and WP pins to be driven high. The use of pullup resistors on these pins will ensure the correct value during single/dual instructions.

30.3.2 Detailed Signal Description

The following paragraphs describe the function of the signals given in Table 30-4 in more detail. Only the modes relevant to the specific signal are mentioned according to Table 30-4.

30.3.2.1 PCS_C0 — Peripheral Chip Select/Slave Select

In SPI Master mode, the PCS0 signal is a Peripheral Chip Select output that selects which slave device the current transmission is intended for.

In SPI Slave mode, the $\overline{SS}$ signal is a Slave Select input signal that allows a SPI master to select the QuadSPI as the target for transmission.

In Serial Flash mode this signal is the chip select for the serial flash device.

30.3.2.2 PCS[3:1] — Peripheral Chip Selects 1–3

PCS[3:1] are Peripheral Chip Select output signals in SPI Master mode.

30.3.2.3 PCS4 — Peripheral Chip Select 4

In SPI Master mode, PCS4 is a Peripheral Chip Select output signal.

30.3.2.4 PCS[7:5] — Peripheral Chip Selects 5–7

PCS[7:5] are Peripheral Chip Select output signals in SPI Master mode.

30.3.2.5 SI_IO0 — Serial Input, QuadSPI Data IO_0

SI is a serial data input signal in the SPI Master and SPI Slave mode.

In the Serial Flash mode it is used as I/O bit 0.

30.3.2.6 SO_IO1 — Serial Output, QuadSPI Data IO_1

SO is a serial data output signal in the SPI Master and SPI Slave mode. Note that the output buffer enable signal belonging to the SO output is driven by the Slave Select input signal SS to allow for several slaves

driving one single SO line. The SO output line is High Impedance when in SPI Slave mode the QuadSPI module is not selected by the SPI master.

In the Serial Flash mode it is used as I/O bit 1.

30.3.2.7 QSPI_IO2—QuadSPI Data IO_2

In Serial Flash mode this signal is used as I/O bit 2.

30.3.2.8 QSPI_IO3—QuadSPI Data IO_3

In Serial Flash mode this signal is used as I/O bit 3.

30.3.2.9 SCK — Serial Clock

SCK is a serial communication clock signal. In SPI Master mode, the QuadSPI generates the SCK from the system clock.

In SPI Slave mode, SCK is an input from an external bus master.

In Serial Flash mode this signal is the serial clock output to the serial flash device and is based on the auxiliary clock.

30.4 Memory map and register definition

30.4.1 IP bus register memory map

Table 30-6 shows the QuadSPI memory map.

Table 30-6. QuadSPI IP bus memory map

Address	Register name
Global register for SPI modes and SFM mode	
QSPI_BASE+0x000	Module Configuration Register (QSPI_MCR)
QSPI_BASE+0x004	Reserved
Registers valid in SPI modes only[1]	
QSPI_BASE+0x008	Transfer Count Register (QSPI_TCR)
QSPI_BASE+0x00C – QSPI_BASE+0x010	Clock and Transfer Attributes Registers 0 – 1 (QSPI_CTAR0 – QSPI_CTAR1)
QSPI_BASE+0x014 – QSPI_BASE+0x028	Reserved
QSPI_BASE+0x02C	SPI Status Register (QSPI_SPISR)
QSPI_BASE+0x030	SPI Interrupt and DMA Request Select and Enable Register (QSPI_SPIRSER)

Table 30-6. QuadSPI IP bus memory map (continued)

Address	Register name
FIFO Registers	
QSPI_BASE+0x034	PUSH TX FIFO Register (QSPI_PUSHR)
QSPI_BASE+0x038	POP RX FIFO Register (QSPI_POPR)
QSPI_BASE+0x03C – QSPI_BASE+0x074	Transmit FIFO Registers 0 – 14 (QSPI_TXFR0 – QSPI_TXFR14)
QSPI_BASE+0x078	Reserved
QSPI_BASE+0x07C – QSPI_BASE+0x0B4	RX FIFO Registers 0 – 14 (QSPI_RXFR0 – QSPI_RXFR14)
QSPI_BASE+0x0B8 – QSPI_BASE+0x0FC	Reserved
Registers valid in SFM mode only[2]	
QSPI_BASE+0x100	Serial Flash Address Register (QSPI_SFAR)
QSPI_BASE+0x104	Instruction Code Register (QSPI_ICR)
QSPI_BASE+0x108	Sampling Register (QSPI_SMPR)
QSPI_BASE+0x10C	RX Buffer Status Register (QSPI_RBSR)
QSPI_BASE+0x110 – QSPI_BASE+0x148	RX Buffer Data Registers 0–14 (QSPI_RBDR0–QSPI_RBDR14)
QSPI_BASE+0x14C	Reserved
QSPI_BASE+0x150	TX Buffer Status Register (QSPI_TBSR)
QSPI_BASE+0x154	TX Buffer Data Register (QSPI_TBDR)
QSPI_BASE+0x158	AMBA Control Register (QSPI_ACR)
QSPI_BASE+0x15C	Serial Flash Mode Status Register (QSPI_SFMSR)
QSPI_BASE+0x160	Serial Flash Mode Flag Register (QSPI_SFMFR)
QSPI_BASE+0x164	SFM Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER)
QSPI_BASE+0x168 – QSPI_BASE+0x1FC	Reserved

1 These registers must not be written if the QuadSPI module is in SFM mode.

2 These registers must not be written if the QuadSPI module is in SPI Master or SPI Slave mode

30.4.2 AMBA bus register memory map

QSPI_AMBA_BASE defines the address to be used as start address of the serial flash device.

Table 30-7. QuadSPI AMBA bus memory map

Address	Register Name
Memory-mapped serial flash data (QSPI_SFD)	
QSPI_AMBA_BASE + 0x000_0000	Serial Flash Data [0x000_0000]
QSPI_AMBA_BASE + 0x000_0004	Serial Flash Data [0x000_0004]
...	...
QSPI_AMBA_BASE + 0x7FF_FFF4	Serial Flash Data [0x7FF_FFF4]
QSPI_AMBA_BASE + 0x7FF_FFF8	Serial Flash Data [0x7FF_FFF8]
AHB RX Data Buffer (QSPI_ARDB)	
QSPI_AMBA_BASE + 0x7FF_FFFC	AHB RX Data Buffer (QSPI_ARDB)

30.4.3 IP bus register descriptions

30.4.3.1 Register write access

This section describes the write access restriction terms that apply to all registers.

30.4.3.1.1 Register write access restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in Table 30-8. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed.

The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled.

Table 30-8. Register write access restrictions

Condition	Description
Anytime	No write access restriction.
Disabled mode	Write access only if the module is in *Module Disable mode*.
Enabled mode	Write access only if the module is in *SPI Master mode* or *SPI Slave mode* or *Serial Flash mode*.
SPI mode	Write access only if the module is in *SPI Master mode* or *SPI Slave mode*.
Serial Flash mode	Write access only if the module is in *Serial Flash mode*.

30.4.3.1.2 Register write access requirements

All registers can be accessed with 8-bit, 16-bit and 32-bit wide operations. For some of the registers, at least a 16/32-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each register affected

30.4.3.2 Module Configuration Register (QSPI_MCR)

The QSPI_MCR contains bits which configure various attributes associated with QuadSPI operation.

NOTE

Configuration data depending from the value of the QMODE bit must be left at their reset values when they are not applicable.

Address: QSPI_BASE + 0x000

Write: QMODE: Disabled mode
DIS_TXF, DIS_RXF: Enabled mode
All Other: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	CONT_SCKE	0	0	FRZ	MTFE	PCSSE	ROOE	PCSIS7	PCSIS6	PCSIS5	PCSIS4	PCSIS3	PCSIS2	PCSIS1	PCSIS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MDIS	DIS_TXF	DIS_RXF	0	0	SMPL_PT		QMODE	VMID				0	0	HALT
W					CLR_TXF	CLR_RXF										
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 30-2. Module Configuration Register (QSPI_MCR)

Table 30-9. QSPI_MCR field descriptions

Field	Description
MSTR	Master/Slave mode Select. **Only applicable if QMODE is cleared**. The MSTR bit configures the QuadSPI for either SPI Master mode or SPI Slave mode. 0 QuadSPI is in SPI Slave mode 1 QuadSPI is in SPI Master mode
CONT_SCKE	Continuous SCK Enable. **Only applicable if QMODE is cleared**. The CONT_SCKE bit enables the Serial Communication Clock (SCK) to run continuously. See Section 30.5.2.9, Continuous Serial Communications Clock, for details. 0 Continuous SCK disabled 1 Continuous SCK enabled
FRZ	Freeze. **Only applicable if QMODE is cleared**. The FRZ bit enables the QuadSPI transfers to be stopped on the next frame boundary when the MCU is stopped by a debugger. 0 Do not halt serial transfers 1 Halt serial transfers
MTFE	Modified Timing Format Enable. **Only applicable if QMODE is cleared**. The MTFE bit enables a modified transfer format to be used. See Section 30.5.2.8.4, Modified SPI Transfer Format (MTFE = 1, CPHA = 1), for more information. 0 Modified SPI transfer format disabled 1 Modified SPI transfer format enabled

Table 30-9. QSPI_MCR field descriptions (continued)

Field	Description
PCSSE	Peripheral Chip Select Strobe Enable. **Only applicable if QMODE is cleared**. The PCSSE bit enables the PCS[5]/$\overline{\text{PCSS}}$ to operate as an PCS Strobe output signal. See Section 30.5.2.7.5, Peripheral Chip Select Strobe Enable (PCSS), for more information. 0 PCS5/$\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal 1 PCS5/$\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal
ROOE	RX FIFO Overflow Overwrite Enable. **Only applicable if QMODE is cleared.** The ROOE bit enables an RX FIFO overflow condition to either ignore the incoming serial data or to overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer that generated the overflow is either ignored or shifted into the shift register. If the ROOE bit is asserted, the incoming data is shifted into the shift register. If the ROOE bit is negated, the incoming data is ignored. See Section 30.5.2.10.6, RX FIFO Overflow Interrupt Request, for more information. 0 Incoming data is ignored 1 Incoming data is shifted into the shift register
PCSIS*x*	Peripheral Chip Select Inactive State. **Only applicable if QMODE is cleared.** The PCSIS bit determines the inactive state of the PCS*x* signal. 0 The inactive state of PCS*x* is low 1 The inactive state of PCS*x* is high
MDIS	Module Disable. The MDIS bit allows the clock to be stopped to the non-memory mapped logic in the QuadSPI effectively putting the QuadSPI in a software controlled power-saving state. See Section 30.5.4, Power saving features, and Section 30.5.2.1, Start and Stop of SPI Transfers, for more information. 0 Enable QuadSPI clocks. 1 Allow external logic to disable QuadSPI clocks.
DIS_TXF	Disable TX FIFO. **Only applicable if QMODE is cleared.** The DIS_TXF bit provides a mechanism to disable the TX FIFO. When the TX FIFO is disabled, the transmit part of the QuadSPI operates as a simplified double-buffered SPI. See Section 30.5.2.4, FIFO Disable Operation, for details. 0 TX FIFO is enabled 1 TX FIFO is disabled
DIS_RXF	Disable RX FIFO. **Only applicable if QMODE is cleared.** The DIS_RXF bit provides a mechanism to disable the RX FIFO. When the RX FIFO is disabled, the receive part of the QuadSPI operates as a simplified double-buffered SPI. See Section 30.5.2.4, FIFO Disable Operation, for details. 0 RX FIFO is enabled 1 RX FIFO is disabled
CLR_TXF	Clear TX FIFO/Buffer. Depending from the QMODE bit CLR_TXF is used to invalidate the TX FIFO or the TX Buffer content. 0 No action 1 (QMODE bit cleared): Read and write pointers of RX FIFO are reset to 0. QSPI_SPISR[TXCTR] and QSPI_SPISR[TXNXTPTR] are reset to 0. 1 (QMODE bit set): Read and write pointers of the RX Buffer are reset to 0. QSPI_TBSR[TRCTR] is reset to 0.
CLR_RXF	Clear RX FIFO. Depending from the QMODE bit CLR_RXF is used to invalidate the RX FIFO or the RX Buffer. 0 No action 1 (QMODE bit cleared): Read and write pointers of the RX FIFO are reset to 0. QSPI_SPISR[RXCTR] and QSPI_SPISR[POPNXTPTR] are reset to 0. 1 (QMODE bit set): Read and write pointers of the RX Buffer are reset to 0. QSPI_RBSR[RDBFL] is reset to 0.

Table 30-9. QSPI_MCR field descriptions (continued)

Field	Description
SMPL_PT	SMPL_PT — **Only applicable if QMODE is cleared**. Sample Point. SMPL_PT allows the host software to select when the QuadSPI Master samples SI in Modified Transfer Format. The table below lists the various delayed sample points. 00 0 system clock cycles 01 1 system clock cycle 10 2 system clock cycles 11 Reserved
QMODE	QMODE — QuadSPI Mode: When this bit is cleared the QuadSPI block is in SPI Master or SPI Slave mode. When this bit is set the QuadSPI block is in Serial Flash mode. 0 Module is in SPI Master or SPI Slave mode 1 Module is in SFM mode
VMID	VMID — Vendor Model ID. **Only applicable if QMODE is set.** 0000 Reserved 0001 Winbond othersReserved
HALT	Halt. **Only applicable if QMODE is cleared.** The HALT bit provides a mechanism by software to start and stop QuadSPI transfers. See Section 30.5.2.1, Start and Stop of SPI Transfers, for details about the operation of this bit. 0 Start transfers 1 Stop transfers

30.4.3.3 Transfer Count Register (QSPI_TCR)

The QSPI_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management.

Address: QSPI_BASE + 0x008

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

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

Figure 30-3. Transfer Count Register (QSPI_TCR)

Table 30-10. QSPI_TCR field descriptions

Field	Description
SPI_TCNT	SPI Transfer Counter. SPI_TCNT is used to keep track of the number of SPI transfers made. The SPI_TCNT field counts the number of SPI transfers the QuadSPI makes. The SPI_TCNT field is incremented every time the last bit of a SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI Command. The Transfer Counter wraps around, i.e., incrementing the counter past 0xFFFF resets the counter to zero.

30.4.3.4 Clock and Transfer Attributes Registers 0 – 1 (QSPI_CTAR0 – QSPI_CTAR1)

The QSPI_CTAR registers are used to define different transfer attribute configurations for the SPI Master mode and the SPI Slave mode. SPI transfers select which one of the QSPI_CTARs to get their transfer attributes from. In the current implementation there are 2 different QSPI_CTARs selectable. The user must not write to the QSPI_CTAR registers while the QuadSPI is in the Running state.

In Master mode, the QSPI_CTAR0–QSPI_CTAR7 registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In Slave mode, a subset of the bitfields in the QSPI_CTAR0 and QSPI_CTAR1 registers are used to set the slave transfer attributes. See the individual bit descriptions for details on which bits are used in Slave modes.

When the QuadSPI is configured as a SPI Master, the CTAS field in the command portion of the TX FIFO entry selects which of the QSPI_CTAR register is used. When the QuadSPI is configured as a SPI bus Slave, the QSPI_CTAR0 register is used.

Address: QSPI_BASE + 0x00C (QSPI_CTAR0)
QSPI_BASE + 0x010 (QSPI_CTAR1)

Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-4. Clock and Transfer Attributes Registers 0 – 1 (QSPI_CTAR0 – QSPI_CTAR1)

Table 30-11. QSPI_CTAR*n* field descriptions

Field	Descriptions
DBR	Double Baud Rate. The DBR bit doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in Master mode. It effectively halves the Baud Rate division ratio supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in Table 30-12. See the BR[0:3] field description for details on how to compute the baud rate. If the overall baud rate is divide by two or divide by three of the system clock then neither the Continuous SCK Enable or the Modified Timing Format Enable bits should be set. 0 The baud rate is computed normally with a 50/50 duty cycle 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler
FMSZ	Frame Size. The FMSZ field selects the number of bits transferred per frame. The FMSZ field is used in Master mode and Slave mode. Table 30-13 lists the frame size encodings.
CPOL	Clock Polarity. The CPOL bit selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both Master and Slave modes. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the QuadSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low 1 The inactive state value of SCK is high
CPHA	Clock Phase. The CPHA bit selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both Master and Slave modes. For successful communication between serial devices, the devices must have identical clock phase settings. Continuous SCK is only supported for CPHA=1. 0 Data is captured on the leading edge of SCK and changed on the following edge 1 Data is changed on the leading edge of SCK and captured on the following edge
LSBFE	LSB First. The LSBFE bit selects if the LSB or MSB of the frame is transferred first. This bit is used only in Master mode. 0 Data is transferred MSB first 1 Data is transferred LSB first
PCSSCK	PCS to SCK Delay Prescaler. The PCSSCK field selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. This field is used only in Master mode. Table 30-14 lists the prescaler values and the associated bit settings. See the CSSCK[0:3] field description for details on how to compute the PCS to SCK Delay. 00 Prescaler value 1 01 Prescaler value 3 10 Prescaler value 5 11 Prescaler value 7
PASC	After SCK Delay Prescaler. The PASC field selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. This field is used only in Master mode. Table 30-15 lists the prescaler values. See the ASC[0:3] field description for details on how to compute the After SCK Delay. 00 Prescaler value 1 01 Prescaler value 3 10 Prescaler value 5 11 Prescaler value 7

Table 30-11. QSPI_CTAR*n* field descriptions (continued)

Field	Descriptions
PDT	Delay after Transfer Prescaler. The PDT field selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is used only in Master mode. Table 30-16 lists the prescaler values. See the DT[0:3] field description for details on how to compute the Delay after Transfer. 00 Delay after Transfer Prescaler value 1 01 Delay after Transfer Prescaler value 3 10 Delay after Transfer Prescaler value 5 11 Delay after Transfer Prescaler value 7
PBR	Baud Rate Prescaler. The PBR field selects the prescaler value for the baud rate. This field is used only in Master mode. The Baud Rate is the frequency of the Serial Communications Clock (SCK). The system clock is divided by the prescaler value before the baud rate selection takes place. The Baud Rate Prescaler values are listed in Table 30-17. See the BR[0:3] field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value 1 01 Baud Rate Prescaler value 3 10 Baud Rate Prescaler value 5 11 Baud Rate Prescaler value 7
CSSCK	PCS to SCK Delay Scaler. The CSSCK field selects the scaler value for the PCS to SCK delay. This field is used only in Master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. Table 30-14 list the scaler values.The PCS to SCK Delay is a multiple of the system clock period and it is computed according to the following equation: $t_{CSC} = \frac{1}{f_{SYS}} \times PCSSCK \times CSSCK$ ***Eqn. 30-1*** See Section 30.5.2.7.2, PCS to SCK Delay (tCSC), for more details.
ASC	After SCK Delay Scaler. The ASC field selects the scaler value for the After SCK Delay. This field is used only in Master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. Table 30-15 list the scaler values.The After SCK Delay is a multiple of the system clock period, and it is computed according to the following equation: $t_{ASC} = \frac{1}{f_{SYS}} \times PASC \times ASC$ ***Eqn. 30-2*** See Section 30.5.2.7.3, After SCK Delay (tASC), for more details.
DT	Delay after Transfer Scaler. The DT field selects the Delay after Transfer Scaler. This field is used only in Master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. Table 30-16 lists the scaler values. In the Continuous Serial Communications Clock operation the DT value is fixed to one TSCK. $t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT$ ***Eqn. 30-3*** See Section 30.5.2.7.4, Delay after Transfer (tDT), for more details.
BR	Baud Rate Scaler. The BR field selects the scaler value for the baud rate. This field is used only in Master mode. The pre-scaled system clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. Table 30-17 lists the Baud Rate Scaler values. The baud rate is computed according to the following equation: $\text{SCK baud rate} = \frac{f_{SYS}}{PBR} \times \frac{1 + DBR}{BR}$ ***Eqn. 30-4*** See Section 30.5.2.7.1, Baud Rate Generator, for more details.

Table 30-12. SCK Duty Cycle

DBR	CPHA	PBR	SCK Duty Cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 30-13. Transfer Frame Size

FMSZ	Framesize	FMSZ	Framesize
0000	Reserved	1000	9
0001	Reserved	1001	10
0010	Reserved	1010	11
0011	4	1011	12
0100	5	1100	13
0101	6	1101	14
0110	7	1110	15
0111	8	1111	16

Table 30-14. PCS to SCK Delay Scaler

CSSCK	PCS to SCK Delay Scaler Value	CSSCK	PCS to SCK Delay Scaler Value
0000	2	1000	512
0001	4	1001	1024
0010	8	1010	2048
0011	16	1011	4096
0100	32	1100	8192
0101	64	1101	16384
0110	128	1110	32768
0111	256	1111	65536

Table 30-15. After SCK Delay Scaler

ASC	After SCK Delay Scaler Value	ASC	After SCK Delay Scaler Value
0000	2	1000	512
0001	4	1001	1024
0010	8	1010	2048
0011	16	1011	4096
0100	32	1100	8192
0101	64	1101	16384
0110	128	1110	32768
0111	256	1111	65536

Table 30-16. After Transfer Scaler

DT	Delay after Transfer Scaler Value	DT	Delay after Transfer Scaler Value
0000	2	1000	512
0001	4	1001	1024
0010	8	1010	2048
0011	16	1011	4096
0100	32	1100	8192
0101	64	1101	16384
0110	128	1110	32768
0111	256	1111	65536

Table 30-17. Baud Rate Scaler

BR	Baud Rate Scaler Value	BR	Baud Rate Scaler Value
0000	2	1000	256
0001	4	1001	512
0010	6	1010	1024
0011	8	1011	2048
0100	16	1100	4096
0101	32	1101	8192
0110	64	1110	16384
0111	128	1111	32768

30.4.3.5 SPI Status Register (QSPI_SPISR)

The QSPI_SPISR contains status and flag bits. The bits reflect the status of the QuadSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the QSPI_SPISR by writing a 1 to it. Writing a 0 to a flag bit has no effect. This register may not be writable in MDIS mode due to the use of power saving mechanisms.

Address: QSPI_BASE + 0x02C Write: Enabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R[1]	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	0	0	RFOF	0	RFDF	0
W	w1c			w1c	w1c		w1c						w1c		w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R[1]	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-5. SPI Status Register (QSPI_SPISR)

[1]When in SFM mode all 0's are read.

Table 30-18. QSPI_SPISR field descriptions

Field	Description
TCF	Transfer Complete Flag. The TCF bit indicates that all bits in a frame have been shifted out. The TCF bit is set at the end of the frame transfer. The TCF bit remains set until cleared by software. 0 Transfer not complete 1 Transfer complete
TXRXS	TX & RX Status. The TXRXS bit reflects the status of the QuadSPI. See Section 30.5.2.1, Start and Stop of SPI Transfers, for information on how what causes this bit to be negated or asserted. 0 TX and RX operations are disabled (QuadSPI is in Stopped state) 1 TX and RX operations are enabled (QuadSPI is in Running state)
EOQF	End of Queue Flag. The EOQF bit indicates that transmission in progress is the last entry in a queue. The EOQF bit is set when TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by software. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command 1 EOQ bit is set in the executing SPI Command
TFUF	TX FIFO Underrun Flag. The TFUF bit indicates that an underrun condition in the TX FIFO has occurred. The transmit underrun condition is detected only in SPI Slave mode. The TFUF bit is set when the TX FIFO of a QuadSPI operating in SPI Slave mode is empty, and a transfer is initiated by an external SPI master. The TFUF bit remains set until cleared by software. 0 TX FIFO underrun has not occurred 1 TX FIFO underrun has occurred

Table 30-18. QSPI_SPISR field descriptions (continued)

Field	Description
TFFF	TX FIFO Fill Flag. The TFFF bit provides a method for the QuadSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by host software or an acknowledgement from the DMA controller when the TX FIFO is full. 0 TX FIFO is full 1 TX FIFO is not full
RFOF	RX FIFO Overflow Flag. The RFOF bit indicates that an overflow condition in the RX FIFO has occurred. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until cleared by software. 0 RX FIFO overflow has not occurred 1 RX FIFO overflow has occurred
RFDF	RX FIFO Drain Flag. The RFDF bit provides a method for the QuadSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by host software or an acknowledgement from the DMA controller when the RX FIFO is empty. 0 RX FIFO is empty 1 RX FIFO is not empty
TXCTR	TX FIFO Counter. The TXCTR field indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the QSPI_PUSHR is written. The TXCTR is decremented every time a SPI Command is executed and the SPI data is transferred to the shift register.
TXNXTPTR	Transmit Next Pointer. The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. See Section 30.5.2.10.4, Transmit FIFO Underrun Interrupt Request, for more details.
RXCTR	RX FIFO Counter. The RXCTR field indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the QSPI_POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
POPNXTPTR	Pop Next Pointer. The POPNXTPTR field contains a pointer to the RX FIFO entry that will be returned when the QSPI_POPR is read. The POPNXTPTR is updated when the QSPI_POPR is read. See Section 30.5.2.6, Receive First In First Out (RX FIFO) Buffering Mechanism, for more details.

30.4.3.6 SPI Interrupt and DMA Request Select and Enable Register (QSPI_SPIRSER)

The QSPI_SPIRSER serves two purposes. It enables flag bits in the QSPI_SPISR to generate DMA requests or interrupt requests. The QSPI_SPIRSER also selects the type of request to be generated. See the individual bit descriptions for information on the types of requests the bits support. The user must not write to the QSPI_SPIRSER while the QuadSPI is in the Running state.

Address: QSPI_BASE + 0x030 Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_IE	0	0	EOQF_IE	TFUF_IE	0	TFFF_RE	TFFF_DIRS	0	0	0	0	RFOF_IE	0	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 30-6. SPI Interrupt and DMA Request Select and Enable Register (QSPI_SPIRSER)

Table 30-19. QSPI_SPIRSER field descriptions

Field	Description
TCF_IE	Transmission Complete IRQ Enable. The TCF_IE bit enables the TCF flag in the QSPI_SPISR to generate an interrupt request. 0 TCF interrupt requests are disabled 1 TCF interrupt requests are enabled
EOQF_RE	QuadSPI Finished IRQ Enable. The EOQF_IE bit enables the EOQF flag in the QSPI_SPISR to generate an interrupt request. 0 EOQF interrupt requests are disabled 1 EOQF interrupt requests are enabled
TFUF_RE	TX FIFO Underrun IRQ Enable. The TFUF_IE bit enables the TFUF flag in the QSPI_SPISR to generate an interrupt request. 0 TFUF interrupt requests are disabled 1 TFUF interrupt requests are enabled
TFFF_RE	TX FIFO Fill Request Enable. The TFFF_RE bit enables the TFFF flag in the QSPI_SPISR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA requests. 0 TFFF interrupt requests or DMA requests are disabled 1 TFFF interrupt requests or DMA requests are enabled
TFFF_DIRS	TX FIFO Fill DMA or Interrupt Request Select. The TFFF_DIRS bit selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the QSPI_SPISR is set, and the TFFF_RE bit in the QSPI_SPIRSER register is set, this bit selects between generating an interrupt request or a DMA request. 0 Interrupt request will be generated 1 DMA request will be generated
RFOF_IE	RX FIFO Overflow IRQ Enable. The RFOF_IE bit enables the RFOF flag in the QSPI_SPISR to generate an interrupt requests. 0 RFOF interrupt requests are disabled 1 RFOF interrupt requests are enabled

Table 30-19. QSPI_SPIRSER field descriptions (continued)

Field	Description
RFDF_RE	RX FIFO Drain Request Enable. The RFDF_RE bit enables the RFDF flag in the QSPI_SPISR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt requests or DMA requests are disabled 1 RFDF interrupt requests or DMA requests are enabled
RFDF_DIRS	RX FIFO Drain DMA or Interrupt Request Select. The RFDF_DIRS bit selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the QSPI_SPISR is set, and the RFDF_RE bit in the QSPI_SPIRSER register is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request will be generated 1 DMA request will be generated

30.4.3.7 PUSH TX FIFO Register (QSPI_PUSHR)

The QSPI_PUSHR provides a means to write to the TX FIFO and to configure the parameters for the SPI transmission. Data written to the TXDATA field is transferred to the TX FIFO. See Section 30.5.2.5, Transmit First In First Out (TX FIFO) Buffering Mechanism, for more information and Table 30-45 for the byte ordering.

Address: QSPI_BASE + 0x034 — 32-bit write access required — Write: SPI mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CONT	CTAS			EOQ	CTCNT	0	0	PCS7	PCS6	PCS5	PCS4	PCS3	PCS2	PCS1	PCS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-7. PUSH TX FIFO Register (QSPI_PUSHR)

Table 30-20. QSPI_PUSHR field descriptions

Field	Descriptions
CONT	Continuous Peripheral Chip Select Enable. The CONT bit selects a Continuous Selection Format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. See Section 30.5.2.8.5, Continuous Selection Format, for more information. 0 Return Peripheral Chip Select signals to their inactive state between transfers 1 Keep Peripheral Chip Select signals asserted between transfers
CTAS	Clock and Transfer Attributes Select. The CTAS field selects which of the QSPI_CTAR register is used to set the clock and transfer attributes for the associated SPI frame. The field is used only in SPI Master mode. In SPI Slave mode QSPI_CTAR0 is used. The table below shows how the CTAS values map to the QSPI_CTAR registers. All values not given below are reserved. 000 QSPI_CTAR0 001 QSPI_CTAR1
EOQ	End Of Queue. The EOQ bit provides a means for host software to signal to the QuadSPI that the current SPI transfer is the last in a queue. At the end of the transfer the EOQF bit in the QSPI_SPISR is set. 0 The SPI data is not the last data to transfer 1 The SPI data is the last data to transfer
CTCNT	Clear SPI_TCNT. The CTCNT provides a means for host software to clear the SPI transfer counter. The CTCNT bit clears the SPI_TCNT field in the QSPI_TCR register. The SPI_TCNT field is cleared before transmission of the current SPI frame begins. 0 Do not clear SPI_TCNT field in the QSPI_TCR 1 Clear SPI_TCNT field in the QSPI_TCR
PCS*x*	Peripheral Chip Select 0–7. The PCS bits select which PCS signals will be asserted for the transfer. 0 Negate the PCSx signal 1 Assert the PCSx signal
TXDATA	TX Data. Writing the TXDATA field pushed the SPI data to be transferred onto the TX FIFO. Reading the TXDATA field provides the value which was written most recently into the TXDATA field. After the TX FIFO has been cleared the result is undefined.

30.4.3.8 POP RX FIFO Register (QSPI_POPR)

The QSPI_POPR provides a means to read the RX FIFO. See Section 30.5.2.6, Receive First In First Out (RX FIFO) Buffering Mechanism, for a description of the RX FIFO operations. Eight or sixteen bit read accesses to the QSPI_POPR will read from the RX FIFO and update the counter and pointer. Refer to Table 30-45 for the byte ordering scheme.

Address: QSPI_BASE + 0x038

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-8. POP RX FIFO Register (QSPI_POPR)

Table 30-21. QSPI_POPR field descriptions

Field	Description
RXDATA	RX Data. The RXDATA field contains the SPI data from the RX FIFO entry pointed to by the Pop Next Data Pointer.

30.4.3.9 Transmit FIFO Registers 0 – 14 (QSPI_TXFR0 – QSPI_TXFR14)

The QSPI_TXFR0–QSPI_TXFR14 registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified.

Address : QSPI_BASE+0x03C (QSPI_TXFR0)
...
QSPI_BASE+0x074 (QSPI_TXFR14)

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-9. Transmit FIFO Registers 0 – 14 (QSPI_TXFR0 – QSPI_TXFR14)

Table 30-22. QSPI_TXFR*n* field descriptions

Field	Description
TXCMD	TX Command. The TXCMD field contains the command that sets the transfer attributes for the SPI data. See Section 30.4.3.7, PUSH TX FIFO Register (QSPI_PUSHR), for details on the command field.
TXDATA	TX Data. The TXDATA field contains the SPI data to be shifted out.

30.4.3.10 RX FIFO Registers 0 – 14 (QSPI_RXFR0 – QSPI_RXFR14)

The QSPI_RXFR0–QSPI_RXFR14 registers provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The QSPI_RXFR registers are read-only. Reading the QSPI_RXFRX registers does not alter the state of the RX FIFO. The RX FIFO implemented has a depth of 15 entries, so the highest address usable belongs to QSPI_RXFR14. Refer to Table 30-45 for the byte ordering scheme.

Address: QSPI_BASE + 0x07C (QSPI_RXFR0)
...
QSPI_BASE + 0x0B4 (QSPI_RXFR14)

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-10. RX FIFO Registers 0 – 14 (QSPI_RXFR0 – QSPI_RXFR14)

Table 30-23. QSPI_RXFR*n* field descriptions

Field	Description
RXDATA	RX Data. The RXDATA field contains the received SPI data.

30.4.3.11 Serial Flash Address Register (QSPI_SFAR)

The Serial Flash Address Register contains the address for the next IP command. The number of bits used from this register depend on the instruction code of the SFM command. See Section 30.5.3, SFM (Serial Flash) mode, for details.

Address: QSPI_BASE + 0x100 Write: QSPI_SFMSR[IP_ACC] = 0

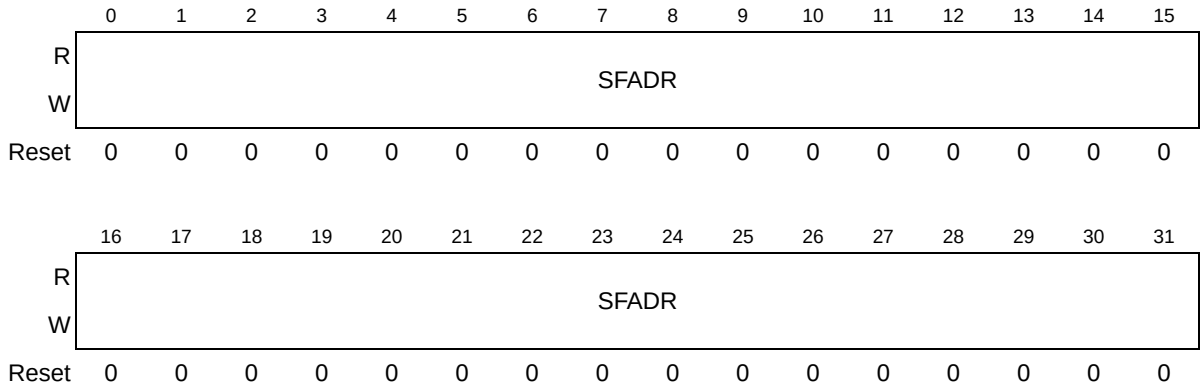


Figure 30-11. Serial Flash Address Register (QSPI_SFAR)

Table 30-24. QSPI_SFAR field descriptions

Field	Description
SFADR	Serial Flash Address, register content is used as address for all following IP Commands.

30.4.3.12 Instruction Code Register (QSPI_ICR)

The Instruction Code Register consists of the generic instruction code (IC) and an additional parameter section (ICO). This contains additional options to parameterize the command as shown in Table 30-53.

If the IC field is written successfully—while the module is in Serial Flash mode and not busy—a new command to the external serial flash device is started with that instruction code if this code is supported by the module (see 30.7.1, Supported Instruction Codes in Winbond Devices).

The QSPI_ICR register is writable only in SFM mode.

Refer to Section 30.5.3.1, Issuing SFM Commands, for further details about the triggering of IP Commands.

Address: QSPI_BASE + 0x104

Write: ICO: QSPI_SFMSR[IP_ACC] = 0 and QSPI_MCR[QMODE] = 1
IC: QSPI_SFMSR[IP_ACC] = 0 and QSPI_SFMSR[AHB_ACC] = 0 and QSPI_MCR[QMODE] = 1

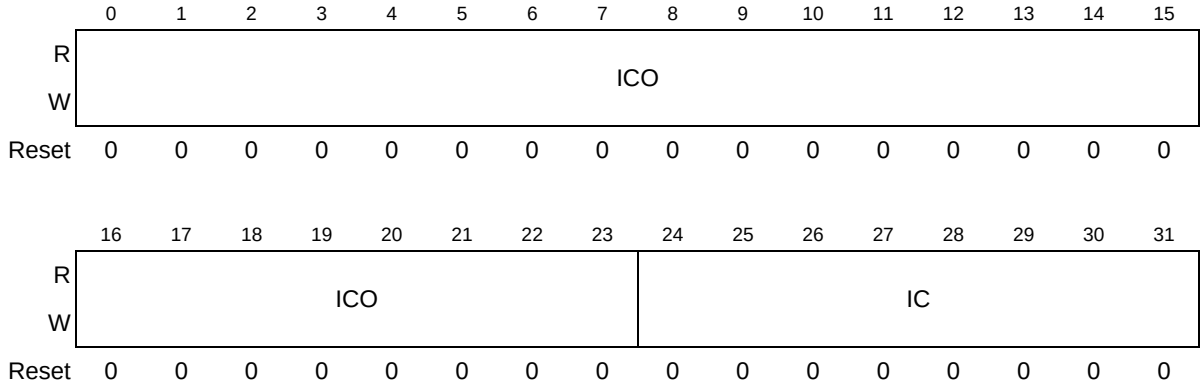


Figure 30-12. Instruction Code Register (QSPI_ICR)

Table 30-25. QSPI_ICR field descriptions

Field	Description
ICO	Instruction Code Options, additional parameters for the IC instruction specified in the IC field. Meaning of the individual bits vary for each instruction code and vendor, detailed description in Table 30-53. Only applicable in SFM mode
IC	Write access: Instruction Code of the SFM command to be executed next. Read access: Instruction Code of the last SFM command successfully written. Upon writing this byte a new command sequence is started to the external serial flash device, if the module is in serial flash mode. Only applicable in SFM mode

30.4.3.13 Sampling Register (QSPI_SMPR)

The Sampling Register allows configuration of the scheme how the incoming data from an external serial flash device are sampled in the QuadSPI module. Its fields are relevant in SFM mode only.

Address: QSPI_BASE + 0x108 Write: Disabled mode

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FSDLY	FSPHS	0	0	HSDLY	HSPHS	HSENA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-13. Sampling Register (QSPI_SMPR)

Table 30-26. QSPI_SMPR field descriptions

Field	Description
FSDLY	Full Speed Delay selection. Select the delay with respect to the reference edge for the sample point valid for full speed commands: 0: One clock cycle delay 1: Two clock cycles delay
FSPHS	Full Speed Phase selection. Select the edge of the sampling clock valid for full speed commands: 0: Select sampling at non-inverted clock 1: Select sampling at inverted clock
HSDLY	Half Speed Delay selection. **Only relevant when HSENA bit is set**. Select the delay with respect to the reference edge for the sample point valid for half speed commands: 0: One clock cycle delay 1: Two clock cycles delay
HSPHS	Half Speed Phase selection. **Only relevant when HSENA bit is set**. Select the edge of the sampling clock valid for half speed commands: 0: Select sampling at non-inverted clock 1: Select sampling at inverted clock
HSENA	Half Speed serial flash clock Enable: This bit enables the divide by 2 of the clock to the external serial flash device for specific commands. Refer to Section 30.7.2, Serial Flash Clock Frequency Limitations, for details. 0: Disable divide by 2 of serial flash clock for half speed commands 1: Enable divide by 2 of serial flash clock for half speed commands

30.4.3.14 RX Buffer Status Register (QSPI_RBSR)

This register contains information related to the receive data buffer.

Address: QSPI_BASE + 0x10C

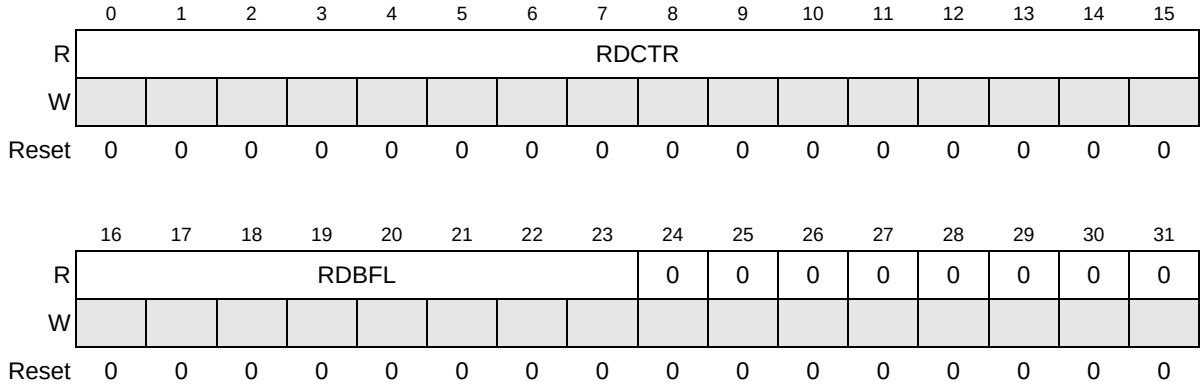


Figure 30-14. RX Buffer Status Register (QSPI_RBSR)

Table 30-27. QSPI_RBSR field descriptions

Field	Description
RDCTR	Read Counter, indicates how many bytes have been removed from the RX Buffer by host accesses. It is incremented by 4 on each successful write on the QSPI_SFMFR[RBDF] bit. For further details please refer to Section 30.4.4.3, AHB RX Data Buffer (QSPI_ARDB), and Section 30.5.3.3.2, Host Read of the QuadSPI Module Internal Buffers.
RBBFL	RX Buffer Fill Level, indicates how many bytes are still available for read in the RX Buffer.

30.4.3.15 RX Buffer Data Registers 0–14 (QSPI_RBDR0–QSPI_RBDR14)

The QSPI_RBDR registers provide access to the individual entries in the RX Buffer. Refer to Table 30-45 for the byte ordering scheme.

Address: QSPI_BASE + 0x110 (QSPI_RBDR0)
...
QSPI_BASE + 0x148 (QSPI_RBDR14)

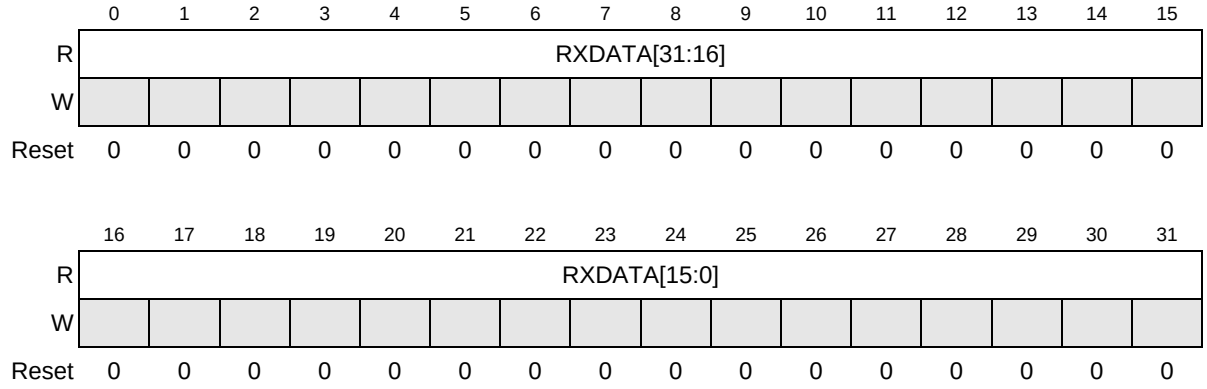


Figure 30-15. RX Buffer Data Registers 0–14 (QSPI_RBDR0–QSPI_RBDR14)

Table 30-28. QSPI_RBDR field descriptions

Field	Description
RXDATA	RX Data. The RXDATA field contains the received data.

30.4.3.16 TX Buffer Status Register (QSPI_TBSR)

This register contains information related to the transmit data buffer.

Address: QSPI_BASE + 0x150

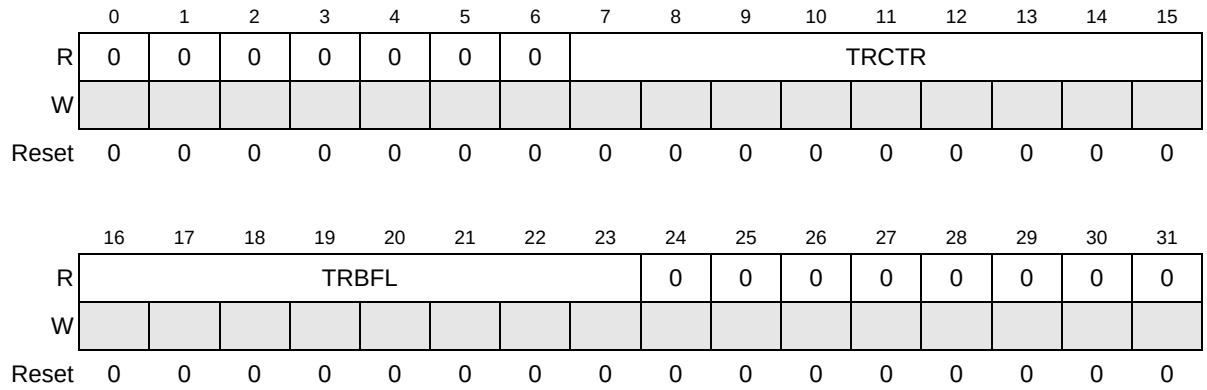


Figure 30-16. TX Buffer Status Register (QSPI_TBSR)

Table 30-29. QSPI_TBSR field descriptions

Field	Description
TRCTR	Transmit Counter. This field indicates how many bytes have been written into the TX Buffer by host accesses. It is reset to 0 when a 1 is written into the QSPI_MCR[CLR_TXF] bit. It is incremented on each write access to the QSPI_TBDR register by the number of bytes pushed onto the TX Buffer. When it is not cleared the TRCTR field wraps around to 0. Refer to Section 30.4.3.17, TX Buffer Data Register (QSPI_TBDR), for details.
TRBFL	TX Buffer Fill Level. The TRBFL field contains the number of bytes available in the TX Buffer for the QuadSPI module to transmit to the serial flash device.

30.4.3.17 TX Buffer Data Register (QSPI_TBDR)

The QSPI_TBDR register provides access to the circular TX Buffer. This buffer provides the data written into it as write data for the page programming commands to the serial flash device. Refer to Table 30-45 for the byte ordering scheme.

Address: QSPI_BASE + 0x154

Write: QSPI_MCR[QMODE] = 1 and QSPI_SFMSR[TXFULL] = 0
32-bit write access required

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	TXDATA[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	TXDATA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-17. TX Buffer (QSPI_TBDR)

Table 30-30. QSPI_TBDR field descriptions

Field	Description
TXDATA	TX Data. On write access the data are written into the next available entry of the TX Buffer and the QPSI_TBSR[TRBFL] field is updated accordingly.

30.4.3.18 AMBA Control Register (QSPI_ACR)

The QSPI_ACR register defines the command that is used for following AHB commands. The read commands allowed are given in 30.7.1, Supported Instruction Codes in Winbond Devices. The execution of the command itself is triggered by an AHB read to the address range assigned to the memory-mapped serial flash data. For further details refer to Section 30.6.7, Command arbitration—SFM mode only.

Address: QSPI_BASE + 0x158 Write: Anytime

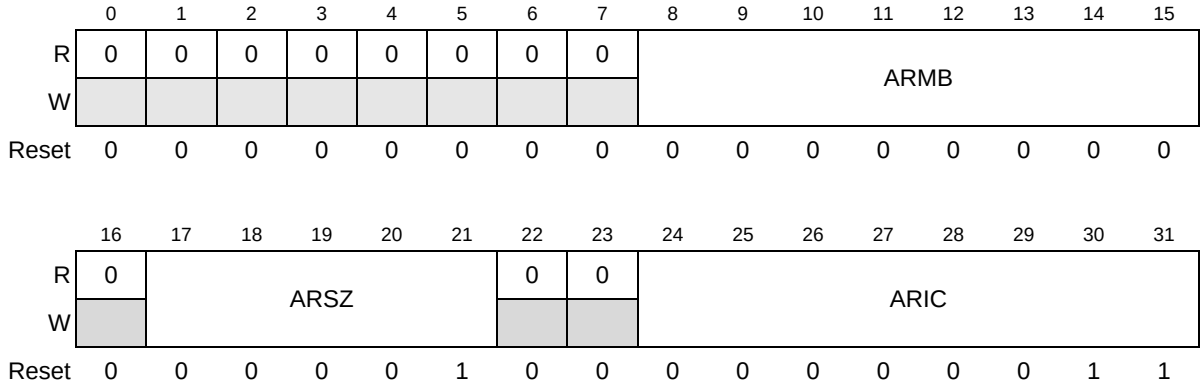


Figure 30-18. AMBA Control Register (QSPI_ACR)

Table 30-31. QSPI_ACR field descriptions

Field	Description
ARMB	AMBA Read Mode Byte. Instruction code option for ARIC for continuos mode (Table 30-53, M7-M0).
ARSZ	AMBA Read Size. Defines size of read burst to the external serial flash device for any starting address not found in the FIFO by a new AMBA AHB read access. The number of bytes read from the external serial flash device is the value in ARSZ * 4. Legal values for ARSZ are in the range from 1 to 16, resulting in 4 up to 64 bytes read from the external serial flash.
ARIC	AMBA Read Instruction Code. Selects the read command to be used for any read access to the external serial flash device. The reset value of the ARIC field is the READ_DATA command with a size of 4 bytes.

30.4.3.19 Serial Flash Mode Status Register (QSPI_SFMSR)

The QSPI_SFMSR register provides all available status information about SFM command execution and arbitration, the RX Buffer and TX Buffer and the AHB Buffer.

Address: QSPI_BASE + 0x15C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TXFULL	0	0	TXNE	0	0	0	0	RXFULL	0	0	RXNE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	AHBFULL	0	0	AHBNE	0	0	0	0	CONTMODE	AHB_ACC	IP_ACC	BUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-19. Serial Flash Mode Status Register (QSPI_SFMSR)

Table 30-32. QSPI_SFMSR field descriptions

Field	Description
TX Buffer Related Status Information	
TXFULL	TX Buffer Full: Asserted when no more data can be stored.
TXNE	TX Buffer Not Empty: Asserted when TX Buffer contains data.
RX Buffer Related Status Information	
RXFULL	RX Buffer Full: Asserted when RX Buffer is full.
RXNE	RX Buffer Not Empty: Asserted when RX Buffer contains data.
AHB Buffer Related Status Information	
AHBFULL	AHB Buffer Full: Asserted when AHB buffer is full.
AHBNE	AHB Buffer Not Empty: Asserted when AHB buffer contains data.
SFM Command Related Status Information	
CONTMODE	Continuous Mode: Asserted when last command was of type “Continuous Read Mode”[1]. It is asserted at the end of the execution of the related command together with the QSPI_SFMFR[TFF] flag.
AHB_ACC	AHB Access: Asserted when the transaction currently executed was initiated by AHB bus.
IP_ACC	IP Access: Asserted when transaction currently executed was initiated by IP bus.
BUSY	Module Busy: Asserted when module is currently busy handling a transaction to an external flash device.

[1] There are special, vendor-dependent commands that use a special access mode of the attached serial flash device. Once this mode is enabled in the serial flash device there are only certain SFM commands allowed until this mode is explicitly disabled. Refer to the serial flash device specification for further details.
Note that a mode bit collision is detected if other SFM commands are sent than those compatible with the special access mode.

30.4.3.20 Serial Flash Mode Flag Register (QSPI_SFMFR)

The QSPI_SFMFR register provides all available flags about SFM command execution and arbitration which may serve as source for the generation of interrupt service requests. Note that the error flags in this register do not relate directly to the execution of the transaction in the serial flash device itself but only to the behavior and conditions visible in the QuadSPI module.

Address: QSPI_BASE + 0x160 Write: Enabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R[1]	0	0	0	0	TBFF	TBUF	0	0	0	0	0	0	0	0	RBOF	RBDF
W					w1c	w1c									w1c	w1c
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R[1]	0	0	0	0	0	0	ABOF	0	IPAEF	IPIEF	ICEF	0	0	0	0	TFF
W							w1c		w1c	w1c	w1c					w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-20. Serial Flash Mode Flag Register (QSPI_SFMFR)

[1]When not in SFM mode all 0's are read

Table 30-33. QSPI_SFMFR field descriptions

Field	Description
TX Buffer Related Flags	
TBFF	TX Buffer Fill Flag: Set when the TX Buffer is not full. Refer to Section 30.5.3.6, TX Buffer Operation, for details.
TBUF	TX Buffer Underrun Flag: Set when FSM QSPI_IF tried to pull data although TX Buffer was empty. The command leading to the TX Buffer underrun is terminated without further write to the serial flash. The application must clear the TX Buffer in response to this event by writing a 1 into the QSPI_MCR[CLR_TXF] bit.
RX Buffer Related Flags	
RBOF	RX Buffer Overflow Flag: Set when not all the data read from the serial flash device could be pushed into the RX Buffer. If the command leading to the RX Buffer Overflow is still running it is terminated without further reads from the serial flash. The content of the RX Buffer is not changed.

Table 30-33. QSPI_SFMFR field descriptions *(continued)*

Field	Description
RBDF	RX Buffer Drain Flag: Will be set if the QSPI_SFMSR[RXNE] status bit is asserted, i.e., the RX Buffer is not empty. Refer to Section 30.5.3.5.2, Receive Buffer Drain Interrupt or DMA Request, for details. Writing a 1 into that bit triggers a RX Buffer POP event. If the RX Buffer is empty writing 1 into that bit clears the flag.
AHB Buffer Related Flags	
ABOF	AHB Buffer Overflow Flag: Set when state machine QSPI_IF tried to push data although AHB buffer was full.
SFM Command Related Flags	
IPAEF	IP Command Trigger during AHB Access Error Flag. Set when the following condition occurs: • A write access occurs to the QSPI_ICR register and the QSPI_SFMFR[AHB_ACC] bit is set. Any command leading to the assertion of the IPAEF flag is ignored
IPIEF	IP Command Trigger during IP Access Error Flag. Set on any of the following conditions: • A write access occurs to the QSPI_ICR register and the QSPI_SFMFR[IP_ACC] bit is set. Any command leading to the assertion of the IPIEF flag is ignored • A write access occurs to the QSPI_SFAR register and the QSPI_SFMFR[IP_ACC] bit is set.
ICEF	Instruction Code Error Flag: Set when a wrong instruction code is encountered or in case of mode bit collision[1]. The instruction which led to the assertion of the ICEF flag is not executed. Wrong commands issued via the AHB interface are not detected.
TFF	Transaction Finished Flag: Set when the QuadSPI module has finished a transaction on the external flash device. If an error occurred or the transaction was terminated the related error flags are valid at the latest in the same clock cycle when the TFF flag is asserted.

[1]Refer to Section 30.4.3.19, Serial Flash Mode Status Register (QSPI_SFMSR), SFMSR[CONTMODE] for the description of a mode bit collision

Additional details about the error flags contained in that register can be found in Table 30-50.

30.4.3.21 SFM Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER)

The QSPI_SFMRSER register provides enables and selectors for all interrupts in serial flash mode.

Address: QSPI_BASE + 0x164 Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TBFIE	TBUIE	0	0	0	0	RBDDE	0	0	0	RBOIE	RBDIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ABOIE	0	IPAEIE	IPIEIE	ICEIE	0	0	0	0	TFIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-21. SFM Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER)

Table 30-34. QSPI_SFMRSER field descriptions

Field	Description
TBFIE	TX Buffer Fill Interrupt Enable
TBUIE	TX Buffer Underrun Interrupt Enable
RBDDE	RX Buffer Drain DMA Enable: Enables generation of DMA requests for RX Buffer Drain. When this bit is set DMA requests via the ipd_req_rfdf line are generated as long as the QSPI_SFMSR[RXNE] status bit is set. 0 No DMA request will be generated 1 DMA request will be generated
RBOIE	RX Buffer Overflow Interrupt Enable
RBDIE	RX Buffer Drain Interrupt Enable: Enables generation of IRQ requests for RX Buffer Drain. When this bit is set the ipi_int_rfdf line is asserted as long as the QSPI_SFMSR[RBDF] flag is set. 0 No RBDF interrupt will be generated 1 RBDF Interrupt will be generated
ABOIE	AHB Buffer Overflow Interrupt Enable
IPAEIE	IP Command Trigger during AHB Access Error Interrupt Enable
IPIEIE	IP Command Trigger during IP Access Error Interrupt Enable
ICEIE	Instruction Code Error Interrupt Enable
TFIE	Transaction Finished Interrupt Enable

30.4.4 AHB bus register memory map descriptions

This chapter contains definitions of registers in the AMBA address space.

30.4.4.1 AHB bus access considerations

It has to be noted that all logic in the QuadSPI module implementing the AHB Bus access is related to read the content of an external serial flash device. Therefore the following restrictions apply to the QuadSPI module with respect to accesses to the AHB bus:

- In the SPI modes all accesses to the AHB interface are served without errors and undefined values are returned on read.
- Any write access in SFM mode is answered with the ERROR condition according to the AMBA AHB Specification. No write occurs.
- AHB Bus access types fully supported are NONSEQ and BUSY. In fact access type BUSY is treated in the same way like NONSEQ.
- AHB access type SEQ is treated in the same way like NONSEQ. Refer to the AMBA AHB Specification for further details.

30.4.4.2 Memory-mapped serial flash data (QSPI_SFD)

Starting with address QSPI_AMBA_BASE the content of the external serial flash is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address QSPI_AMBA_BASE with increasing order. Refer to Table 30-35 below for details.

Table 30-35. Memory-mapped serial flash address scheme

Memory-mapped address[1]	Serial flash byte address			
QSPI_AMBA_BASE +0x000_0000	0x000_0000	0x000_0001	0x000_0002	0x000_0003
QSPI_AMBA_BASE + 0x000_0004	0x000_0004	0x000_0005	0x000_0006	0x000_0007
...	...	...	...	...
QSPI_AMBA_BASE + 0x7FF_FFF8	0x7FF_FFF8	0x7FF_FFF9	0x7FF_FFFA	07FF_FFFB

1 Access scheme is limited to 32 bit data width

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

Note that for serial flash devices of 128MB size the last 4 bytes are not accessible via the memory-mapped interface.

For details concerning the read process refer to Section 30.5.3.3, Flash Read.

30.4.4.3 AHB RX Data Buffer (QSPI_ARDB)

The AHB RX Data Buffer register is used to read the buffer content of the RX Buffer. Any read access is redirected to the RX Buffer register entry corresponding to the current value of the read pointer. The increment of the read pointer depends from the access scheme (DMA or flag-driven). Refer to Section 30.5.3.3.2, Host Read of the QuadSPI Module Internal Buffers, topic **RX Buffer, data read via register interface** and **AHB read** for the description of successive accesses to the RX Buffer content. Refer also to Section 30.5.3.4, Byte Ordering of Serial Flash Data, for the byte ordering scheme.

Address: QSPI_AMBA_BASE + 0x07FF FFFC

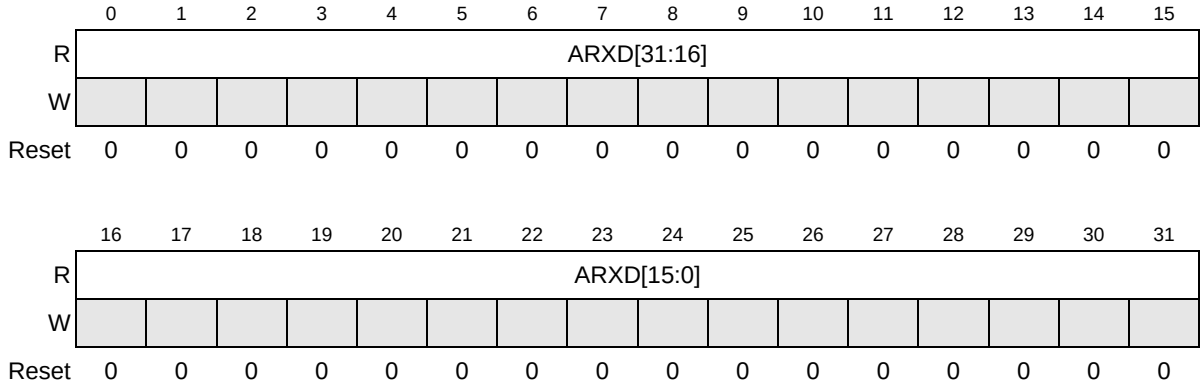


Figure 30-22. AHB RX Data Buffer (QSPI_ARDB)

Table 30-36. QSPI_ARDB field descriptions

Field	Description
ARXD	AMBA provided RX Buffer Data. Byte order (endianess) is identical to the RX Buffer Data Registers.

30.5 Functional description

The Quad Serial Peripheral Interface (QuadSPI) block can be used for two different types (mutually exclusive) of serial communications:

- It supports standard SPI full-duplex, synchronous serial communications with peripheral devices.
- It acts as an interface to external serial flash devices via up to 4 bidirectional data lines.

30.5.1 Modes of operation

Refer to Section 30.2.3, QuadSPI modes of operation for an overview over the possible operational modes of the QuadSPI block.

- SPI Master mode: In Master mode, the QuadSPI can initiate SPI communications with peripheral devices. See Section 30.5.2.2, Master mode, for more details.
- SPI Slave mode: In Slave mode, the QuadSPI responds to SPI transfers initiated by an external SPI master. SeeSection 30.5.2.3, Slave mode. for more details.
- Serial Flash mode can be used for write or read accesses to an external serial flash device.
  - Serial Flash Write: Data can be programmed into the flash of the serial flash device. Refer to Section 30.5.3.2, Flash Programming, for further details.
  - Serial Flash Read: Read the contents of the serial flash device. Two separate read channels are available via RX Buffer and AHB Buffer, see Section 30.5.3.3, Flash Read.
- Stop mode: The mode is used for power management. When a request is made to enter Stop mode, the QuadSPI block acknowledges the request and completes the transfer in progress, then the system clocks to the QuadSPI block may be shut off, see Section , Like all power saving features,

the Stop mode requires logic external to the QuadSPI module for power management and clock gating control. Stop mode.

- Module Disable mode: The mode is used for power management. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable mode.The module enters the mode by setting QSPI_MCR[MDIS]. See Section 30.5.4.1, Module Disable mode, for more details.

30.5.2 SPI (Serial Peripheral Interface) modes

In the SPI Master and SPI Slave modes, serial data are transferred using a shift register and a selection of programmable transfer attributes. The SPI frames can be from four to sixteen bits long. The data to be transmitted can come from queues stored in RAM external to the QuadSPI. Host software or a DMA Controller can transfer the SPI data from the queues to a First-In First-Out (FIFO) buffer. The received data is stored in entries in the RX FIFO. Host software or a DMA Controller transfer the received data from the RX FIFO to memory external to the QuadSPI. The FIFO buffer operations are described in Section 30.5.2.5, Transmit First In First Out (TX FIFO) Buffering Mechanism, and Section 30.5.2.6, Receive First In First Out (RX FIFO) Buffering Mechanism. The interrupt and DMA request conditions are described in Section 30.5.2.10, SPI mode interrupt and DMA requests.

There are two different SPI modes: Master mode and Slave mode. The FIFO operations are similar for the Master mode and Slave mode. The main difference is that in Master mode, the QuadSPI initiates and controls the transfer according to the fields in the SPI Command field of the TX FIFO entry. In Slave mode, the QuadSPI only responds to transfers initiated by a bus master external to the QuadSPI and the SPI Command field of the TX FIFO entry is ignored.

The 16-bit shift register in the Master and the 16-bit shift register in the Slave are linked by the SO and SI signals to form a distributed 32-bit register. When a data transfer operation is performed, data is serially shifted a pre-determined number of bit positions. Because the registers are linked, data is exchanged between the Master and the Slave; the data that was in the Master's shift register is now in the shift register of the Slave, and vice versa. At the end of a transfer, the TCF bit in the QSPI_SPISR is set to indicate a completed transfer. Figure 30-23 illustrates how Master and Slave data is exchanged.

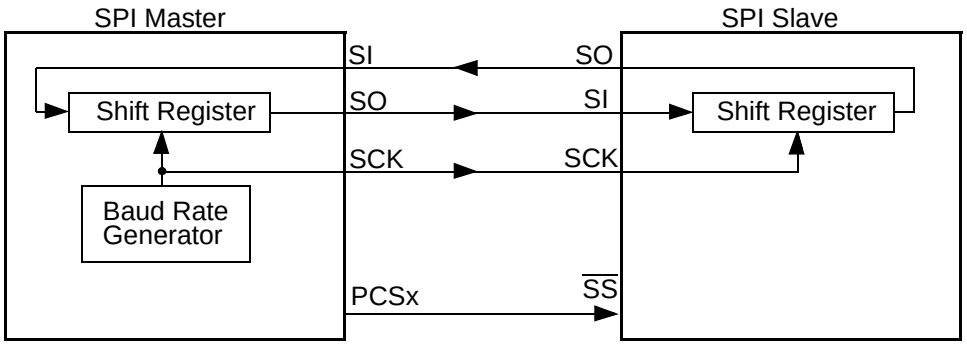


Figure 30-23. SPI Serial Protocol Overview

The QuadSPI has eight Peripheral Chip Select (PCS) signals that are used to select which of the Slaves to communicate with.

Both SPI modes and the SFM mode share transfer protocol and timing properties so they are described independently of the mode in Section 30.5.2.8, SPI Transfer Formats. The transfer rate and delay settings are described in Section 30.5.2.7, Baud Rate and Clock Delay Generation.

The QSPI_CTAR0–QSPI_CTAR7 registers hold clock and transfer attributes. In the SPI Master mode the user can select which CTAR to use on a frame by frame basis by setting a field in the SPI Command. See Section 30.4.3.4, Clock and Transfer Attributes Registers 0 – 1 (QSPI_CTAR0 – QSPI_CTAR1), for information on the fields of the QSPI_CTAR registers.

See Section 30.5.4, Power saving features, for information on the power-saving features of the QuadSPI.

30.5.2.1 Start and Stop of SPI Transfers

The QuadSPI in SPI Master or Slave mode has two operating states; Stopped and Running. The default state is Stopped. In the Stopped state no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the QuadSPI without causing undetermined results. The TXRXS bit in the QSPI_SPISR is negated in this state. In the Running state serial transfers take place. The TXRXS bit in the QSPI_SPISR is asserted in the Running state. Figure 30-24 shows a state diagram of the start and stop mechanism. The transitions are described in Table 30-37.

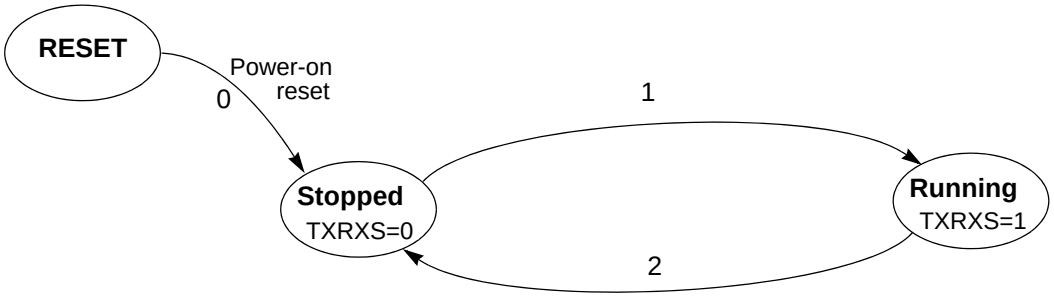


Figure 30-24. QuadSPI Start and Stop State Diagram

Table 30-37. State Transitions for Start and Stop of QuadSPI Transfers

Transition #	Current State	Next State	Description
0	RESET	Stopped	Generic power-on reset transition
1	Stopped	Running	The QuadSPI is started (SPI state transitions to Running) when all of the following conditions are true: • EOQF bit is clear • HALT bit is clear • ipg_debug is clear *OR* FRZ is clear
2	Running	Stopped	The QuadSPI stops (transitions from Running to Stopped) after the current frame for any one of the following conditions: • EOQF bit is set • HALT bit is set • ipg_debug is set *AND* FRZ is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or on the next system clock cycle if no transfers are in progress.

30.5.2.2 Master mode

In SPI Master mode (MSTR bit in the QSPI_MCR is set) the QuadSPI operates as bus master and initiates the serial transfers by controlling the Serial Communications Clock (SCK) and the Peripheral Chip Select (PCS) signals. The SPI command field CTAS in the executing TX FIFO entry determines which CTAR register will be used to set the transfer attributes and which PCS signal to assert. The command fields also contains various bits that help with queue management and transfer protocol. See Section 30.4.3.7, PUSH TX FIFO Register (QSPI_PUSHR). for details on the SPI command fields. The data field in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SO) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it allowing for transfer attribute control on a frame by frame basis.

30.5.2.3 Slave mode

In SPI Slave mode (MSTR bit in the QSPI_MCR register is not set) the QuadSPI responds to transfers initiated by an external SPI bus master and cannot initiate transfers. The QuadSPI slave is selected by a bus master by having the slave's $\overline{SS}$ asserted. In Slave mode the SCK is provided by the bus master. All transfer attributes are controlled by the bus master but clock polarity, clock phase and numbers of bits to transfer must still be configured in the QuadSPI slave for proper communications with an external SPI master. In SPI Slave mode, the slave transfer attributes are set in the QSPI_CTAR0. The QuadSPI transfers the MSB first. The LSBFE field of the associated CTAR is ignored.

30.5.2.4 FIFO Disable Operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The QuadSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The TX and RX FIFOs are disabled separately. The TX FIFO is disabled by writing a 1 to the DIS_TXF bit in the QSPI_MCR. The RX FIFO is disabled by writing a 1 to the DIS_RXF bit in the QSPI_MCR.

The FIFO Disable mechanisms are transparent to the user and to host software; Transmit data and commands are written to the QSPI_PUSHR and received data is read from the QSPI_POPR.

When the TX FIFO is disabled the TFFF, TFUF and TXCTR fields in QSPI_SPISR behave as if there is a one-entry FIFO but the contents of the QSPI_TXFR registers and TXNXTPTR are undefined. When the RX FIFO is disabled the RFDF, RFOF and RXCTR fields in the QSPI_SPISR behave as if there is a one-entry FIFO but the contents of the QSPI_RXFR registers and POPNXTPTR are undefined.

30.5.2.5 Transmit First In First Out (TX FIFO) Buffering Mechanism

The TX FIFO functions as a buffer of SPI data and SPI Commands for transmission. The TX FIFO holds in total 15 entries, each consisting of a command field and a data field. SPI Commands and data are added to the TX FIFO by writing to the PUSH TX FIFO Register (QSPI_PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the SPI Status Register (QSPI_SPISR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time the QuadSPI _PUSHR is written or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. The TXNXTPTR contains the positive offset from QSPI_TXFR0 in number of 32-bit registers. For example, TXNXTPTR equal to two means that the QSPI_TXFR2 contains the SPI data and command for the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register.

30.5.2.5.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO by writing to the QSPI_PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the QSPI_SPISR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to QSPI_PUSHR is complete or by host software writing a 1 to the TFFF in the QSPI_SPISR. The TFFF can generate a DMA request or an interrupt request. See Section 30.5.3.5.1, Transmit buffer fill interrupt request, for details.

The QuadSPI ignores attempts to push data to a full TX FIFO, i.e., the state of the TX FIFO is unchanged. No error condition is indicated.

30.5.2.5.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter is decremented by one. At the end of a transfer, the TCF bit in the QSPI_SPISR is set to indicate the completion of a transfer. The TX FIFO is cleared by writing a 1 to the CLR_TXF bit in QSPI_MCR.

If an external bus master initiates a transfer with a QuadSPI slave while the slave's QuadSPI TX FIFO is empty, the Transmit FIFO Underrun Flag (TFUF) in the slave's QSPI_SPISR is set. See Section Section 30.5.2.10.4, Transmit FIFO Underrun Interrupt Request, for details.

30.5.2.6 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SI pin. The RX FIFO holds in total 15 received SPI data frames. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the POP RX FIFO Register (QSPI_POPR). RX FIFO entries can only be removed from the RX FIFO by reading the QSPI_POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the SPI Status Register (QSPI_SPISR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the QSPI_POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the QSPI_SPISR points to the RX FIFO entry that is returned when the QSPI_POPR is read. The POPNXTPTR contains the positive offset from QSPI_RXFR0 in number of 32-bit registers. For example, POPNXTPTR equal to two means that the QSPI_RXFR2 contains the

received SPI data that will be returned when QSPI_POPR is read. The POPNXTPTR field is incremented every time the QSPI_POPR is read.

30.5.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time a SPI frame is transferred to the RX FIFO the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the QSPI_SPISR is asserted indicating an overflow condition. Depending on the state of the ROOE bit in the QSPI_MCR, the data from the transfer that generated the overflow is either ignored or shifted into the shift register. If the ROOE bit is asserted, the incoming data is shifted into the shift register. If the ROOE bit is negated, the incoming data is ignored.

30.5.2.6.2 Draining the RX FIFO

Host software or other intelligent blocks can remove (pop) entries from the RX FIFO by reading the POP RX FIFO Register (QSPI_POPR). A read of the QSPI_POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored, the RX FIFO Counter remains unchanged. The data returned from reading an empty RX FIFO is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the QSPI_SPISR is set. The RFDF bit is cleared when the RX FIFO is empty and the DMA controller indicates that a read from QSPI_POPR is complete or by host software writing a 1 to the RFDF.

30.5.2.7 Baud Rate and Clock Delay Generation

The SCK frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option for doubling the baud rate. Figure 30-25 shows conceptually how the SCK signal is generated.

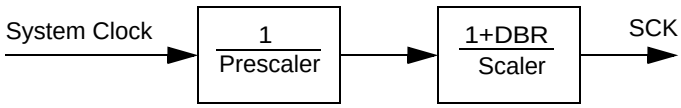


Figure 30-25. Communications Clock Prescalers and Scalers

30.5.2.7.1 Baud Rate Generator

The Baud Rate is the frequency of the Serial Communication Clock (SCK). The system clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR and BR fields in the QSPI_CTAR registers select the frequency of SCK by the formula in the BR[0:3] field description. Table 30-38 shows an example of how to compute the baud rate.

Table 30-38. Baud Rate Computation Example

PBR	Prescaler	BR	Scaler	DBR	Fsys	Baud Rate
0b00	2	0b0000	2	0	100 MHz	25 Mb/s
0b00	2	0b0000	2	1	20 MHz	10 Mb/s

30.5.2.7.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 30-27 and Figure 30-28 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the QSPI_CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK[0:3] bit description. Table 30-39 shows an example of how to compute the PCS to SCK delay.

Table 30-39. PCS to SCK Delay Computation Example

PCSSCK	Prescaler	CSSCK	Scaler	Fsys	PCS to SCK Delay
0b01	3	0b0100	32	100 MHz	0.96 µs

30.5.2.7.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 30-35 for illustrations of the After SCK delay. The PASC and ASC fields in the QSPI_CTAR*x* registers select the After SCK Delay by the formula in the ASC[0:3] field description. Table 30-40 shows an example of how to compute the After SCK delay.

Table 30-40. After SCK Delay Computation Example

PASC	Prescaler	ASC	Scaler	Fsys	After SCK Delay
0b01	3	0b0100	32	100 MHz	0.96 µs

30.5.2.7.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the length of time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 30-27 for an illustration of the Delay after Transfer. The PDT and DT fields in the QSPI_CTAR*x* registers select the Delay after Transfer by the formula in the DT[0:3] field description. Table 30-41 shows an example of how to compute the Delay after Transfer.

Table 30-41. Delay after Transfer Computation Example

PDT	Prescaler	DT	Scaler	Fsys	Delay after Transfer
0b01	3	0b1110	32768	100 MHz	0.98 ms

When in non-continuous clock mode the T_{DT} delay can be configured as outlined in the QSPI_CTAR*x* registers. When in continuous clock mode the delay is fixed at 1 SCK period.

30.5.2.7.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the QuadSPI is in Master mode and PCSSE bit is set in the QSPI_MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode the PCS[4:0] and PCS[7:6] signals into as many as 128 glitch-free PCS signals. Figure 30-26 shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

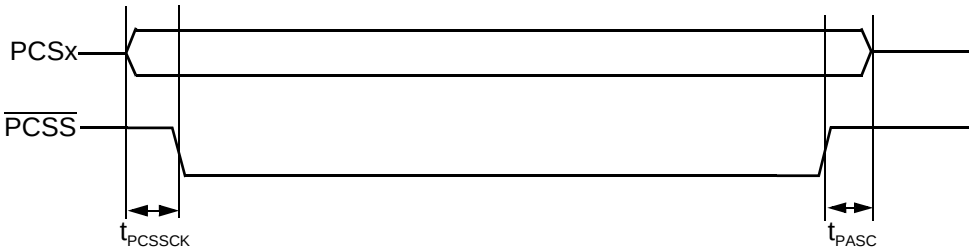


Figure 30-26. Peripheral Chip Select Strobe Timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the QSPI_CTAR based on the following formula:

$$t_{PCSSCK} = \frac{1}{f_{SYS}} \times PCSSCK \qquad \textit{Eqn. 30-5}$$

At the end of the transfer the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the QSPI_CTAR based on the following formula:

$$t_{PASC} = \frac{1}{f_{SYS}} \times PASC \qquad \textit{Eqn. 30-6}$$

Table 30-42 shows an example of how to compute the t_{pcssck} delay.

Table 30-42. Peripheral Chip Select Strobe Assert Computation Example

PCSSCK	Prescaler	Fsys	Delay before Transfer
0b11	7	100 MHz	70.0 ns

Table 30-43 shows an example of how to compute the t_{pasc} delay.

Table 30-43. Peripheral Chip Select Strobe Negate Computation Example

PASC	Prescaler	Fsys	Delay after Transfer
0b11	7	100 MHz	70.0 ns

The $\overline{\text{PCSS}}$ signal is not supported when Continuous Serial Communication SCK is enabled (CONT_SCKE=1).

30.5.2.8 SPI Transfer Formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the Master device synchronizes shifting and sampling of the data on the SI and SO pins. The PCS signals serve as enable signals for the slave devices.

When the QuadSPI is the bus master, the CPOL and CPHA bits in the QuadSPI Clock and Transfer Attributes Registers (QSPI_CTAR*x*) select the polarity and phase of the serial clock, SCK. The polarity bit selects the idle state of the SCK. The clock phase bit selects if the data on SO is valid before or on the first SCK edge.

When the QuadSPI is the bus Slave, CPOL and CPHA bits in the QSPI_CTAR0 select the polarity and phase of the serial clock. For SPI Slaves the QSPI_CTAR0 is used, and for DSI Slaves the QSPI_CTAR1 is used. Even though the bus Slave does not control the SCK signal, clock polarity, clock phase and number of bits to transfer must be identical for the master device and the slave device to ensure proper transmission.

The QuadSPI supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer format with CPHA = 0
- Modified Transfer format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The QuadSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the QSPI_MCR selects between Classic SPI Format and Modified Transfer Format. The Classic SPI Formats are described in Section 30.5.2.8.1, Classic SPI Transfer Format (CPHA = 0), and Section 30.5.2.8.2, Classic SPI Transfer Format (CPHA = 1). The Modified Transfer Formats are described in Section 30.5.2.8.3, Modified SPI Transfer Format (MTFE = 1, CPHA = 0), and Section 30.5.2.8.4, Modified SPI Transfer Format (MTFE = 1, CPHA = 1).

In SPI Master mode and SPI Slave mode the QuadSPI provides the option of keeping the PCS signals asserted between frames. See Section 30.5.2.8.5, Continuous Selection Format, for details.

30.5.2.8.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in Figure 30-27 is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SI pins on the odd-numbered SCK edges and change the data on their SO pins on the even-numbered SCK edges.

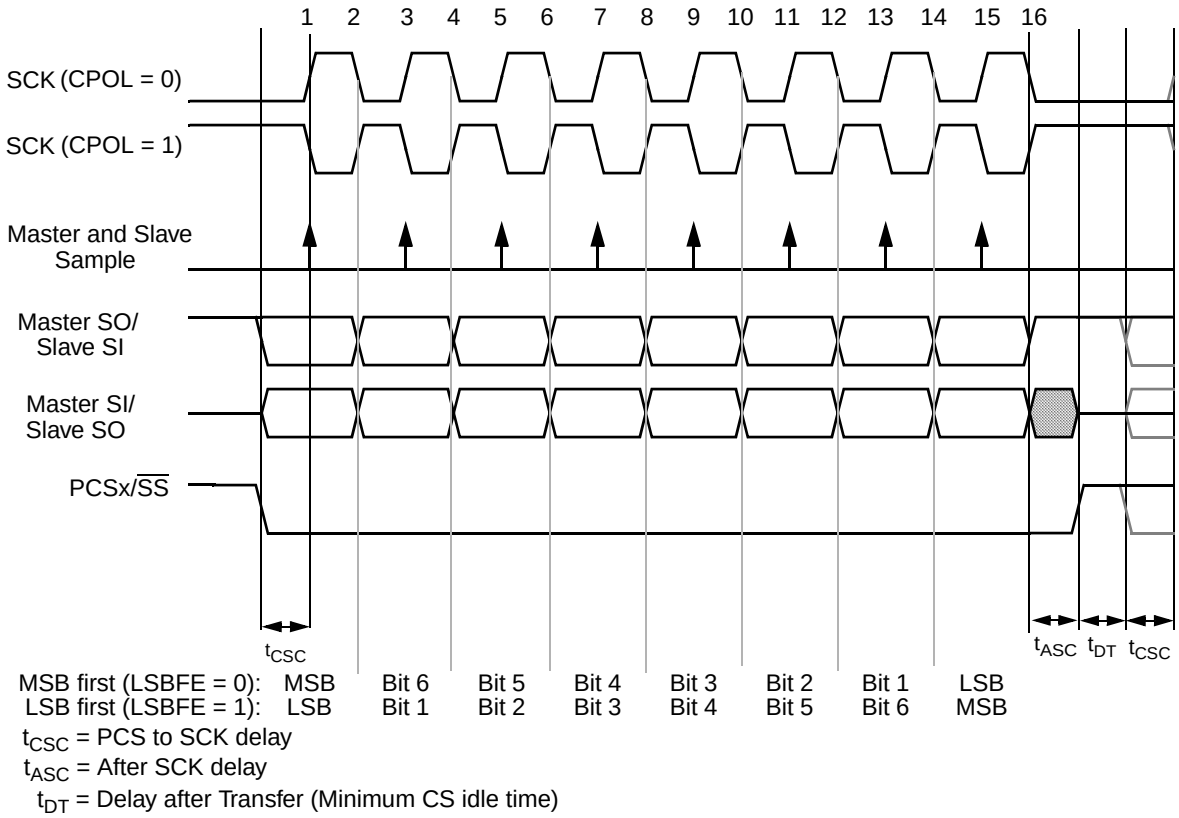


Figure 30-27. QuadSPI Transfer Timing Diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SO pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SO pin. After the t_{CSC} delay has elapsed, the master outputs the first edge of SCK. This is the edge used by the master and slave devices to sample the first input data bit on their serial data input signals. At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SI pins on the odd-numbered clock edges and changes the data on their SO pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

30.5.2.8.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in Figure 30-28 is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SO pin. In this format the master and slave devices change the data on their SO pins on the odd-numbered SCK edges and sample the data on their SI pins on the even-numbered SCK edges

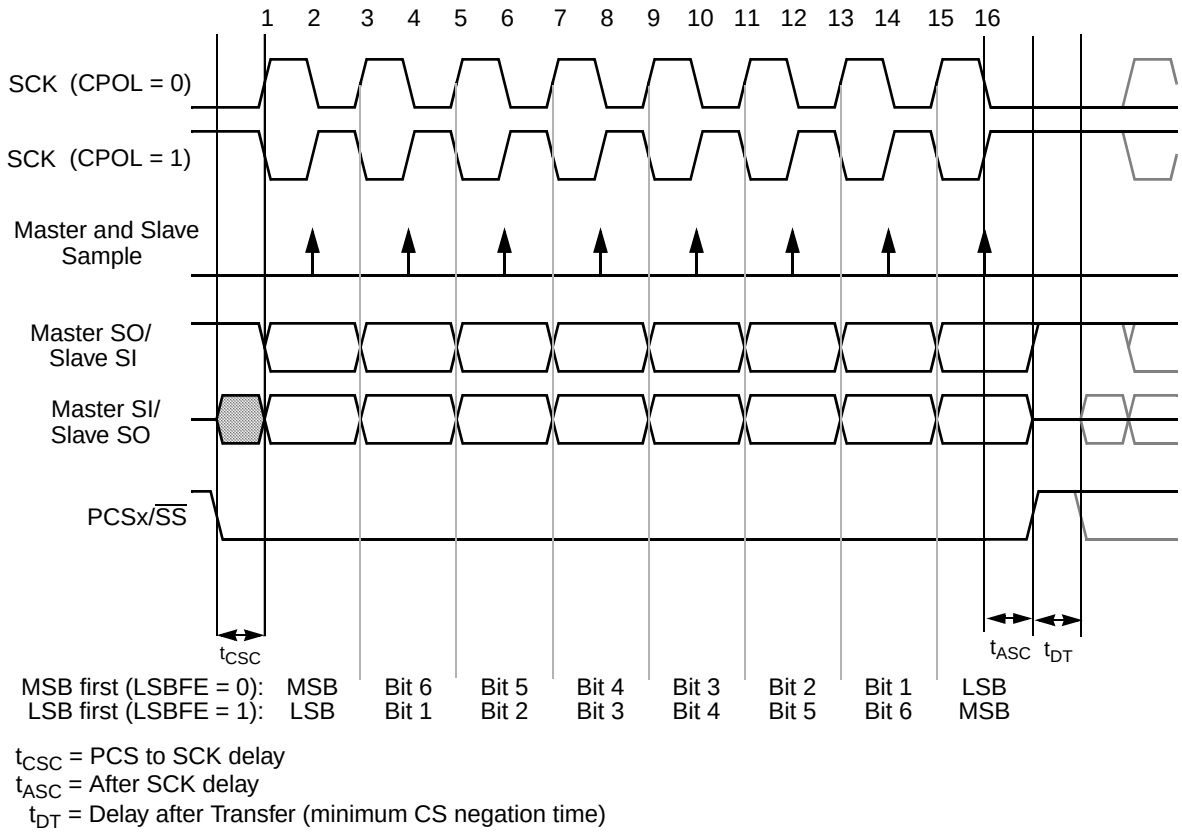


Figure 30-28. QuadSPI Transfer Timing Diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SO pin. The slave responds to the first SCK edge by placing its first data bit on its slave SO pin.

At the second edge of the SCK the master and slave sample their SI pins. For the rest of the frame the master and the slave change the data on their SO pins on the odd-numbered clock edges and sample their SI pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

30.5.2.8.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the Master and the Slave sample later in the SCK period than in the classical modes to allow for delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The Master and the Slave places data on the SO pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The Slave samples the Master SO signal on every odd numbered SCK edge. The Slave also places new data on the Slave SO on every odd numbered clock edge.

The Master places its second data bit on the SO line one system clock after odd numbered SCK edge. The point where the Master samples the Slave SO is selected by writing to the SMPL_PT field in the QSPI_MCR. The SMPL_PT field description in Table 30-9 lists the number of system clock cycles between the active edge of SCK and the Master Sample point. The Master sample point can be delayed by one or two system clock cycles.

Figure 30-29 shows the modified transfer format for CPHA = 0. Only the condition where CPOL = 0 is illustrated. The delayed Master sample points are indicated with a lighter shaded arrow.

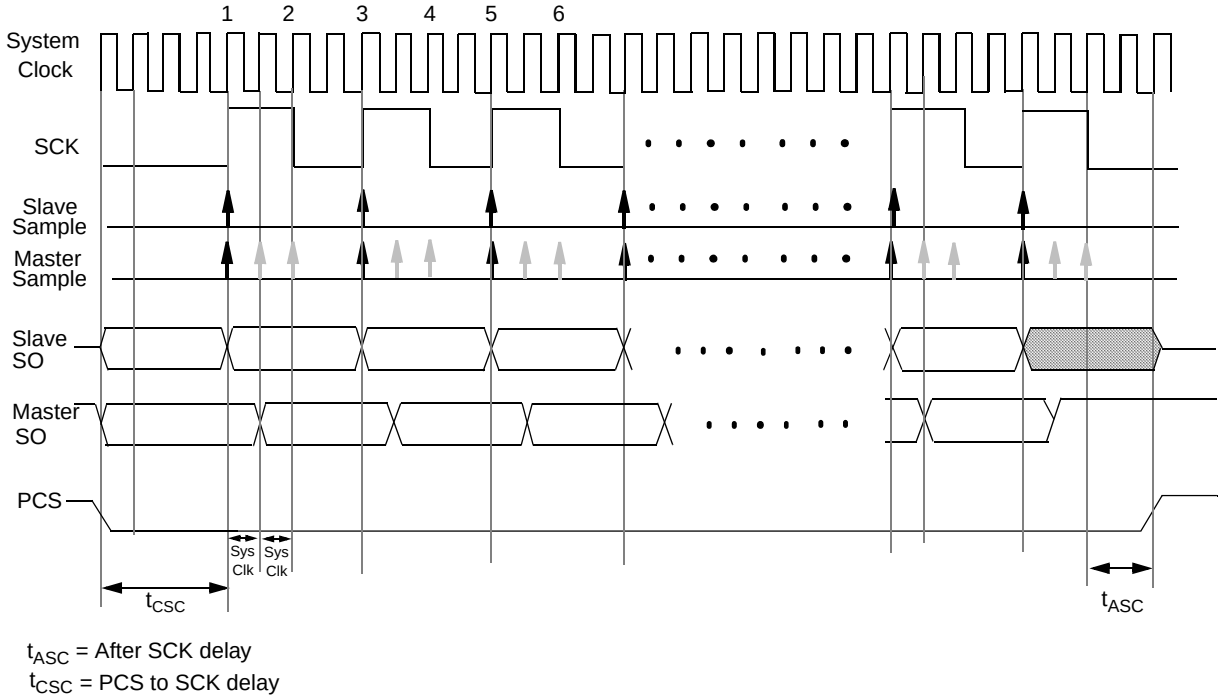
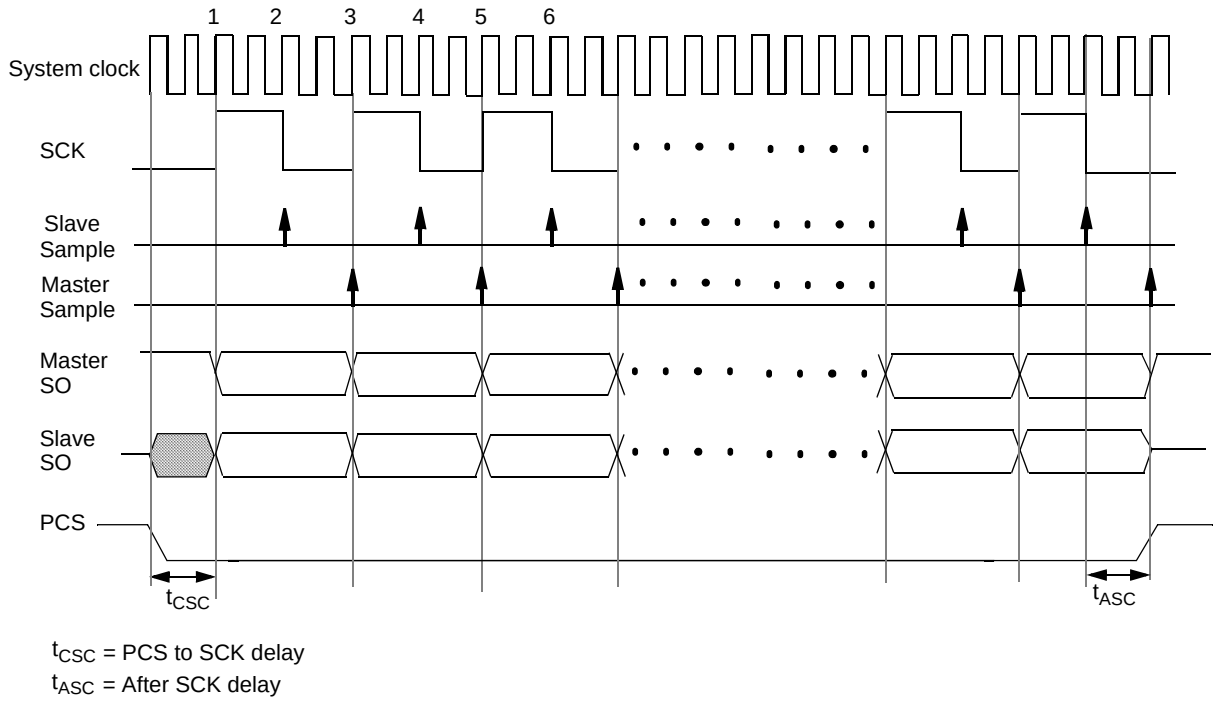


Figure 30-29. QuadSPI Modified Transfer Format (MTFE=1, CPHA=0, Fsck = Fsys/4)

30.5.2.8.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

Figure 30-30 shows the Modified Transfer Format for CPHA = 1. Only the condition where CPOL = 0 is described. At the start of a transfer the QuadSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SO pins at the first edge of SCK. The Slave samples the Master SO signal on the even numbered edges of SCK. The Master samples the Slave SO signal on the odd numbered SCK edges starting with the third SCK edge. The Slave samples the last bit on the last edge of the SCK. The Master samples the last Slave SO bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the Master SCK pin during the sampling of the last bit. The SCK to PCS delay must be greater or equal to half of the SCK period.



t_{CSC} = PCS to SCK delay
t_{ASC} = After SCK delay

Figure 30-30. QuadSPI Modified Transfer Format (MTFE=1, CPHA=1, Fsck = Fsys/4)

30.5.2.8.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both cases. The Continuous Selection Format is enabled for both SPI modes by setting the CONT bit in the SPI Command.

When the CONT bit = 0, the QuadSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSIS field in the QSPI_MCR. Figure 30-31 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 0.

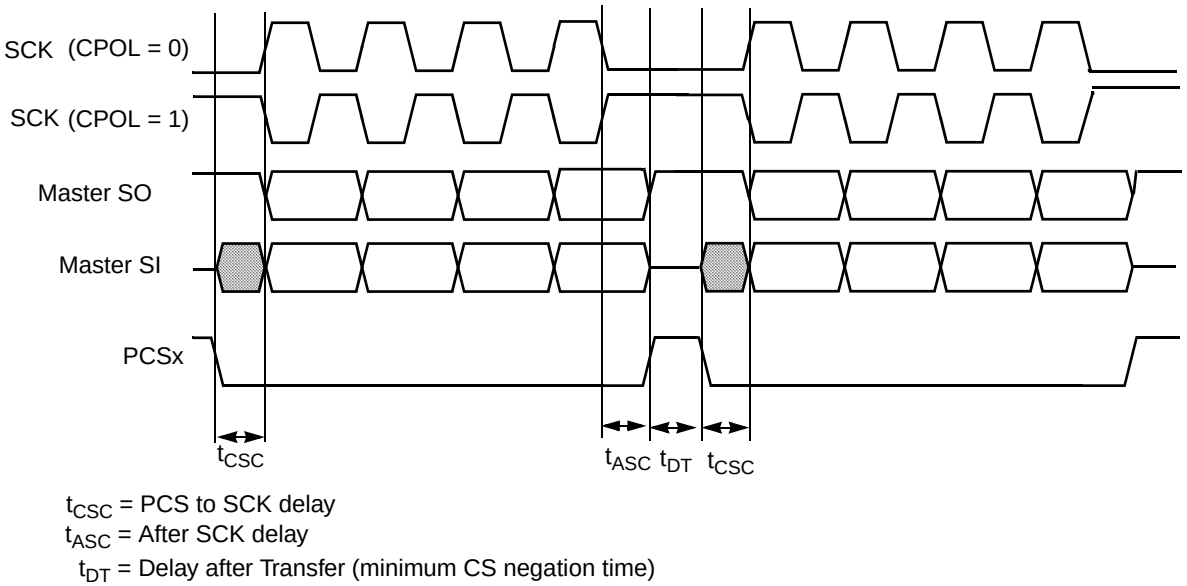


Figure 30-31. Example of Non-Continuous Format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. Figure 30-32 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

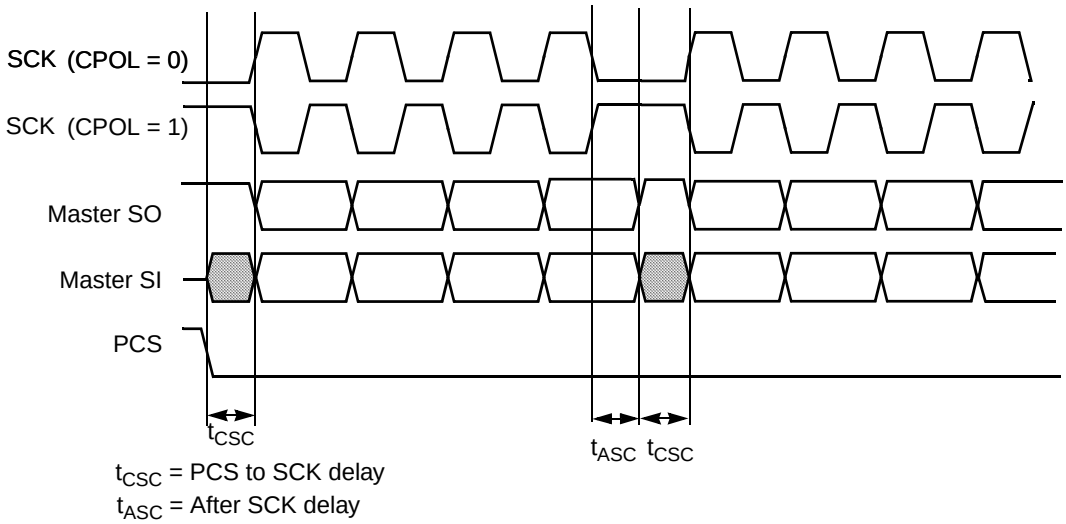


Figure 30-32. Example of Continuous Transfer (CPHA=1, CONT=1)

Switching CTAR registers or changing which PCS signals are asserted between frames while using Continuous Selection can cause errors in the transfer. The PCS signal should be negated before CTAR is switched or different PCS signals are selected.

30.5.2.9 Continuous Serial Communications Clock

The QuadSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the QSPI_MCR. Continuous SCK is valid in both SPI modes.

Continuous SCK is only supported for CPHA=1. Setting CPHA=0 will be ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- In both SPI modes CTAR0 shall be used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame shall be used.
- In both SPI modes the currently selected CTAR shall remain in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended that the baud rate is the same for all transfers made while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the QuadSPI is put into the Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (T_{DT}) is fixed at one T_{SCK} cycle. Figure 30-33 shows timing diagram for Continuous SCK format with Continuous Selection disabled.

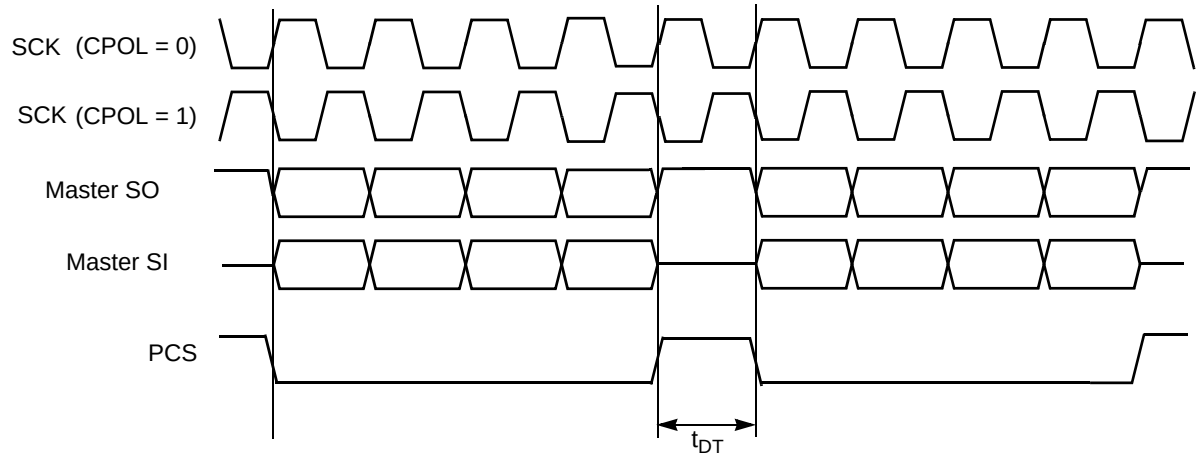


Figure 30-33. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SO (SO pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering Stopped state (refer to Section 30.5.2.1, Start and Stop of SPI Transfers).

- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 30-34 shows timing diagram for Continuous SCK format with Continuous Selection enabled.

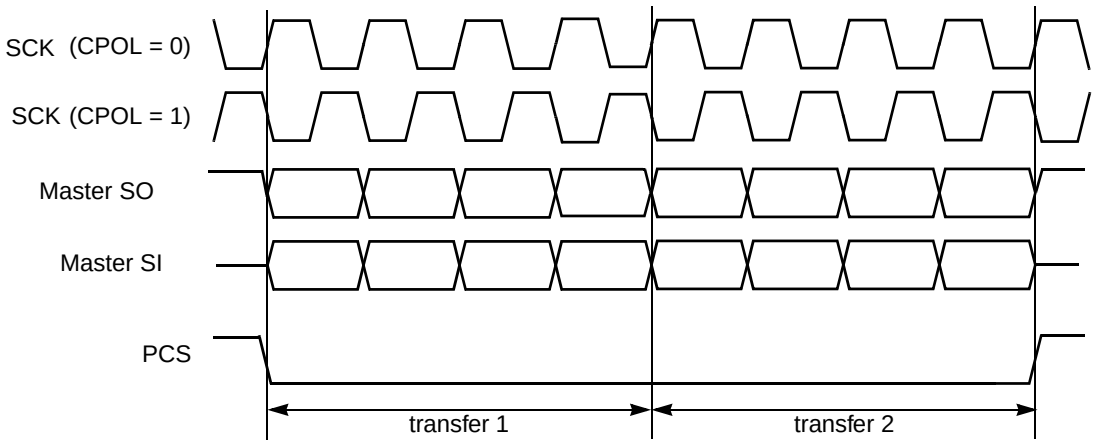


Figure 30-34. Continuous SCK timing diagram (CONT=1)

30.5.2.10 SPI mode interrupt and DMA requests

In both SPI modes the QuadSPI has four conditions that can only generate interrupt requests and two conditions that can generate interrupt or DMA request alternatively. Table 30-44 lists the six conditions. Note that the flags mentioned in the table relate to the SPI status register QSPI_SPISR

Table 30-44. SPI mode interrupt and DMA request conditions

Condition	Flag (QSPI_SPISR)	Interrupt	DMA
End of Queue (EOQ)	EOQF	X	
TX FIFO Fill	TFFF	X	X
Transfer Complete	TCF	X	
TX FIFO Underrun	TFUF	X	
RX FIFO Drain	RFDF	X	X
RX FIFO Overflow	RFOF	X	

Each condition has a flag bit in the SPI Status Register (QSPI_SPISR) and a Request Enable bit in the SPI Interrupt and DMA Request Select and Enable Register (QSPI_SPIRSER). The TX FIFO Fill Flag (TFFF) and RX FIFO Drain Flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the QSPI_SPIRSER.

30.5.2.10.1 End of Queue Interrupt Request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI Command is asserted and the EOQF_RE bit in the QSPI_SPIRSER is asserted.

30.5.2.10.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the QSPI_SPIRSER is asserted. The TFFF_DIRS bit in the QSPI_SPIRSER selects whether a DMA request or an interrupt request is generated.

30.5.2.10.3 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the QSPI_SPIRSER.

30.5.2.10.4 Transmit FIFO Underrun Interrupt Request

The Transmit FIFO Underrun Request indicates that an underrun condition in the TX FIFO has occurred. The transmit underrun condition is detected only in SPI Slave mode. The TFUF bit is set when the TX FIFO is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the QSPI_SPIRSER is asserted, an interrupt request is generated.

30.5.2.10.5 RX FIFO Drain Interrupt or DMA Request

The RX FIFO Drain Request indicates that the RX FIFO is not empty. The RX FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the QSPI_SPIRSER is asserted. The RFDF_DIRS bit in the QSPI_SPIRSER selects whether a DMA request or an interrupt request is generated.

30.5.2.10.6 RX FIFO Overflow Interrupt Request

The RX FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A RX FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the QSPI_SPIRSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the QSPI_MCR, the data from the transfer that generated the overflow is either ignored or shifted into the shift register. If the ROOE bit is set, the incoming data is shifted into the shift register. If the ROOE bit is negated, the incoming data is ignored.

30.5.3 SFM (Serial Flash) mode

This mode is used to allow communication with an external serial flash device. Compared to the standard SPI protocol, this communication method uses up to 4 bidirectional data lines operating at high data rates. The communication to the external serial flash device consists of an instruction code and optional address, mode, dummy and data transfers. All operations to the external serial flash device may use only instruction codes listed in Section 30.7, Serial Flash Devices.

Note that all the information given in this paragraph is applicable only if the QSPI_MCR[QMODE] bit is set.

30.5.3.1 Issuing SFM Commands

Each access to the external device follows the same sequence:

1. The user must provide the required components of a SFM command to the QuadSPI module.
2. From these components the complete transaction is built. The transaction starts and the status bit QSPI_SFMSR[BUSY] is set.
3. Communication with the external serial flash device is started and the transaction is executed.
4. When the transaction is finished (all transmit- and receive operations with the external serial flash device are finished) or terminated (an error condition occurred during the transaction) the status bit QSPI_SFMSR[BUSY] is reset and the QSPI_SFMFR[TFF] flag is set.

Further details are given in below in Section 30.5.3.2, Flash Programming, and Section 30.5.3.3, Flash Read.

Note that there are 2 different ways to trigger the processing of SFM Commands in the QuadSPI module.

30.5.3.1.1 IP Commands

For IP Commands the required components need to be written into the following registers:

- Read address of the serial flash into QSPI_SFAR, refer to Section 30.4.3.11, Serial Flash Address Register (QSPI_SFAR).
- Instruction code options belonging to the IP Command into the QSPI_ICR[ICO] field.
- Instruction code belonging to the IP Command into the QSPI_ICR[IC] field.

  Note that the write into the QSPI_ICR[IC] field must be the last step of the sequence. It is possible to combine both fields of the QSPI_ICR into one single write. Refer to Section 30.4.3.12, Instruction Code Register (QSPI_ICR), for details.

Note that there are some conditions were no IP Command is executed after writing the QSPI_ICR[IC] field and the write operation itself is ignored. They are described in Section 30.6.7, Command arbitration—SFM mode only.

30.5.3.1.2 AHB Commands

Note that the required components of the AHB commands are located in different registers w.r.t. the IP Commands. They need to be written into the QSPI_ACR register like described in Section 30.4.3.18, AMBA Control Register (QSPI_ACR).

The AHB Command itself is triggered by a read access of the host into the memory-mapped serial flash data, like described in Section 30.4.4.2, Memory-mapped serial flash data (QSPI_SFD).

Again the possible error conditions are described in Section 30.6.7, Command arbitration—SFM mode only.

30.5.3.2 Flash Programming

In all cases the memory sector to be written needs to be erased first. The programming sequence itself is then initiated in the following way:

1. Check that the TX Buffer is empty. If the QSPI_SFMSR[TXNE] bit is set the TX Buffer must be cleared by writing 1 into the QSPI_MCR[CLR_TXF] bit.
2. Program the address related to the command in the QSPI_SFAR register. Optionally one can clear the QSPI_TBSR[TRCTR] field by writing 1 into QSPI_MCR[CLR_TXF].
3. Provide initial data for the program command into the circular buffer via register TX Buffer Data Register (QSPI_TBDR). At least one word of data must be written into the TX Buffer.
4. Program the required instruction code options (i.e. size of data) into the QSPI_ICR[ICO] register.
5. Trigger the IP Command to program the serial flash device by writing the instruction code into the QSPI_ICR[IC] register.
6. Depending from the amount of data required step 3 must be repeated until all the required data have been written into the QSPI_TBDR register. At any time the QSPI_TBDR[TRCTR] field can be read to check how many words have been written actually into the TX Buffer.

Steps 4 and 5 may be executed together.

Upon writing the QSPI_ICR[IC] field (refer to step 5) the QuadSPI module will start to execute the command by transferring instruction code, address and then data to the external device. The data are fetched from the TX Buffer. It consists of 15 entries with 32-bit and is organized as a circular FIFO, whose read pointer is incremented after each fetch. When all data are transmitted, the QuadSPI module will return from ‘busy’ to ‘idle’. However, this is not true for the external device since the internal programming is still ongoing. It is up to the user to monitor the relevant status information available from the serial flash device and to ensure that the programming is finished properly.

30.5.3.3 Flash Read

Host access to the data stored in the external serial flash device is done in two steps: First the data must be read into the internal buffers and in the second step these internal buffers can be read by the host.

30.5.3.3.1 Reading Serial Flash Data into the QuadSPI Module

Read access to the external serial flash device can be triggered in two different ways:

- **IP Command Read**: For **flash read via the register interface** the user must provide the required components of a SFM command to the QSPI_SFAR and the QSPI_ICR registers. All available read commands supported by the external serial flash are possible.

  Optionally it is possible to clear the RX Buffer pointer prior to triggering the IP Command by writing a 1 into the QSPI_MCR[CLR_RXF] bit.

  From these inputs the complete transaction is built when the QSPI_ICR[IC] field is written. The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash device into the RX Buffer. Since the read access is triggered via the register interface the IP_ACC status bit is set driving in turn the BUSY bit (both are located in the QSPI_SFMSR register).

The communication with the external serial flash is stopped when the specified number of bytes has been read (successful completion of the transaction) or if the RX Buffer overrun condition is detected (signaled by setting the QSPI_SFMFR[RBOF] flag). In this case the transaction leading to the RX Buffer overrun is terminated.

- **AHB Command Read**: For a **memory-mapped flash read** the user must setup a read access to the address range were the external serial flash device is mapped to by programming the QSPI_ACR register *with the requested data not already available in the AHB Buffer.*

  On each AHB read access to the memory-mapped area the valid data in the AHB Buffer are checked against the address requested in the actual read. When the AHB read request can't be served from the content of the AHB Buffer the complete transaction to access the external serial flash device is built from the QSPI_ACR register contents and started. The requested number of bytes defined in the QSPI_ACR[ARSZ] field is then fetched from the external serial flash device into the internal AHB Buffer. Since the read access is triggered via the AHB bus the AHB_ACC status bit is set driving in turn the BUSY bit (both are located in the QSPI_SFMSR register) until the transaction is finished. The communication with the external serial flash is stopped when the specified number of bytes has been read.

  Basically the AHB buffer behaves similar to a cache memory with a size of one single line.

30.5.3.3.2 Host Read of the QuadSPI Module Internal Buffers

The data read out from the external serial flash device by the QuadSPI module are stored in the internal buffers. Depending from the buffer to which the data from the external serial flash has been loaded there are several different ways to access these data in the internal buffers:

- **Flag-based Data Read of the 1RX Buffer** is done by reading the address QSPI_ARDB, see Section 30.4.4.3, AHB RX Data Buffer (QSPI_ARDB).

  The QSPI_SFMSR[RXNE] bit indicates that data are available in the RX Buffer and can be read by AHB read access to address QSPI_ARDB. The RX Buffer is implemented as circular buffer. This means that after the increment of the read pointer the next read accesses to that (same) address provides the next data word from the serial flash device. The increment of the read pointer itself is done by writing a 1 into the QSPI_SFMFR[RBDF] bit.

  For the remaining status related bits QSPI_SPISR[RFOF] and QSPI_SPISR[RFDF] refer to Section 30.4.3.5, SPI Status Register (QSPI_SPISR). It's up to the user to decide whether the relevant flags are polled by software or to drive the data read by the interrupt capabilities of the QuadSPI module.
- **DMA triggered Data Read of the RX Buffer** is done by reading address QSPI_ARDB, see Section 30.4.4.3, AHB RX Data Buffer (QSPI_ARDB), by using the DMA capabilities of the QuadSPI and the device containing the QuadSPI module. The circular buffer pointer is updated automatically without interaction from the application.

  Refer to reference Semiconductor Reuse Standard V3.2 Section 04 IP Interface for details about the DMA usage. The application must ensure that the DMA controller of the related device is programmed appropriately.
- **RX Buffer, data read via IPS registers**: By reading the RX Buffer Data Registers 0–14 (QSPI_RBDR0–QSPI_RBDR14) the individual entries in the RX Buffer can be accessed

arbitrarily. Refer to Section 30.4.3.15, RX Buffer Data Registers 0–14 (QSPI_RBDR0–QSPI_RBDR14), for details.

Note that after clearing the RX Buffer the next entry is written into QSPI_RBDR0. Aside from that there is no information available about the current positions of the RX Buffer read and write pointers.

It is not recommended to use this access scheme for subsequent reads of more than 15 data words.

- **AHB Buffer data read via memory-mapped access**: This kind of access is done by reading one of the addresses assigned to the external serial flash device within the range given in Table 30-35 *under the condition that the data requested are already present in the AHB Buffer or it is currently read from the serial flash device*. If this is not the case a memory-mapped read of the AHB Buffer is triggered like described above). As long as the requested data are already available in the AHB Buffer they are provided to the host. The host can read the available data out of the AHB Buffer in any order.

  If the address requested by the current read is the one currently fetched by the QuadSPI module from the serial flash the execution of the current command remains running with the AHB read access stalled. As soon as the data from the requested address have been read by the QuadSPI module the AHB read access is served. So it's possible to run sequential read from the AHB buffer at arbitrary speed without the need to monitor any information about the availability of the data. Nevertheless this access scheme stalls the AHB bus for the time required to read the data from the serial flash device.

  As long as the host restricts its accesses to the data already in the buffer and the data currently fetched from the serial flash it is possible to run the host read from the AHB Buffer in parallel to the serial flash read into the AHB Buffer.

30.5.3.4 Byte Ordering of Serial Flash Data

Table 30-45 below gives the byte ordering scheme how the byte oriented data space of the serial flash device is mapped into the data space of the external serial flash. This scheme is valid for all read and write operations.

Table 30-45. Byte Ordering of Serial Flash Data in the QuadSPI Module

	Serial Flash Byte Address [1:0]			
	00	**01**	**10**	**11**
QuadSPI Register Bit Position [31:0] (32 Bit data width)	[7:0]	[15:8]	[23:16]	[31:24]
Example Data 0x0123_4567	0x01	0x23	0x45	0x67

Refer to the individual register descriptions for details like misaligned or partial accesses to the QuadSPI address space representing serial flash read data.

30.5.3.5 Serial Flash mode interrupt and DMA requests

In Serial Flash mode the QuadSPI has 8 different flags that can only generate interrupt requests and one flag that can generate interrupt as well as DMA requests. Table 30-46 lists the eight conditions. Note that the flags mentioned in the table relate to the Serial Flash Mode Flag Register (QSPI_SFMFR).

Table 30-46. Serial Flash mode interrupt and DMA request conditions

Condition	Flag (QSPI_SPISR)	DMA
TX Buffer Fill	TBFF	
TX Buffer Underrun	TBUF	
RX Buffer Drain	RBDF	X
RX Buffer Overflow	RBOF	
AHB Buffer Overflow	ABOF	
IP Command Trigger during AHB Access Error	IPAEF	
IP Command Trigger during IP Access	IPIEF	
Instruction Code Error	ICEF	
Transaction Finished	TFF	

Each condition has a flag bit in the Serial Flash Mode Flag Register (QSPI_SFMFR) and a Request Enable bit in the SFM Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER). The RX Buffer Drain Flag (RBDF) has separate enable bits for generating IRQ and DMA requests. Note that not each single flag is represented by an individual IRQ line.

30.5.3.5.1 Transmit buffer fill interrupt request

The Transmit Buffer Fill IRQ indicates that the TX Buffer can accept new data. It is asserted if the QSPI_SFMFR[TBFF] flag is asserted and if the corresponding enable bit (QSIP_SFMRSER[TBFIE]) is set. Refer to Section 30.5.3.6, TX Buffer Operation, for details about the assertion of the QSPI_SFMFR[TBFF] flag.

30.5.3.5.2 Receive Buffer Drain Interrupt or DMA Request

The Receive Buffer Drain IRQ derived from the QSPI_SFMFR[RBDF] flag indicates that the RX Buffer of the QuadSPI module has data available from the serial flash device to be read by the host. It remains set as long as the RX Buffer is not empty. The QSPI_SFMRSER[RBDIE] bit enables the related IRQ.Aside from the IRQ it is possible to handle RX Buffer drain by DMA. If the QSPI_SFMRSER[RBDDE] bit is set each write of the module into the RX Buffer triggers a DMA request. The application must set the environment appropriately (for example, the DMA controller) for the DMA transfers.

30.5.3.5.3 Buffer Overflow/Underrun Interrupt Request

The Buffer Overflow/Underrun IRQ is a combination of the following flags (all located in the QSPI_SPIFR register with the related enable bits in the QSPI_SPIRSER register):

- TBUF—TX Buffer Underrun, enabled by TBU_IE
- RBOF—RX Buffer Overflow, enabled by RBO_IE
- ABOF—AHB Buffer Overflow, enabled by ABO_IE

The Transmit Buffer Underrun indicates that an underrun condition in the TX Buffer has occurred. It is generated when the TX Buffer is empty, a transfer to the serial flash is initiated and the QSPI_SPIRSER[TFUF_IE] bit is set.

The Receive Buffer Overflow indicates that an overflow condition in the RX Buffer has occurred. It is generated when the RX Buffer is full, an additional read transfer attempts to write into the RX Buffer and the QSPI_SFMRSER[RBO_IE] bit is set.

The AHB Buffer Overflow indicates that an overflow condition in the AHB Buffer has occurred. It is generated when the AHB Buffer is full, an additional read transfer attempts to write into the AHB Buffer and the QSPI_SFMRSER[ABO_IE] bit is set.

The data from the transfers that generated the individual overflow conditions are ignored.

30.5.3.5.4 Serial Flash Communication Error Interrupt Request

The IPAEF, IPIEF or ICEF flags in the QSPI_SFMSR and the related interrupt enable bits in the QSPI_SFMRSER determine the assertion of the SFMERRIRQ interrupt line.

30.5.3.5.5 Transaction Finished Interrupt Request

The Transaction Finished IRQ indicates the completion of the current command. It is masked by the QSPI_SFMSR[TF_IE] bit.

30.5.3.6 TX Buffer Operation

The TX Buffer provides the data used for page programming. For proper operation it is required to provide at least one entry in the TX Buffer prior to starting the execution of the page programming command. The application must ensure that the required number of data bytes is written into the TX Buffer fast enough as long as the command is executed without a TX Buffer overflow or underrun.

The QuadSPI module sets the QSPI_SFMFR[TBFF] flag initially when entering the SFM mode and subsequently as long as the TX Buffer can accept more data to be written into.

When the QuadSPI module tries to pull data out of an empty TX Buffer the TX Buffer underrun is signaled by the QSPI_SFMFR[TBUF] flag and the current instruction is terminated without further write access to the serial flash and consequently no further assertions of the QSPI_SFMFR[TBFF] flag.

The TX Buffer overflow isn't signaled explicitly, but the TX Buffer fill level can be monitored by the QSPI_TBSR[TRBFL] field.

Refer to Section 30.4.3.16, TX Buffer Status Register (QSPI_TBSR), and Section 30.4.3.20, Serial Flash Mode Flag Register (QSPI_SFMFR), for details about the TX Buffer related registers and to Section 30.5.3.5.1, Transmit buffer fill interrupt request, for details about the usage of the associated interrupt.

30.5.4 Power saving features

The QuadSPI supports three power-saving strategies:

- Stop mode
- Module Disable mode—Clock gating of non-memory mapped logic
- Clock gating of slave bus signals and clock to memory-mapped logic

Like all power saving features, the Stop mode requires logic external to the QuadSPI module for power management and clock gating control. Stop mode

The QuadSPI supports the global signal Stop mode protocol using the ipg_stop → ipg_stop_ack handshake. By default the ipg_stop_ack signal is de-asserted. When a request is made to enter Stop mode, the QuadSPI block acknowledges the request by asserting ipg_stop_ack when it is ready to have its clocks shut off. Depending from the mode of operation the following conditions must be met for the assertion of ipg_stop_ack:

- If a serial transfer is in progress in one of the SPI modes the QuadSPI waits until it reaches the frame boundary before asserting ipg_stop_ack.
- If a SFM command is currently executed in SFM mode the assertion of the ipg_stop_ack is postponed until this command is finished.

While the clocks are shut off, the QuadSPI memory-mapped logic is not accessible. The states of the interrupt and DMA request signals cannot be changed while in Stop mode.

Note that the following actions are illegal in SFM mode during the time starting with raising the request to enter Stop mode and ending with leaving the Stop mode:

- Issue a new SFM command
- Issue a new AHB request

30.5.4.1 Module Disable mode

Module Disable mode is a block-specific mode that the QuadSPI can enter to save power. Host software can initiate the Module Disable mode by writing a 1 to the MDIS bit in the QSPI_MCR.

When a request is encountered to enter the Module Disable mode the QuadSPI negates ipg_enable_clk when it is ready to enter the Module Disable mode. Depending from the mode of operation the following conditions must be met for the negation of ipg_enable_clk:

- If a serial transfer is in progress in one of the SPI modes the QuadSPI waits until it reaches the frame boundary before negating ipg_enable_clk.
- If a SFM command is currently executed in SFM mode the negation of ipg_enable_clk is postponed until this command is finished.

Note that there is only a limited possibility to read back whether the QuadSPI block is waiting for the completion of these conditions or whether it has already negated the ipg_enable_clk. The host software can read the QSPI_SFMSR[BUSY] bit to check for pending execution of a SFM command, but there is no possibility to check pending DMA or CPU read requests on the AHB Buffer.

If implemented, the ipg_enable_clk signal can stop the clock to the non-memory mapped logic. When ipg_enable_clk is negated, the QuadSPI is in a dormant state, but the memory-mapped registers are still accessible. Certain read or write operations have a different effect when the QuadSPI is in the Module Disable mode. Clearing either of the FIFOs will not have any effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the QuadSPI will return the correct values when read, but writing to them will have no effect. Writing to the QSPI_TCR during Module Disable mode will not have any effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

It is not allowed to write to the FIFO registers in this mode.

Note that the following actions are illegal in SFM mode during the time starting with raising the request to enter Module Disable mode and ending with leaving the Module Disable mode:

- Issue a new SFM command
- Issue a new AHB request

30.5.4.2 Leaving power-saving modes

In the Stop mode and the Module Disable mode the clocks to the QuadSPI module are switched off by external circuitry. Note that after the QuadSPI module has left these power saving modes and has returned to normal operation in SFM mode the execution of the first SFM command is deferred until the clock to drive that part of the module related to the serial flash device is available. Depending from the point in time when the first SFM command is programmed the actual execution of that command will start with a slight delay w.r.t. the re-enabling of the clock signal.

30.5.4.3 Slave bus signal gating

The QuadSPI's module enable signal is used to gate slave bus signals such as address, byte enable, read/write and data. This prevents toggling slave bus signals from propagating through parts of the QuadSPI's combinational logic and consuming power unless it is a QuadSPI access. The module enable signal can also be used to gate the clock (ipg_clk_s) to the memory-mapped logic.

30.6 Initialization/application information

30.6.1 How to change queues—SPI modes only

This section presents an example of how to change queues for the QuadSPI. The queues are not part of the QuadSPI, but the QuadSPI includes features in support of queue management. Queues are supported in both SPI modes.

1. Only the last command word from a queue is executed. The EOQ bit in the command word is set to indicate to the QuadSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the QSPI_SPISR is set.
3. The setting of the EOQF flag will disable both serial transmission, and serial reception of data, putting the QuadSPI in the Stopped state. The TXRXS bit is negated to indicate the Stopped state.

4. The DMA will continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable QuadSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in QSPI_SPISR or by checking RFDF in the QSPI_SPISR after each read operation of the QSPI_POPR.
7. Modify DMA descriptor of TX and RX channels for “new” queues
8. Flush TX FIFO by writing a 1 to the CLR_TXF bit in the QSPI_MCR, Flush RX FIFO by writing a 1 to the CLR_RXF bit in the QSPI_MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the QSPI_TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the QuadSPI TX FIFO, and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

30.6.2 Baud rate settings—SPI modes only

Table 30-47 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the QSPI_CTAR registers. The values calculated cover the most usual bus frequencies. They assume that the double baud rate DBR bit is clear. Note that they are rounded appropriately.

Table 30-47. Baud rate values

		Bus clock 64 MHz				Bus clock 100 MHz			
		Baud rate divider prescaler values				Baud rate divider prescaler values			
		2	3	5	7	2	3	5	7
Baud rate scaler values	2	16 M	10.7 M	6.4 M	4.57 M	25.0 M	16.7 M	10.0 M	7.14 M
	4	8 M	5.33 M	3.2 M	2.29 M	12.5 M	8.33 M	5.00 M	3.57 M
	6	5.33 M	3.56 M	2.13 M	1.52 M	8.33 M	5.56 M	3.33 M	2.38 M
	8	4 M	2.67 M	1.6 M	1.14 M	6.25 M	4.17 M	2.50 M	1.79 M
	16	2 M	1.33 M	800 K	571.4 K	3.12 M	2.08 M	1.25 M	893 K
	32	1 M	666.7 K	400 K	285.7 K	1.56 M	1.04 M	625 K	446 K
	64	500 K	333.3 K	200 K	142.9 K	781 K	521 K	312 K	223 K
	128	250 K	166.7 K	100 K	71.4 K	391 K	260 K	156 K	112 K
	256	125 K	83.3 K	50 K	35.7 K	195 K	130 K	78.1 K	55.8 K
	512	62.5 K	41.7 K	25 K	19.9 K	97.7 K	65.1 K	39.1 K	27.9 K
	1024	31.3 K	20.8 K	12.5 K	8.9 K	48.8 K	32.6 K	19.5 K	14.0 K
	2048	15.6 K	10.4 K	6.3 K	4.5 K	24.4 K	16.3 K	9.77 K	6.98 K
	4096	7.8 K	5.2 K	3.1 K	2.2 K	12.2 K	8.14 K	4.88 K	3.49 K
	8192	3.9 K	2.6 K	1.6 K	1.1 K	6.10 K	4.07 K	2.44 K	1.74 K
	16384	2.0 K	1.3 K	781	558	3.05 K	2.04 K	1.22 K	872
	32768	977	651	391	279	1.53 K	1.02 K	610	436

30.6.3 Delay settings—SPI modes only

Table 30-48 shows the values for the Delay after Transfer (T_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the QSPI_CTAR registers. The values reflect the bus frequencies most commonly used, they are rounded appropriately.

Table 30-48. Delay values

		Bus Clock 64 MHz				Bus clock 100 MHz			
		Delay prescaler values				Delay prescaler values			
		1	3	5	7	1	3	5	7
Delay scaler values	2	31.25 ns	93.75 ns	156.25 ns	218.75 ns	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	62.5 ns	187.5 ns	312.5 ns	437.5 ns	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	125 ns	375 ns	625 ns	875 ns	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	250 ns	750 ns	1.25 µs	1.75 µs	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	500 ns	1.5 µs	2.5 µs	3.5 µs	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	1 µs	3 µs	5 µs	7 µs	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	2 µs	6 µs	10 µs	14 µs	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	4 µs	12 µs	20 µs	28 µs	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	8 µs	24 µs	40 µs	56 µs	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	16 µs	48 µs	80 µs	112 µs	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	32 µs	96 µs	160 µs	224 µs	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	64 µs	192 µs	320 µs	448 µs	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	128 µs	384 µs	640 µs	896 µs	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	256 µs	768 µs	1.28 ms	1.8 ms	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	512 µs	1.5 ms	2.6 ms	3.6 ms	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	1 ms	3.1 ms	5.1 ms	7.2 ms	655.4 µs	2.0 ms	3.3 ms	4.6 ms

30.6.4 Oak family compatibility with the QuadSPI—SPI modes only

Table 30-49 shows the translation of commands written to the TX FIFO command halfword with commands written to the Command Ram of the Oak family QSPI. The table illustrates how to configure the QSPI_CTAR registers to match the default cases for the possible combinations of the Oak Family Control Bits in its Command RAM. The defaults for the Oak Family are based on a system clock of 40MHz. All delay variables below will generate the same delay, or as close a possible, from the QuadSPI 100MHz system clock that an Oak Family part would generate from its 40MHz system clock. For other system clock frequencies, the customer can recompute the values using Delay settings—SPI modes only.

- For BITSE = 0 → 8 bits per transfer
- For DT = 0 → 0.425µs delay: For this value, the closest value in the QuadSPI is 0.480µs
- For DSCK = 0 → 1/2 SCK period: For this value, the value for the QuadSPI is 20ns

Table 30-49. Oak family QuadSPI Compatibility with the QuadSPI

Oak family control bits QuadSPI corresponding control bits						Corresponding QSPI_CTAR Register Setting					
BITSE	CTAS[0]	DT	CTAS[1]	DSCK	CTAS[2]	QSPI_CTAR*x*	FMSZ	PDT	DT	PCSSCK	CSSCK
0		0		0		0	1111	10	0011	00	0000
0		0		1		1	1111	10	0011	user	user
0		1		0		2	1111	user[1]	user	00	0000
0		1		1		3	1111	user	user	user	user
1		0		0		4	user	10	0011	00	0000
1		0		1		5	user	10	0011	user	user
1		1		0		6	user	user	user	00	0000
1		1		1		7	user	user	user	user	user

[1] Selected by user

30.6.5 Calculation of FIFO pointer addresses—SPI modes only

The user has complete visibility of the TX and RX FIFO contents through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). Figure 30-35 illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over to the RX FIFO. See Section 30.5.2.5, Transmit First In First Out (TX FIFO) Buffering Mechanism, and Section 30.5.2.6, Receive First In First Out (RX FIFO) Buffering Mechanism, for details on the FIFO operation.

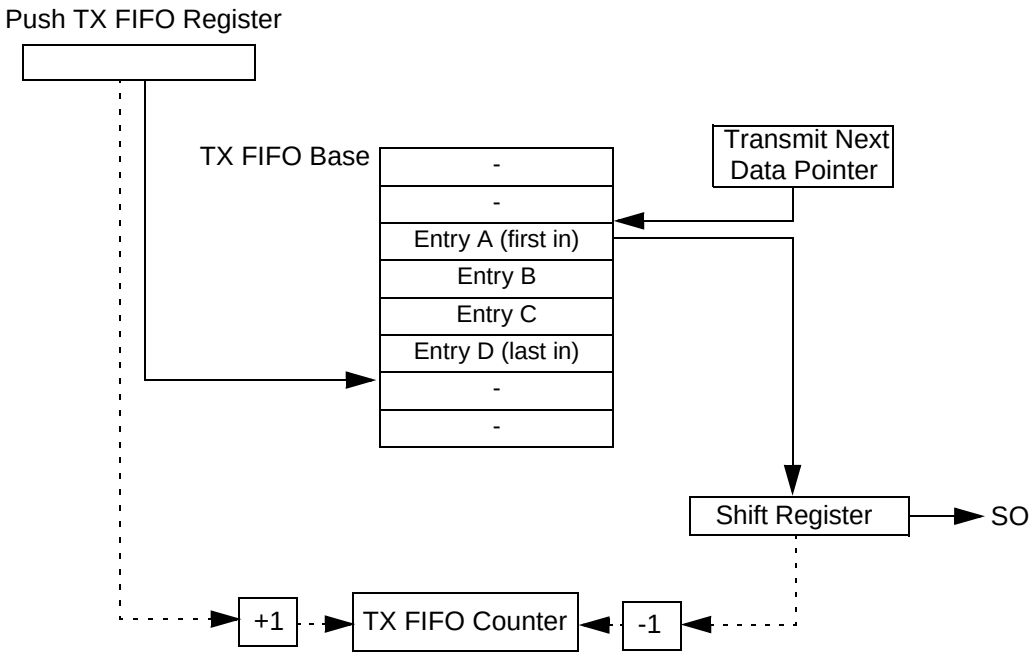


Figure 30-35. TX FIFO Pointers and Counter

30.6.5.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in Entry Address} = \text{TX FIFO Base} + (4 \times \text{TXNXTPTR}) \qquad \textit{Eqn. 30-7}$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-in Entry address} = \text{TX FIFO Base} + 4 \times \text{modulo TX FIFO depth}(\text{TXCTR} + \text{TXNXTPTR} - 1) \qquad \textit{Eqn. 30-8}$$

TX FIFO Base—Base address of TX FIFO
TXCTR—TX FIFO Counter
TXNXTPTR—Transmit Next Pointer
TX FIFO Depth—implementation specific

30.6.5.2 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in Entry Address} = \text{RX FIFO Base} + (4 \times \text{POPNXTPTR}) \qquad \textit{Eqn. 30-9}$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-in Entry address} = \text{RX FIFO Base} + 4 \times \text{modulo RX FIFO depth}(\text{RXCTR} + \text{POPNXTPTR} - 1) \qquad \textit{Eqn. 30-10}$$

RX FIFO Base—Base address of RX FIFO

RXCTR—RX FIFO counter
POPNXTPTR—Pop Next Pointer
RX FIFO Depth—implementation specific

30.6.6 Available status/flag information—SFM mode only

This paragraph gives an overview over the different status and flag information in the SFM mode and their interdependencies for different use cases. Related registers are QSPI_SFMSR and QSPI_SFMFR. Refer to the related descriptions how to set up the QuadSPI module appropriately.

30.6.6.1 IP bus related commands

Refer to Section 30.4.3.12, Instruction Code Register (QSPI_ICR), for additional details not explicitly covered in this paragraph.

30.6.6.1.1 IP bus related commands—normal operation

Writing the QSPI_ICR[IC] field triggers the execution of a new command. Given that this is a legal command the QSPI_SFMSR[IPACC] and the QSPI_SFMSR[BUSY] bits are asserted simultaneously immediately after the execution is started.

When the instruction on the serial flash device has been finished these bits are de-asserted and the QSPI_SFMFR[TFF] flag is set.

30.6.6.1.2 IP bus related commands—error situations

Refer to Table 30-50 below.

30.6.6.2 AHB bus related commands

Refer to Section 30.5.3.3.1, Reading Serial Flash Data into the QuadSPI Module, for additional details not explicitly covered in this paragraph.

30.6.6.2.1 AHB bus related commands—normal operation

Memory-mapped read access to a serial flash address not covered in the AHB Buffer triggers the execution of an AHB Command. Given that this is a legal command the QSPI_SFMSR[AHBACC] and the QSPI_SFMSR[BUSY] bits are asserted simultaneously immediately after the execution is started.

When the instruction on the serial flash device has been finished these bits are de-asserted and the QSPI_SFMFR[TFF] flag is set.

30.6.6.2.2 IP bus related commands—error situations

Refer to Table 30-50 below.

30.6.6.3 Overview of error flags

Table 30-50 below gives an overview of the different error flags in the QSPI_SFMFR register and additional error-related details.

Table 30-50. Overview of QSPI_SFMFR error flags

Error category	Error flag in QSPI_SFMFR	Command Execution on serial flash device TFF Behavior	Description
Command arbitration errors	IPIEF	no → TFF not asserted in conjunction with that command	• IP Command already running, another IP Command could not be executed. • IP Command already running, write attempt to QSPI_ICR register. • IP Command already running, write attempt to QSPI_SFAR register.
	IPAEF		• AHB Command already running, another IP Command could not be executed. • AHB Command already running, write attempt to QSPI_ICR[IC] field.
Instruction code error	ICEF	no → TFF not asserted in conjunction with that command	• Instruction Code Error or Mode Bit Collision[1]
Buffer related errors	RBOF	yes → TFF is asserted on completion	• RX Buffer Overrun
	TBUF		• TX Buffer Underrun
	ABOF		• AHB Buffer Overrun

[1] Refer to Section 30.4.3.19, Serial Flash Mode Status Register (QSPI_SFMSR), SFMSR[CONTMODE] for the description of a mode bit collision

Note that only the buffer related errors are related to a transaction on the external serial flash. All the other errors do not trigger an actual transaction.

30.6.6.4 IP bus and AHB Access command collisions

There are two flags related to this topic, the QSPI_SFMFR[IPAEF] and QSPI_SFMFR[IPIEF]. Refer to Section 30.5.3.3.1, Reading Serial Flash Data into the QuadSPI Module, for a description of the flags itself and to Section 30.6.7, Command arbitration—SFM mode only, for details about possible command collisions.

30.6.7 Command arbitration—SFM mode only

In case of coinciding or overlapping commands the arbitration scheme is as follows: During the execution of an IP Bus related command the running command can’t be terminated by issuing another IP Bus or AHB

Bus related command. The QSPI_SFMFR[IPIEF] flag is asserted when the host tries to trigger an IP Bus related command. When the host triggers an AHB Bus related command (refer to Section 30.5.3.3.1, Reading Serial Flash Data into the QuadSPI Module, for details) this command is stalled until the currently running IP Bus related command is finished.

During the execution of an AHB Bus related command the running command can't be terminated by issuing an IP Bus related command. The command is ignored and the QSPI_SFMFR[IPAEF] flag is asserted. Refer to Section 30.4.3.20, Serial Flash Mode Flag Register (QSPI_SFMFR), for the description of these flags.

When another AHB Bus related command is triggered the address of the memory-mapped access is considered. If the requested address is currently read from the serial flash device the running command is continued. If this is not the case the currently running command is terminated and another AHB Bus related command related to the requested address is executed. Refer to Section 30.5.3.3.1, Reading Serial Flash Data into the QuadSPI Module, for further details.

The commands ignored in case of command collision will not result in the assertion of the QSPI_SFMFR[TFF] flag. It's up to the application to watch the error flags provided to assign the assertions of the TFF flag to the appropriate commands.

30.6.8 DMA usage

For the complete description of the DMA module refer to the related chapter. In this paragraph only the details specific to the DMA usage related to the QuadSPI module are given.

30.6.8.1 DMA usage in SPI Slave mode

30.6.8.1.1 DMA setup in SPI Slave mode

When using the DMA in the SPI Slave mode the standard IP DMA interface protocol is used. For proper operation the DMA controller must be set up in the following way:

- Size of the source minor loop must be set to 2, corresponding to the width of the QSPI_POPR[RXDATA] field.
- Size of the source major loop must be set to the number of bytes which are expected from the SPI Master.
- Source address must be set to the address of the QSPI_POPR register.
- Source address increment must be set to 0.
- Remaining DMA controller setup depends from the application.

30.6.8.2 DMA usage in SFM mode

30.6.8.2.1 Bandwidth considerations in SFM mode

Careful consideration of the throughput rate of the entire chain (serial flash → AHB bus → DMA controller) involved in the read data process is essential for proper operation. Such analysis must take into

account not only the data rate provided by the serial flash but also the data rate of the AHB bus and the performance of the DMA controller in reading data from the RX Buffer.

Depending from the clock frequency of the serial flash and from the clock frequency of the AHB bus it may be that the serial flash read bandwidth is higher than the data are read out from the RX Buffer. If this is the case the RX Buffer will be filled up continuously. The limiting factor is the number of bytes read out from the serial flash, this read out must be finished before the RX Buffer is forced in the overrun condition. This is illustrated in the examples below, the common setup for both of them is:

- AHB bus clock frequency 64 MHz
- RX Buffer read rate (two successive reads of the RX Buffer by DMA) 11 cycles
- Serial flash clock identical to AHB bus clock (64 MHz)
- No other DMA transfers slowing down the DMA channel assigned to the QuadSPI module
- 14 out of the 15 entries in the RX Buffer available for actual buffering

Table 30-51. DMA example in SFM mode

	Fast read dual I/O	Fast read quad I/O
Number of cycles to read 4 bytes from serial flash	16	8
Number of cycles to transfer 4 bytes via DMA	11	
Number of bytes available for buffering (14 * 4)	56	
Number of cycles to fill up available buffer	N/a	3 * (14 * 4) = 168
Number of 4 byte words read from serial flash in that time	N/a	168 / 8 = 21

From Table 30-51 above it can be seen that despite the assumed undisturbed DMA transfers the transfer length is limited to e.g. 16 words when reading the serial flash in Quad I/O mode. Careful analysis of the DMA usage in the entire device containing the QuadSPI module is required to avoid RX Buffer overrun. It is highly recommended to enable the interrupt associated with the QSPI_SFMFR[RBOF] to notice this overrun condition.

When running the serial flash in Dual I/O mode the RX Buffer read rate is higher than the data from the serial flash are written to the RX Buffer so the RX Buffer overrun can only happen if the DMA channel of the QuadSPI is stalled for more than (16 * 14) = 224 cycles in sequence or if the average DMA transfer rate is lengthened from 11 cycles to more than 16 cycles.

30.7 Serial Flash Devices

Currently flash memory devices with serial flash bus are developed by several vendors. Most standard commands currently have the same instruction code for all vendors, some commands are however unique for one vendor. The currently supported list of instruction codes and the required instruction code options are provided in this chapter.

30.7.1 Supported Instruction Codes in Winbond Devices

Table 30-52. Winbond Instruction Codes

Command	Instruction Code	Required Input Data / Parameters			Output Data
		Options QSPI_ICR[[ICO]	Address [QSPI_SFAD]	Data [TX Buffer]	Status / Data [RX Buffer]
Write Enable	06h				
Write Disable	04h				
Read Status Reg1	05h	size			S7–S0 [8]
Read Status Reg2	35h	size			S15–S8 [8]
Write Status Reg	01h	size		S15–S0	
Page Program	02h	size	A23–A0 [24]	size × (D7 - D0)[1]	
Quad Page Program	32h	size	A23–A0 [24]	size × (D7 - D0)	
32k Block Erase	52h		A23–A0 [24]		
64k Block Erase	D8h		A23–A0 [24]		
Sector Erase	20h		A23–A0 [24]		
Chip Erase	C7h / 60h				
Erase Suspend	75h				
Erase Resume	7Ah				
Power down	B9h				
High Performance mode	A3h				
Read Data	03h[2]	size	A23–A0 [24]		size × (D7–D0)[1]
Fast Read	0Bh	size	A23–A0 [24]		size × (D7–D0)
Fast Read Dual Output	3Bh	size	A23–A0 [24]		size × (D7–D0)
Fast Read Dual I/O	BBh	M7–M0, size	A23–A0 [24]		size × (D7–D0)
Fast Read Quad Output	6Bh	size	A23–A0 [24]		size × (D7–D0)
Fast Read Quad I/O	EBh	M7–M0, size	A23–A0 [24]		size × (D7–D0)
Octal Word Read Quad I/O	E3h	M7–M0, size	A23–A0 [24]		size × (D7–D0)
(Continuous Read) mode Bit Reset	FFh	(FFh), size			
Release PowerDown/HighPerf.Mode (optional): read device ID	ABh	(opt.): read size			(opt.): ID7– ID0
Read Manufacturer/Device ID	90h	size	A23–A0 [24] = 0h / 1h		ManID7–ManID0 , DevID7–Dev0

Table 30-52. Winbond Instruction Codes

Command	Instruction Code	Required Input Data / Parameters			Output Data
		Options QSPI_ICR[[ICO]	Address [QSPI_SFAD]	Data [TX Buffer]	Status / Data [RX Buffer]
Read Unique ID	4Bh	size			UniqID63– UniqID0
Read JEDEC ID	9Fh	size			ManID7–ManID0, MemID7–Mem0, CapID7–CapID0

[1]Denotes that 'size'-times one byte is transferred on the serial flash data bus. Total number of bytes must be provided in the TX Buffer or can be read from the RX Buffer

[2]According to Winbond documentation this command only supports a maximum clock speed of 50 MHz. If the Serial Flash is operated at a higher clock frequency, the clock frequency for this command must be decreased. Refer to Section 30.7.2, Serial Flash Clock Frequency Limitations, for details.

The following table shows only the upper 3 bytes of register QSPI_ICR, byte 0 contains the Instruction Code. All sizes in Byte 2 are interpreted as the number of bytes, the number of sclk clocks required in the transaction will be calculated by multiplying with 8 and dividing by number of data lines.

Table 30-53. Instruction Code Options on Winbond Devices

Instruction Code	Byte 3								Byte 2								Byte 1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
01h															Size									
02h	Size (bytes to be written)																							
05h															Size									
32h	Size (bytes to be written)																							
35h															Size									
03h	Size (bytes to be read)																							
0Bh	Size (bytes to be read)																							
3Bh	Size (bytes to be read)																							
BBh	Size (bytes to be read)																M7–M0							
6Bh	Size (bytes to be read)																							
E3h	Size (bytes to be read)																M7–M0							
EBh	Size (bytes to be read)																M7–M0							
ABh															read[1]									
90h															Size									
4Bh													Size (bytes)											

Table 30-53. Instruction Code Options on Winbond Devices (continued)

Instruction Code	Byte 3								Byte 2								Byte 1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
9Fh															Size									
FFh																Size	opt. FFh[2]							

1 'read' bit controls if 8 SCLK clocks are added to read the device ID, equivalent to Size = 1 (1byte to read)

2 To reset continuos read mode while in Dual I/O operations, set size to 1 and Byte 1 to FFh (ICR = 0001FFFFh). To reset the continuos read mode while in Quad I/O operations, only instruction FFh is required (ICR = 000000FFh).

30.7.2 Serial Flash Clock Frequency Limitations

Certain commands of the Winbond serial flash devices are limited in the frequency applied to the serial flash device on command execution. To allow for higher clock speeds for the remaining commands the serial flash device clock can be divided by 2 (half speed) when executing such a command limited in frequency by setting the QSPI_SMPR[HSENA] bit. Refer to Table 30-52 for the commands affected.

30.8 Internal Sampling of Serial Flash Input Data

Depending from the actual implementation there is a delay between the internal clocking in the QuadSPI module and the external serial flash device. This means that the incoming data from the serial flash appear this delay later in time at the QuadSPI sampling logic w.r.t. internal reference clock. Refer to Figure 30-36 for an overview of this scheme.

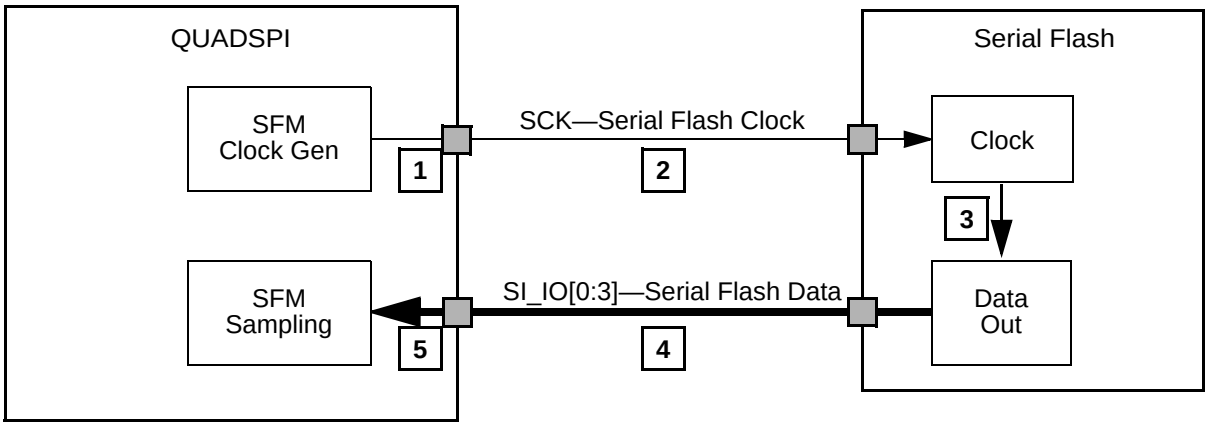


Figure 30-36. Serial Flash Sampling Clock Overview

NOTE

The arrival of the serial flash data in the sampling stage of the QuadSPI module are given in fig Figure 30-37 below. Note that the amount of the total delay $t_{Del,total}$ is very specific to the characteristics of the actual implementation.

Note also that the serial flash device clock SCK is inverted w.r.t. the QuadSPI internal reference clock.

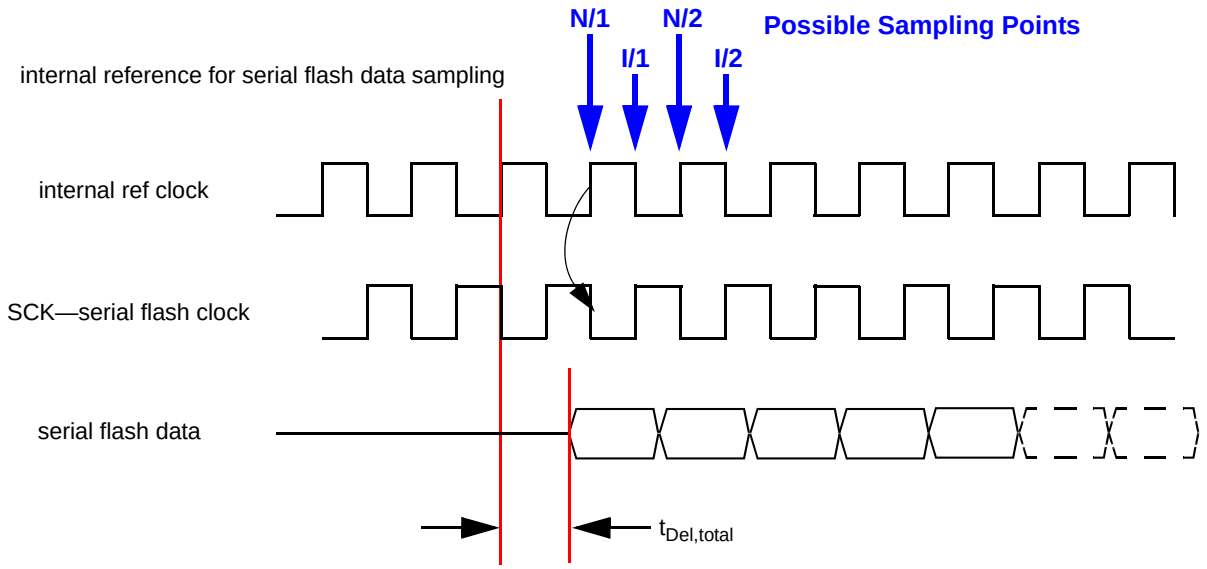


Figure 30-37. Serial Flash Sampling Clock Timing

The rising edge of the internal reference clock is taken as timing reference for the data output of the serial flash. After a time of $t_{Del,total}$ the data arrive at the internal sampling stage of the QuadSPI module. According to Figure 30-36 the following parts of the delay chain contribute to $t_{Del,total}$:

1. Output delay of the serial flash clock output of the device containing the QuadSPI module
2. Wire delay of application/PCB from the device containing the QuadSPI module to the external serial flash device
3. Clock to data out delay of the external serial flash device, including input and output delays
4. Wire delay of application/PCB from the external serial flash device to the device containing the QuadSPI module
5. Input delay belonging to the data in input

The possible points in time for the sampling of the incoming data are denoted as N/1, I/1, N/2 and I/2 above. The sampling point relevant for the internal sampling is configured in the QSPI_SMPR register, refer to Section 30.4.3.13, Sampling Register (QSPI_SMPR), for details. Note that the falling edges of the reference clock are not actually used, instead the inverted clock is used for sampling at these positions. Table 30-54 below gives an overview of the available configurations for the commands running at regular (full) speed:

Table 30-54. Sampling Configuration

Sampling Point	Description	Delay [FSDLY] [HSDLY]	Phase [FSPHS] [HSPHS]	QSPI_SMPR for Full Speed Setting[1]
N/1	sampling with non-inverted clock, 1 sample delay	0	0	0x0000000x
I/1	sampling with inverted clock, 1 sample delay	0	1	0x0000002x
N/2	sampling with non-inverted clock, 2 samples delay	1	0	0x0000004x
I/2	sampling with inverted clock, 2 samples delay	1	1	0x0000006x

[1] 'x' is not considered here

Depending from the actual delay and the serial flash clock frequency the appropriate sampling point can be chosen. The following remarks should be considered when selecting the appropriate setting:

- Theoretically there should be 2 settings possible to capture the correct data since the serial flash output is valid for 1 clock cycle, disregarding rise and fall times and timing uncertainties.
- Depending from the timing uncertainties it may turn out in actual applications that only one possible sample positions remains. This is subject to careful consideration depending from the actual implementation.
- The delay $t_{Del,total}$ is an absolute size to shift the point in time when the serial flash date get valid at the QuadSPI input.
- For decreasing frequency of the serial flash clock the distance between the edges increases. So for large differences in the frequency the required setting may change.
- For commands running at half of the regular serial flash clock (QSPI_SMPR[HSENA] bit set) the sampling point must be figured separately to allow for the compensation of the absolute shift in time w.r.t. the sample-relative setting in the QSPI_SPMPR register.

Chapter 31
Reset Generation Module (MC_RGM)

31.1 Introduction

31.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the device. It provides a register interface and the reset sequencer. The different registers are available to monitor and control the device reset sequence. The reset sequencer is a state machine which controls the different phases (Phase0, Phase1, Phase2, Phase3, and Idle) of the reset sequence and control the reset signals generated in the system.

Figure 31-1 depicts the MC_RGM block diagram.

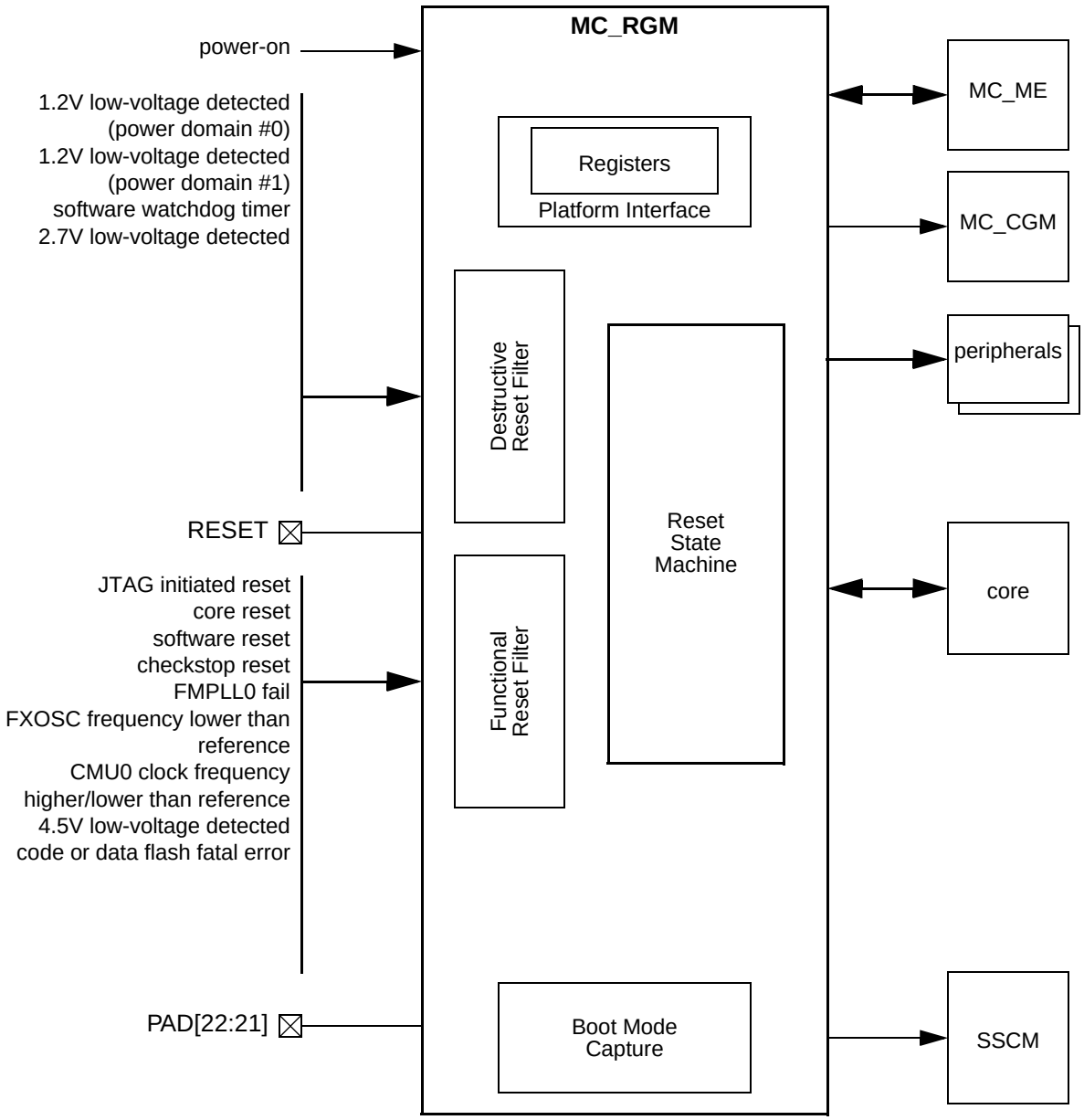


Figure 31-1. MC_RGM block diagram

31.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'Destructive' resets management
- 'Functional' resets management
- Signalling of reset events after each reset sequence (reset status flags)
- Conversion of reset events to Safe mode or interrupt request events (for further mode details, please see the MC_ME chapter)
- Short reset sequence configuration

- Bidirectional reset behavior configuration
- Selection of alternate boot via the backup RAM on Standby mode exit (for further mode details, please see Chapter 31, Reset Generation Module (MC_RGM))
- Boot mode capture on RESET deassertion

31.1.3 Modes of operation

The different reset sources are organized into two families: ‘destructive’ and ‘functional’.

- A ‘destructive’ reset source is associated with an event related to a critical—usually hardware—error or dysfunction. When a ‘destructive’ reset event occurs, the full reset sequence is applied to the device starting from Phase0. This resets the full device ensuring a safe startup state for both digital and analog modules. ‘Destructive’ resets are
  - power-on reset
  - 1.2V low-voltage detected (power domain #0)
  - 1.2V low-voltage detected (power domain #1)
  - software watchdog timer
  - 2.7V low-voltage detected
- A ‘functional’ reset source is associated with an event related to a less-critical—usually non-hardware—error or dysfunction. When a ‘functional’ reset event occurs, a partial reset sequence is applied to the device starting from Phase1. In this case, most digital modules are reset normally, while analog modules or specific digital modules’ (e.g. debug modules, flash modules) state is preserved. ‘Functional’ resets are
  - external reset
  - JTAG initiated reset
  - core reset
  - software reset
  - checkstop reset
  - FMPLL0 fail
  - FXOSC frequency lower than reference
  - CMU0 clock frequency higher/lower than reference
  - 4.5V low-voltage detected
  - code or data flash fatal error

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e. Phase*n* states). Each phase is associated with a particular device reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The device reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the Idle phase. During this entire process, the MC_ME state machine is held in Reset mode. Only at the end of the reset sequence, when the Idle phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a Safe mode request issued to the MC_ME or to an interrupt issued to the core (see Section 31.3.1.4, Destructive Event Reset Disable Register (RGM_DERD), and Section 31.3.1.6, Destructive Event Alternate Request Register (RGM_DEAR), for 'destructive' resets and Section 31.3.1.3, Functional Event Reset Disable Register (RGM_FERD), and Section 31.3.1.5, Functional Event Alternate Request Register (RGM_FEAR), for 'functional' resets).

31.2 External signal description

The MC_RGM interfaces to the bidirectional reset pin RESET and the boot mode pins PAD[22:21].

31.3 Memory map and register definition

Table 31-1. MC_RGM Register description

Address	Name	Description	Size	Access
0xC3FE_4000	RGM_FES	Functional Event Status	half-word	read/write[1]
0xC3FE_4002	RGM_DES	Destructive Event Status	half-word	read/write[1]
0xC3FE_4004	RGM_FERD	Functional Event Reset Disable	half-word	read/write[2]
0xC3FE_4006	RGM_DERD	Destructive Event Reset Disable	half-word	read
0xC3FE_4010	RGM_FEAR	Functional Event Alternate Request	half-word	read/write
0xC3FE_4012	RGM_DEAR	Destructive Event Alternate Request	half-word	read
0xC3FE_4018	RGM_FESS	Functional Event Short Sequence	half-word	read/write
0xC3FE_401A	RGM_STDBY	Standby Reset Sequence	half-word	read/write
0xC3FE_401C	RGM_FBRE	Functional Bidirectional Reset Enable	half-word	read/write

1 individual bits cleared on writing 1

2 write once: 0 = disable, 1 = enable.

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- Not change register content
- Cause a transfer error

Table 31-2. MC_RGM memory map

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FE_4000	RGM_FES / RGM_DES	R	F_EXR							F_FLASH	F_LVD45	F_CMU0_FHL	F_CMU0_OLR	F_FMPLL0	F_CHKSTOP	F_SOFT	F_CORE	F_JTAG
		W	w1c							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
		R	F_POR												F_LVD27	F_SWT	F_LVD12_PD1	F_LVD12_PD0
		W	w1c												w1c	w1c	w1c	w1c
0xC3FE_4004	RGM_FERD / RGM_DERD	R	D_EXR							D_FLASH	D_LVD45	D_CMU0_FHL	D_CMU0_OLR	D_FMPLL0	D_CHKSTOP	D_SOFT	D_CORE	D_JTAG
		W																
		R	0												D_LVD27	D_SWT	D_LVD12_PD1	D_LVD12_PD0
		W																
0xC3FE_4008 … 0xC3FE_400C	Reserved																	
0xC3FE_4010	RGM_FEAR / RGM_DEAR	R	AR_EXR							AR_FLASH	AR_LVD45	AR_CMU0_FHL	AR_CMU0_OLR	AR_FMPLL0	AR_CHKSTOP	AR_SOFT	AR_CORE	AR_JTAG
		W																
		R	0												AR_LVD27	AR_SWT	AR_LVD12_PD1	AR_LVD12_PD0
		W																
0xC3FE_4014	Reserved																	

Table 31-2. MC_RGM memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FE_4018	RGM_FESS / RGM_STDBY	R	SS_EXR							SS_FLASH	SS_LVD45	SS_CMU0_FHL	SS_CMU0_OLR	SS_FMPLL0	SS_CHKSTOP	SS_SOFT	SS_CORE	SS_JTAG
		W																
		R	0	0	0	0	0	0	0	0	BOOT_FROM_BKP_RAM	0	0	0	0	0	0	0
		W																
0xC3FE_401C	RGM_FBRE	R	BE_EXR							BE_FLASH	BE_LVD45	BE_CMU0_FHL	BE_CMU0_OLR	BE_FMPLL0	BE_CHKSTOP	BE_SOFT	BE_CORE	BE_JTAG
		W																
0xC3FE_4020 … 0xC3FE_7FFC	Reserved																	

31.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_STDBY register may be accessed as a word at address 0xC3FE_4018, as a half-word at address 0xC3FE_401A, or as a byte at address 0xC3FE_401B.

31.3.1.1 Functional Event Status Register (RGM_FES)

Address 0xC3FE_4000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F_EXR							F_FLASH	F_LVD45	F_CMU0_FHL	F_CMU0_OLR	F_FMPLL0	F_CHKSTOP	F_SOFT	F_CORE	F_JTAG
W	w1c							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-2. Functional Event Status Register (RGM_FES)

This register contains the status of the last asserted functional reset sources. It can be accessed in read/write on either supervisor mode or test mode. Register bits are cleared on write 1.

NOTE

Clearing each flag in this register requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.

Table 31-3. Functional Event Status Register (RGM_FES) field descriptions

Field	Description
F_EXR	**Flag for External Reset** 0 No external reset event has occurred since either the last clear or the last power-on reset 1 An external reset event has occurred
F_FLASH	**Flag for code or data flash fatal error** 0 No code or data flash fatal error event has occurred since either the last clear or the last power-on reset 1 A code or data flash fatal error event has occurred
F_LVD45	**Flag for 4.5V low-voltage detected** 0 No 4.5V low-voltage detected event has occurred since either the last clear or the last power-on reset 1 A 4.5V low-voltage detected event has occurred
F_CMU0_FHL	**Flag for CMU0 clock frequency higher/lower than reference** 0 No CMU0 clock frequency higher/lower than reference event has occurred since either the last clear or the last power-on reset 1 A CMU0 clock frequency higher/lower than reference event has occurred
F_CMU0_OLR	**Flag for FXOSC frequency lower than reference** 0 No FXOSC frequency lower than reference event has occurred since either the last clear or the last power-on reset 1 A FXOSC frequency lower than reference event has occurred
F_FMPLL0	**Flag for FMPLL0 fail** 0 No FMPLL0 fail event has occurred since either the last clear or the last power-on reset 1 A FMPLL0 fail event has occurred

Table 31-3. Functional Event Status Register (RGM_FES) field descriptions (continued)

Field	Description
F_CHKSTOP	**Flag for checkstop reset** 0 No checkstop reset event has occurred since either the last clear or the last power-on reset 1 A checkstop reset event has occurred
F_SOFT	**Flag for software reset** 0 No software reset event has occurred since either the last clear or the last power-on reset 1 A software reset event has occurred
F_CORE	**Flag for core reset** 0 No core reset event has occurred since either the last clear or the last power-on reset 1 A core reset event has occurred
F_JTAG	**Flag for JTAG initiated reset** 0 No JTAG initiated reset event has occurred since either the last clear or the last power-on reset 1 A JTAG initiated reset event has occurred

31.3.1.2 Destructive Event Status Register (RGM_DES)

Address 0xC3FE_4002 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F_POR												F_LVD27	F_SWT	F_LVD12_PD1	F_LVD12_PD0
W	w1c												w1c	w1c	w1c	w1c
POR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-3. Destructive Event Status Register (RGM_DES)

This register contains the status of the last asserted destructive reset sources. It can be accessed in read/write on either supervisor mode or test mode. Register bits are cleared on write 1.

NOTE

Clearing each flag in this register requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.

Table 31-4. Destructive Event Status Register (RGM_DES) field descriptions

Field	Description
F_POR	**Flag for Power-On reset** 0 No power-on event has occurred since the last clear (due to either a software clear or a low-voltage detection) 1 A power-on event has occurred
F_LVD27	**Flag for 2.7V low-voltage detected** 0 No 2.7V low-voltage detected event has occurred since either the last clear or the last power-on reset assertion 1 A 2.7V low-voltage detected event has occurred
F_SWT	**Flag for software watchdog timer** 0 No software watchdog timer event has occurred since either the last clear or the last power-on reset assertion 1 A software watchdog timer event has occurred
F_LVD12_PD1	**Flag for 1.2V low-voltage detected (power domain #1)** 0 No 1.2V low-voltage detected (power domain #1) event has occurred since either the last clear or the last power-on reset assertion 1 A 1.2V low-voltage detected (power domain #1) event has occurred
F_LVD12_PD0	**Flag for 1.2V low-voltage detected (power domain #0)** 0 No 1.2V low-voltage detected (power domain #0) event has occurred since either the last clear or the last power-on reset assertion 1 A 1.2V low-voltage detected (power domain #0) event has occurred

NOTE

The F_POR flag is automatically cleared on a 1.2V low-voltage detected (power domain #0 or #1) or a 2.7V low-voltage detected (VREG). This means that if the startup sequence is not monotonic (i.e the voltage rises and then drops enough to trigger a low-voltage detection), the F_POR flag may not be set but instead the <register>F_LVD12_PD0, <register>F_LVD12_PD1, or <register>F_LVD27_VREG flag is set on exiting the reset sequence. Therefore, if the F_POR, <register>F_LVD12_PD0, <register>F_LVD12_PD1, or <register>F_LVD27_VREG flags are set on reset exit, software should interpret the reset cause as power-on.

NOTE

In contrast to all other reset sources, the 1.2V low-voltage detected (power domain #0) event is captured on its deassertion. Therefore, the status bit F_LVD12_PD0 is also asserted on the reset's deassertion. In case an alternate event is selected, the Safe mode or interrupt request are similarly asserted on the reset's deassertion.

31.3.1.3 Functional Event Reset Disable Register (RGM_FERD)

Address 0xC3FE_4004 Access: Supervisor read/write

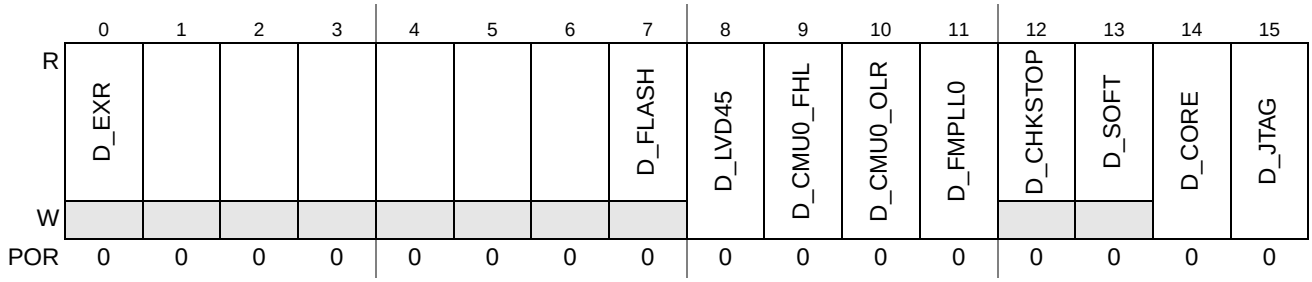


Figure 31-4. Functional Event Reset Disable Register (RGM_FERD)

This register provides dedicated bits to disable functional reset sources.When a functional reset source is disabled, the associated functional event will trigger either a Safe mode request or an interrupt request (see Section 31.3.1.5, Functional Event Alternate Request Register (RGM_FEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

Table 31-5. Functional Event Reset Disable Register (RGM_FERD) field descriptions

Field	Description
D_EXR	**Disable External Reset** 0 An external reset event triggers a reset sequence 1 An external reset event generates a Safe mode request
D_FLASH	**Disable code or data flash fatal error** 0 A code or data flash fatal error event triggers a reset sequence 1 A code or data flash fatal error event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_FLASH
D_LVD45	**Disable 4.5V low-voltage detected** 0 A 4.5V low-voltage detected event triggers a reset sequence 1 A 4.5V low-voltage detected event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_LVD45
D_CMU0_FHL	**Disable CMU0 clock frequency higher/lower than reference** 0 A CMU0 clock frequency higher/lower than reference event triggers a reset sequence 1 A CMU0 clock frequency higher/lower than reference event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_CMU0_FHL
D_CMU0_OLR	**Disable FXOSC frequency lower than reference** 0 A FXOSC frequency lower than reference event triggers a reset sequence 1 A FXOSC frequency lower than reference event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_CMU0_OLR
D_FMPLL0	**Disable FMPLL0 fail** 0 A FMPLL0 fail event triggers a reset sequence 1 A FMPLL0 fail event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_FMPLL0
D_CHKSTOP	**Disable checkstop reset** 0 A checkstop reset event triggers a reset sequence 1 A checkstop reset event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_CHKSTOP

Table 31-5. Functional Event Reset Disable Register (RGM_FERD) field descriptions (continued)

Field	Description
D_SOFT	**Disable software reset** 0 A software reset event triggers a reset sequence 1 A software reset event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_SOFT
D_CORE	**Disable core reset** 0 A core reset event triggers a reset sequence 1 A core reset event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_CORE
D_JTAG	**Disable JTAG initiated reset** 0 A JTAG initiated reset event triggers a reset sequence 1 A JTAG initiated reset event generates either a Safe mode or an interrupt request depending on the value of RGM_FEAR.AR_JTAG

31.3.1.4 Destructive Event Reset Disable Register (RGM_DERD)

Address 0xC3FE_4006 Access: Supervisor read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												D_LVD27	D_SWT	D_LVD12_PD1	D_LVD12_PD0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-5. Destructive Event Reset Disable Register (RGM_DERD)

This register provides dedicated bits to disable particular destructive reset sources. When a destructive reset source is disabled, the associated destructive event will trigger either a safe mode request or an interrupt request (see Section 31.3.1.6, Destructive Event Alternate Request Register (RGM_DEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

Table 31-6. Destructive Event Reset Disable Register (RGM_DERD) field descriptions

Field	Description
D_LVD27	**Disable 2.7V low-voltage detected** 0 A 2.7V low-voltage detected event triggers a reset sequence 1 A 2.7V low-voltage detected event generates either a Safe mode or an interrupt request depending on the value of RGM_DEAR.AR_LVD27
D_SWT	**Disable software watchdog timer** 0 A software watchdog timer event triggers a reset sequence 1 A software watchdog timer event generates either a Safe mode or an interrupt request depending on the value of RGM_DEAR
D_LVD12_PD1	**Disable 1.2V low-voltage detected (power domain #1)** 0 A 1.2V low-voltage detected (power domain #1) event triggers a reset sequence 1 A 1.2V low-voltage detected (power domain #1) event generates either a Safe mode or an interrupt request depending on the value of RGM_DEAR.AR_LVD12_PD1
D_LVD12_PD0	**Disable 1.2V low-voltage detected (power domain #0)** 0 A 1.2V low-voltage detected (power domain #0) event triggers a reset sequence 1 A 1.2V low-voltage detected (power domain #0) event generates either a Safe mode or an interrupt request depending on the value of RGM_DEAR.AR_LVD12_PD0

31.3.1.5 Functional Event Alternate Request Register (RGM_FEAR)

Address 0xC3FE_4010 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AR_EXR							AR_FLASH	AR_LVD45	AR_CMU0_FHL	AR_CMU0_OLR	AR_FMPLL0	AR_CHKSTOP	AR_SOFT	AR_CORE	AR_JTAG
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-6. Functional Event Alternate Request Register (RGM_FEAR)

This register defines an alternate request to be generated when a reset on a functional event has been disabled. The alternate request can be either a Safe mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Table 31-7. Functional Event Alternate Request Register (RGM_FEAR) field descriptions

Field	Description
AR_EXR	**Alternate Request for External Reset** 0 Generate a Safe mode request on an external reset event if the reset is disabled 1 Generate an interrupt request on an external reset event if the reset is disabled
AR_FLASH	**Alternate Request for code or data flash fatal error** 0 Generate a Safe mode request on a code or data flash fatal error event if the reset is disabled 1 Generate an interrupt request on a code or data flash fatal error event if the reset is disabled

Table 31-7. Functional Event Alternate Request Register (RGM_FEAR) field descriptions (continued)

Field	Description
AR_LVD45	**Alternate Request for 4.5V low-voltage detected** 0 Generate a Safe mode request on a 4.5V low-voltage detected event if the reset is disabled 1 Generate an interrupt request on a 4.5V low-voltage detected event if the reset is disabled
AR_CMU0_FHL	**Alternate Request for CMU0 clock frequency higher/lower than reference** 0 Generate a Safe mode request on a CMU0 clock frequency higher/lower than reference event if the reset is disabled 1 Generate an interrupt request on a CMU0 clock frequency higher/lower than reference event if the reset is disabled
AR_CMU0_OLR	**Alternate Request for FXOSC frequency lower than reference** 0 Generate a Safe mode request on a FXOSC frequency lower than reference event if the reset is disabled 1 Generate an interrupt request on a FXOSC frequency lower than reference event if the reset is disabled
AR_FMPLL0	**Alternate Request for FMPLL0 fail** 0 Generate a Safe mode request on a FMPLL0 fail event if the reset is disabled 1 Generate an interrupt request on a FMPLL0 fail event if the reset is disabled
AR_CHKSTOP	**Alternate Request for checkstop reset** 0 Generate a Safe mode request on a checkstop reset event if the reset is disabled 1 Generate an interrupt request on a checkstop reset event if the reset is disabled
AR_SOFT	**Alternate Request for software reset** 0 Generate a Safe mode request on a software reset event if the reset is disabled 1 Generate an interrupt request on a software reset event if the reset is disabled
AR_CORE	**Alternate Request for core reset** 0 Generate a Safe mode request on a core reset event if the reset is disabled 1 Generate an interrupt request on a core reset event if the reset is disabled
AR_JTAG	**Alternate Request for JTAG initiated reset** 0 Generate a Safe mode request on a JTAG initiated reset event if the reset is disabled 1 Generate an interrupt request on a JTAG initiated reset event if the reset is disabled

31.3.1.6 Destructive Event Alternate Request Register (RGM_DEAR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												AR_LVD27	AR_SWT	AR_LVD12_PD1	AR_LVD12_PD0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-7. Destructive Event Alternate Request Register (RGM_DEAR)

This register defines an alternate request to be generated when a reset on a destructive event has been disabled. The alternate request can be either a Safe mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Table 31-8. Destructive Event Alternate Request Register (RGM_DEAR) field descriptions

Field	Description
AR_LVD27	**Alternate Request for 2.7V low-voltage detected** 0 Generate a Safe mode request on a 2.7V low-voltage detected event if the reset is disabled 1 Generate an interrupt request on a 2.7V low-voltage detected event if the reset is disabled
AR_SWT	**Alternate Request for software watchdog timer** 0 Generate a Safe mode request on a software watchdog timer event if the reset is disabled 1 Generate an interrupt request on a software watchdog timer event if the reset is disabled
AR_LVD12_PD1	**Alternate Request for 1.2V low-voltage detected (power domain #1)** 0 Generate a Safe mode request on a 1.2V low-voltage detected (power domain #1) event if the reset is disabled 1 Generate an interrupt request on a 1.2V low-voltage detected (power domain #1) event if the reset is disabled
AR_LVD12_PD0	**Alternate Request for 1.2V low-voltage detected (power domain #0)** 0 Generate a Safe mode request on a 1.2V low-voltage detected (power domain #0) event if the reset is disabled 1 Generate an interrupt request on a 1.2V low-voltage detected (power domain #0) event if the reset is disabled

31.3.1.7 Functional Event Short Sequence Register (RGM_FESS)

Address 0xC3FE_4018 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SS_EXR							SS_FLASH	SS_LVD45	SS_CMU0_FHL	SS_CMU0_OLR	SS_FMPLL0	SS_CHKSTOP	SS_SOFT	SS_CORE	SS_JTAG
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-8. Functional Event Short Sequence Register (RGM_FESS)

This register defines which reset sequence will be done when a functional reset sequence is triggered.The functional reset sequence can either start from Phase1 or from Phase3, skipping Phase1 and Phase2.

NOTE

This could be useful for fast reset sequence, for example to skip flash reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

Table 31-9. Functional Event Short Sequence Register (RGM_FESS) field descriptions

Field	Description
SS_EXR	**Short Sequence for External Reset** 0 The reset sequence triggered by an external reset event will start from Phase1 1 The reset sequence triggered by an external reset event will start from Phase3, skipping Phase1 and Phase2
SS_FLASH	**Short Sequence for code or data flash fatal error** 0 The reset sequence triggered by a code or data flash fatal error event will start from Phase1 1 The reset sequence triggered by a code or data flash fatal error event will start from Phase3, skipping Phase1 and Phase2
SS_LVD45	**Short Sequence for 4.5V low-voltage detected** 0 The reset sequence triggered by a 4.5V low-voltage detected event will start from Phase1 1 The reset sequence triggered by a 4.5V low-voltage detected event will start from Phase3, skipping Phase1 and Phase2
SS_CMU0_FHL	**Short Sequence for CMU0 clock frequency higher/lower than reference** 0 The reset sequence triggered by a CMU0 clock frequency higher/lower than reference event will start from Phase1 1 The reset sequence triggered by a CMU0 clock frequency higher/lower than reference event will start from Phase3, skipping Phase1 and Phase2
SS_CMU0_OLR	**Short Sequence for FXOSC frequency lower than reference** 0 The reset sequence triggered by a FXOSC frequency lower than reference event will start from Phase1 1 The reset sequence triggered by a FXOSC frequency lower than reference event will start from Phase3, skipping Phase1 and Phase2
SS_FMPLL0	**Short Sequence for FMPLL0 fail** 0 The reset sequence triggered by a FMPLL0 fail event will start from Phase1 1 The reset sequence triggered by a FMPLL0 fail event will start from Phase3, skipping Phase1 and Phase2
SS_CHKSTOP	**Short Sequence for checkstop reset** 0 The reset sequence triggered by a checkstop reset event will start from Phase1 1 The reset sequence triggered by a checkstop reset event will start from Phase3, skipping Phase1 and Phase2
SS_SOFT	**Short Sequence for software reset** 0 The reset sequence triggered by a software reset event will start from Phase1 1 The reset sequence triggered by a software reset event will start from Phase3, skipping Phase1 and Phase2
SS_CORE	**Short Sequence for core reset** 0 The reset sequence triggered by a core reset event will start from Phase1 1 The reset sequence triggered by a core reset event will start from Phase3, skipping Phase1 and Phase2
SS_JTAG	**Short Sequence for JTAG initiated reset** 0 The reset sequence triggered by a JTAG initiated reset event will start from Phase1 1 The reset sequence triggered by a JTAG initiated reset event will start from Phase3, skipping Phase1 and Phase2

31.3.1.8 Standby Reset Sequence Register (RGM_STDBY)

Address 0xC3FE_401A Access: Supervisor read/write

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

Figure 31-9. Standby Reset Sequence Register (RGM_STDBY)

This register defines reset sequence to be applied on Standby mode exit. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Table 31-10. Standby Reset Sequence Register (RGM_STDBY) field descriptions

Field	Description
BOOT_ FROM_ BKP_RAM	**Boot from Backup RAM indicator** — This bit indicates whether the system will boot from backup RAM or flash out of Standby exit. 0 Boot from flash on Standby exit 1 Boot from backup RAM on Standby exit

NOTE

This register is reset on any enabled ‘destructive’ or ‘functional’ reset event.

31.3.1.9 Functional Bidirectional Reset Enable Register (RGM_FBRE)

Address 0xC3FE_401C Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE_EXR							BE_FLASH	BE_LVD45	BE_CMU0_FHL	BE_CMU0_OLR	BE_FMPLL0			BE_CORE	BE_JTAG
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-10. Functional Bidirectional Reset Enable Register (RGM_FBRE)

This register enables the generation of an external reset on functional reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

NOTE

It is not possible for a functional event to perform a short reset sequence and also assert the external reset pin. If any functional event is defined to perform a short reset sequence in the RGM_FESS register and also assert the external reset pin, then it will perform a long reset starting with Phase3 and not assert the external reset pin.

Table 31-11. Functional Bidirectional Reset Enable Register (RGM_FBRE) field descriptions

Field	Description
BE_EXR	**Bidirectional Reset Enable for External Reset** 0 RESET is asserted on an external reset event if the reset is enabled 1 RESET is not asserted on an external reset event
BE_FLASH	**Bidirectional Reset Enable for code or data flash fatal error** 0 RESET is asserted on a code or data flash fatal error event if the reset is enabled 1 RESET is not asserted on a code or data flash fatal error event
BE_LVD45	**Bidirectional Reset Enable for 4.5V low-voltage detected** 0 RESET is asserted on a 4.5V low-voltage detected event if the reset is enabled 1 RESET is not asserted on a 4.5V low-voltage detected event
BE_CMU0_FHL	**Bidirectional Reset Enable for CMU0 clock frequency higher/lower than reference** 0 RESET is asserted on a CMU0 clock frequency higher/lower than reference event if the reset is enabled 1 RESET is not asserted on a CMU0 clock frequency higher/lower than reference event
BE_CMU0_OLR	**Bidirectional Reset Enable for FXOSC frequency lower than reference** 0 RESET is asserted on a FXOSC frequency lower than reference event if the reset is enabled 1 RESET is not asserted on a FXOSC frequency lower than reference event
BE_FMPLL0	**Bidirectional Reset Enable for FMPLL0 fail** 0 RESET is asserted on a FMPLL0 fail event if the reset is enabled 1 RESET is not asserted on a FMPLL0 fail event
BE_CORE	**Bidirectional Reset Enable for core reset** 0 RESET is asserted on a core reset event if the reset is enabled 1 RESET is not asserted on a core reset event
BE_JTAG	**Bidirectional Reset Enable for JTAG initiated reset** 0 RESET is asserted on a JTAG initiated reset event if the reset is enabled 1 RESET is not asserted on a JTAG initiated reset event

31.4 Functional description

31.4.1 Reset State Machine

The main role of MC_RGM is the generation of the reset sequence which ensures that the correct parts of the device are reset based on the reset source event. This is summarized in Table 31-12.

Table 31-12. MC_RGM Reset Implications

Source	What gets reset	External reset assertion	Boot mode capture
power-on reset	all	yes	yes

Table 31-12. MC_RGM Reset Implications (continued)

'destructive' resets	all except some clock/reset management	yes	yes
external reset	all except some clock/reset management and debug	yes	yes
'functional' resets	all except some clock/reset management and debug	programmable[1]	programmable[2]
shortened 'functional' resets[3]	flip-flops except some clock/reset management	programmable[1]	programmable[2]

1 The assertion of the external reset is controlled via the RGM_FBRE register.

2 The boot mode is captured if the external reset is asserted.

3 The short sequence is enabled via the RGM_FESS register.

NOTE

JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in Figure 31-11.

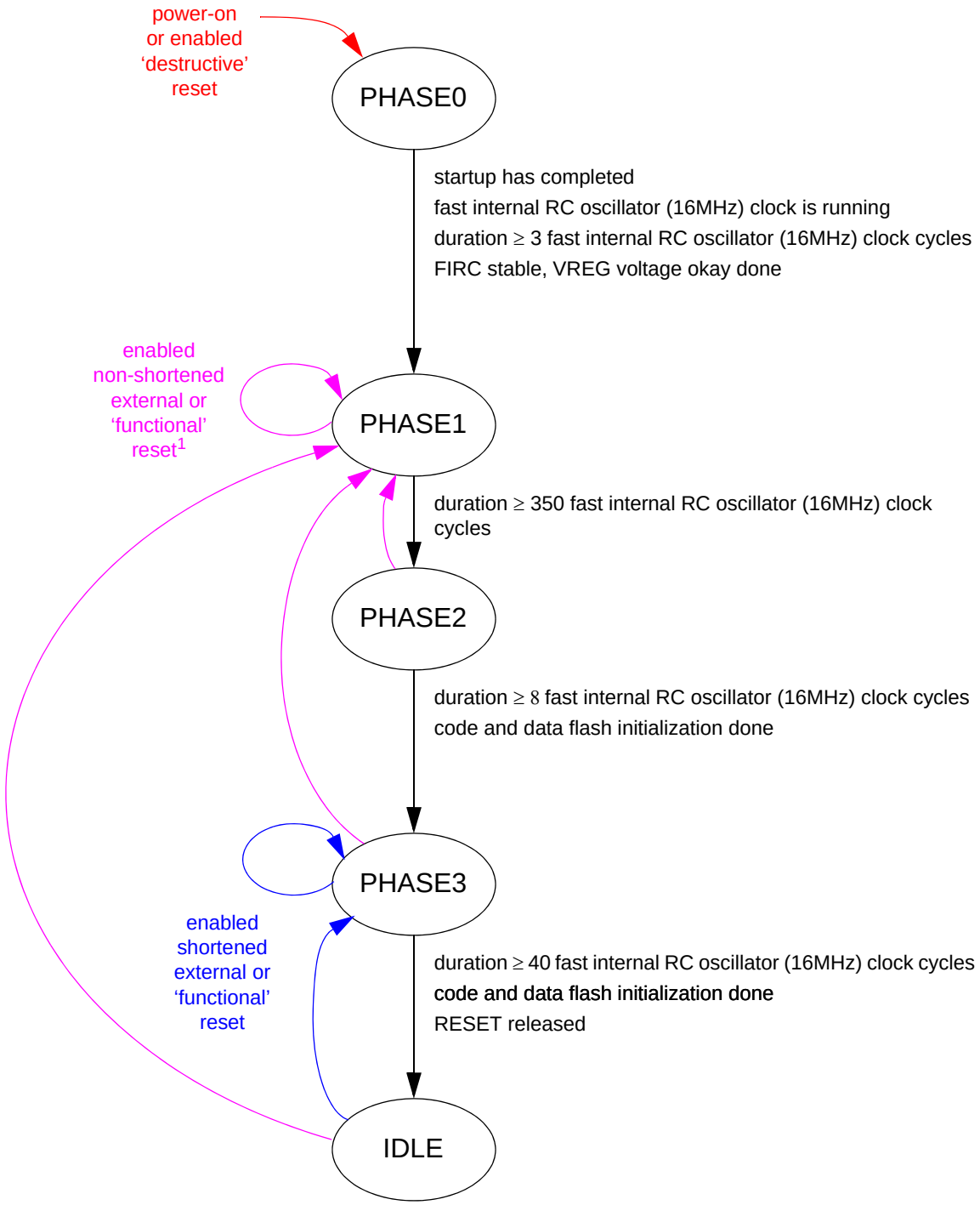


Figure 31-11. MC_RGM State Machine

31.4.1.1 Phase0 Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits Phase0 and enters Phase1 on verification of the following:

- Startup has completed

- fast internal RC oscillator (16MHz) clock is running
- All enabled ‘destructive’ resets have been processed
- All processes that need to be done in Phase0 are completed
  - FIRC stable, VREG voltage okay
- A minimum of 3 fast internal RC oscillator (16MHz) clock cycles have elapsed since startup completion and the last enabled ‘destructive’ reset event

31.4.1.2 Phase1 Phase

This phase is entered either on exit from Phase0 or immediately from Phase2, Phase3, or Idle on a non-masked external or ‘functional’ reset event if it has not been configured to trigger a ‘short’ sequence. The reset state machine exits Phase1 and enters Phase2 on verification of the following:

- All enabled, non-shortened ‘functional’ resets have been processed
- A minimum of 350 fast internal RC oscillator (16MHz) clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

31.4.1.3 Phase2 Phase

This phase is entered on exit from Phase1. The reset state machine exits Phase2 and enters Phase3 on verification of the following:

- All processes that need to be done in Phase2 are completed
  - code and data flash initialization
- A minimum of 8 fast internal RC oscillator (16MHz) clock cycles have elapsed since entering Phase2

31.4.1.4 Phase3 Phase

This phase is a entered either on exit from Phase2 or immediately from Idle on an enabled, shortened ‘functional’ reset event. The reset state machine exits Phase3 and enters Idle on verification of the following:

- All processes that need to be done in Phase3 are completed
  - code and data flash initialization
- A minimum of 40 fast internal RC oscillator (16MHz) clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event

31.4.1.5 Idle Phase

This is the final phase and is entered on exit from Phase3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

31.4.2 Destructive Resets

A ‘destructive’ reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given ‘destructive’ reset event (RGM_DES.F_<*destructive reset*> bit) is set when the ‘destructive’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

The ‘destructive’ reset can be optionally disabled by writing bit RGM_DERD.D_<*destructive reset*>.

NOTE

The RGM_DERD register can be written only once between two power-on reset events.

The device’s low-voltage detector threshold ensures that, when 1.2V low-voltage detected (power domain #0) is enabled, the supply is sufficient to have the destructive event correctly propagated through the digital logic. Therefore, if a given ‘destructive’ reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given ‘destructive’ reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled destructive reset will trigger a reset sequence starting from the beginning of Phase0.

31.4.3 External Reset

The MC_RGM manages the external reset coming from RESET. The detection of a falling edge on RESET will start the reset sequence from the beginning of Phase1.

The status flag associated with the external reset falling edge event (RGM_FES.F_EXR bit) is set when the external reset is asserted and the power-on reset is not asserted.

The external reset can optionally be disabled by writing bit RGM_FERD.D_EXR.

NOTE

The RGM_FERD register can be written only once between two power-on reset events.

An enabled external reset will normally trigger a reset sequence starting from the beginning of Phase1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_EXR is set, the external reset will trigger a reset sequence starting directly from the beginning of Phase3, skipping Phase1 and Phase2. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- A power-on reset
- A ‘destructive’ reset event

- An external reset event
- A ‘functional’ reset event configured via the RGM_FBRE register to assert the external reset

In this case, the external reset is asserted until the end of Phase3.

31.4.4 Functional Resets

A ‘functional’ reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given ‘functional’ reset event (RGM_FES.F_<*functional reset*> bit) is set when the ‘functional’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

The ‘functional’ reset can be optionally disabled by software writing bit RGM_FERD.D_<*functional reset*>.

NOTE

The RGM_FERD register can be written only once between two power-on reset events.

An enabled functional reset will normally trigger a reset sequence starting from the beginning of Phase1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by a functional reset. When RGM_FESS.SS_<*functional reset*> is set, the associated ‘functional’ reset will trigger a reset sequence starting directly from the beginning of Phase3, skipping Phase1 and Phase2. This can be useful especially in case a functional reset should not reset the flash module.

31.4.5 Standby Entry Sequence

Standby mode can be entered only when the MC_RGM is in Idle. On Standby entry, the MC_RGM moves to Phase1. The minimum duration counter in Phase1 does not start until Standby mode is exited. On entry to Phase1 due to Standby mode entry, the resets for all power domains except power domain #0 are asserted. During this time, RESET is not asserted as the external reset can act as a wakeup for the device.

There is an option to keep the flash inaccessible and in low-power mode on Standby exit by configuring the DRUN mode before Standby entry so that the flash is in power-down or low-power mode. If the flash is to be inaccessible, the Phase2 and Phase3 states do not wait for the flash to complete initialization before exiting, and the reset to the flash remains asserted.

See the MC_ME chapter for details on the Standby and DRUN modes.

NOTE

If the device is in Standby mode and an external reset occurs, the MC_RGM may not assert the external reset for the duration of the reset sequence even when RGM_FBRE[BE_EXR] = 0. This incorrect behavior occurs only if the system releases the external reset before the end of reset sequence Phase1.

31.4.6 Alternate Event Generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for each reset source event (except the power-on reset event) to be converted from a reset to either a Safe mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_F/DERD and RGM_F/DEAR registers as shown in Table 31-13.

Table 31-13. MC_RGM alternate event selection

RGM_F/DERD bit value	RGM_F/DEAR bit value	Generated event
0	X	Reset
1	0	Safe mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (i.e. at the reset source that caused the alternate request) and also clearing the appropriate RGM_F/DES status bit.

NOTE

Alternate requests (Safe mode as well as interrupt requests) are generated asynchronously.

NOTE

If a masked ‘destructive’ reset event which is configured to generate a Safe mode/interrupt request occurs during Phase0, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME. The same is true for masked ‘functional’ reset events during Phase1.

31.4.7 Boot mode capturing

The MC_RGM provides sampling of the boot mode PAD[22:21] for use by the system to determine the boot mode. This sampling is done five fast internal RC oscillator (16MHz) clock cycles before the rising edge of RESET. The result of the sampling is then provided to the system. For each bit, a value of 1 is produced only if each of the oldest three of the five samples have the value 1, otherwise a value of 0 is produced.

NOTE

In order to ensure that the boot mode is correctly captured, the application needs to apply the valid boot mode value to the device at least five fast internal RC oscillator (16MHz) clock periods before the external reset deassertion crosses the V_{IH} threshold.

NOTE

RESET can be low as a consequence of the internal reset generation. This will force re-sampling of the boot mode pins.

Chapter 32
Real-Time Clock (RTC/API)

32.1 Overview

The RTC is a free running counter used for timekeeping applications. The RTC may be configured to generate an interrupt at a predefined interval independent of the mode of operation (run mode or low-power mode). If in a low-power mode when the RTC interval is reached, the RTC will first generate a wakeup and then assert the interrupt request. The RTC also supports an autonomous periodic interrupt (API) function used to generate a periodic wakeup request to exit a low-power mode or an interrupt request.

32.2 Features

Features of the RTC include:

- Four selectable counter clock sources
  - 4–16 MHz FXOSC
  - 128 kHz SIRC
  - 32 KHz SXOSC
  - 16 MHz FIRC
- Optional 512 prescaler and optional 32 prescaler
- 32-bit counter
  - supports times up to 1.5 months with 1 ms resolution
  - runs in all modes of operation
  - reset when disabled by software and by POR
- 12-bit compare value to support interrupt intervals of 1 s up to greater than 1 hr with 1 s resolution
- RTC compare value changeable while counter is running
- RTC status and control register are reset only by POR
- Autonomous periodic interrupt (API)
  - 10-bit compare value to support wakeup intervals of 1.0 ms to 1 s
  - compare value changeable while counter is running
- Configurable interrupt for RTC match, API match, and RTC rollover
- Configurable wakeup event for RTC match, API match, and RTC rollover

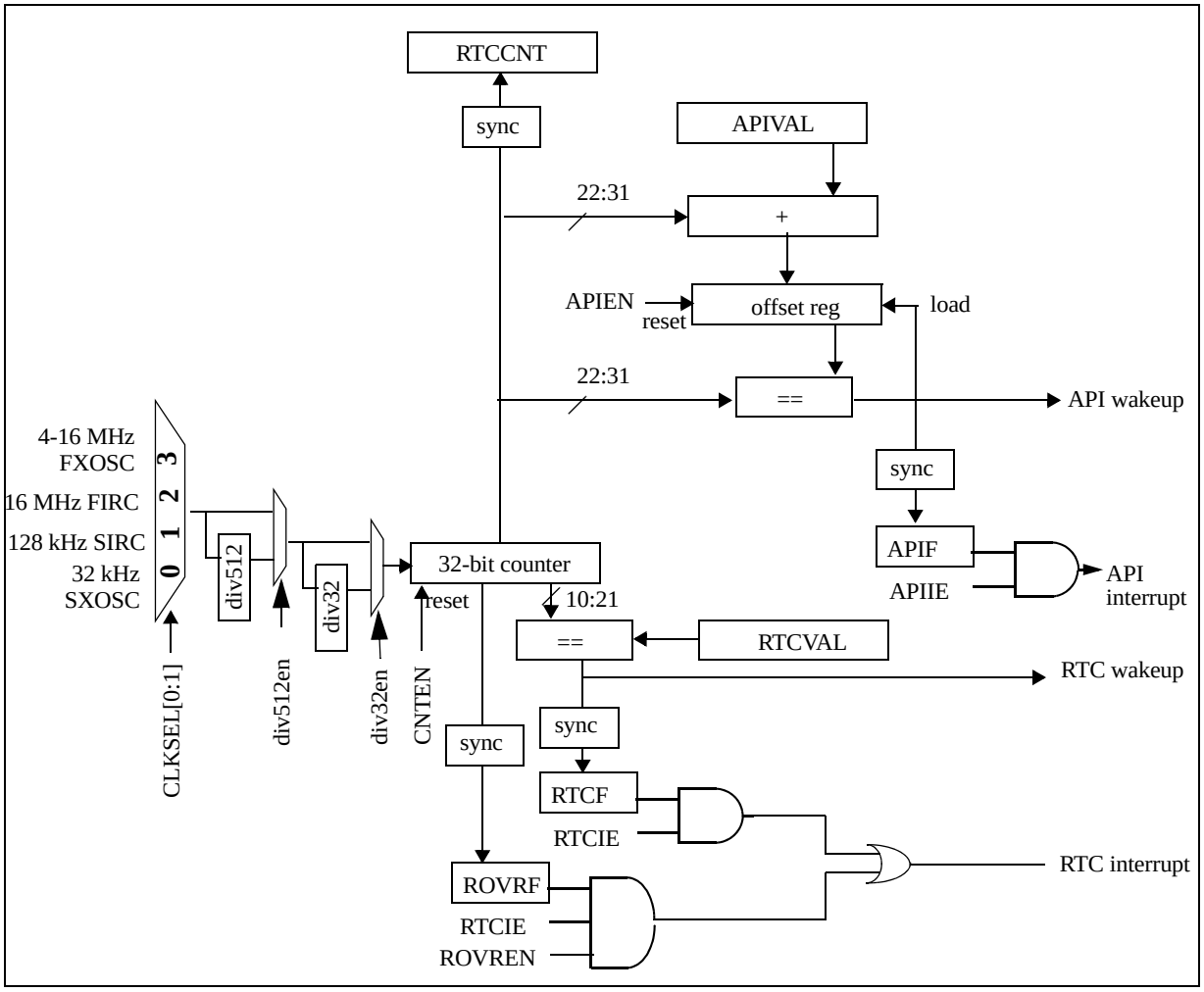


Figure 32-1. RTC/API block diagram

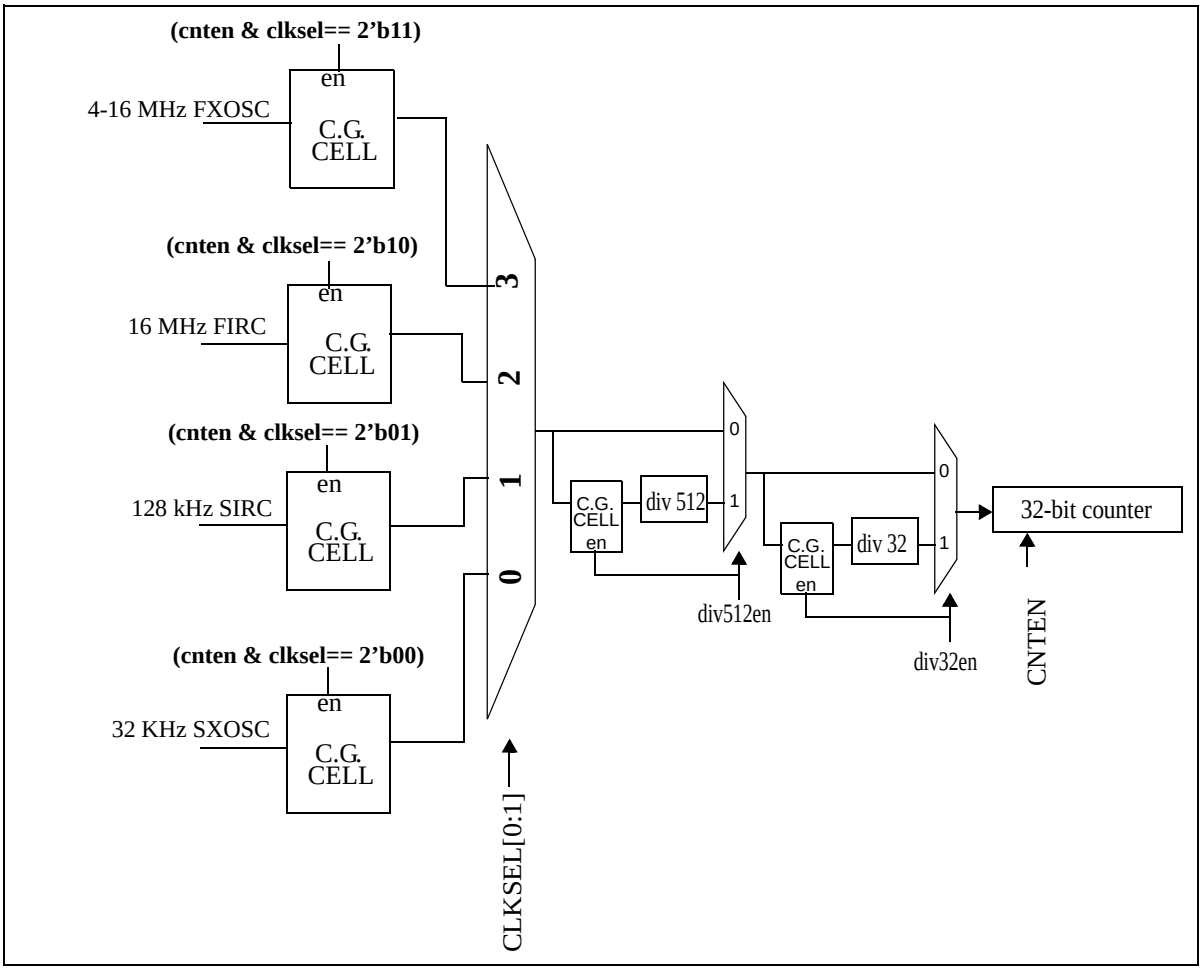


Figure 32-2. Clock gating for RTC clocks

32.3 Device specific information

For this device:

- FXOSC, SXOSC, FIRC and SIRC clocks are provided as counter clocks for the RTC. Default clock on reset is SIRC divided by 4.
- The RTC will be reset on destructive reset, with the exception of software watchdog reset.
- The RTC provides a configurable divider by 512 to be optionally used when FXOSC source is selected.

32.4 Modes of operation

There are two functional modes of operation for the RTC: normal operation and low-power mode. In normal operation, all RTC registers can read or written and the input isolation is disabled. The RTC/API and associated interrupts are optionally enabled. In low-power mode, the bus interface is disabled and the input isolation is enabled. The RTC/API is enabled if enabled prior to entry into low-power mode.

32.5 Debug support

To simplify software development it is possible to temporarily suspend the RTC counter while the MCU is stopped by a debugger. While the MCU is running the RTC counter runs normally. This feature is enabled by setting the RTCC[FRZ] bit and is only available when the CPU has debug mode active (see the CPU reference manual for more information on debug mode and support).

32.6 Register descriptions

32.6.1 RTC Supervisor Control Register (RTCSUPV)

The RTCSUPV register contains the SUPV bit which determines whether other registers are accessible in supervisor mode or user mode.

NOTE

RTCSUPV register is accessible only in supervisor mode.

Figure 32-3. RTC Supervisor Control Register (RTCSUPV)

Offset: RTC_BASE + 0x0000

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

Table 32-1. RTCSUPV Register Bit/field descriptions

Field	Description
0 SUPV	RTC Supervisor Bit 0 All registers are accessible in both user as well as supervisor mode. 1 All other registers are accessible in supervisor mode only.

32.6.2 RTC Control Register (RTCC)

The RTCC register contains:

- RTC counter enable
- RTC interrupt enable
- RTC clock source select
- RTC compare value
- API enable
- API interrupt enable
- API compare value

Figure 32-4. RTC Control Register (RTCC)

Offset RTC_BASE + 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNTEN	RTCIE	FRZEN	ROVREN	RTCVAL											
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APIEN	APIIE	CLKSEL		DIV512EN	DIV32EN	APIVAL									
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-2. RTCC field descriptions

Field	Description
CNTEN	Counter Enable The CNTEN bit enables the RTC counter. Making CNTEN bit 1’b0 has the effect of asynchronously resetting (synchronous reset negation) all the RTC logic. This allows for the RTC configuration and clock source selection to be updated without causing synchronization issues. 1 Counter enabled 0 Counter disabled
RTCIE	RTC Interrupt Enable The RTCIE bit enables interrupts requests to the system if RTCF is asserted. 1 RTC interrupts enabled 0 RTC interrupts disabled
FRZEN	Freeze Enable Bit The counter freezes when the MCU is stopped by a debugger on the last valid count value if the FRZEN bit is set. After coming of the debug mode counter starts from the frozen value. 0 Counter does not freeze in debug mode. 1 Counter freezes in debug mode.

Table 32-2. RTCC field descriptions (continued)

Field	Description
ROVREN	Counter Roll Over Interrupt Enable The ROVREN bit enables interrupt requests when the RTC has rolled over from 0xffff_ffff to 0x0000_0000. The RTCIE bit must also be set in order to generate an interrupt from a counter rollover. 1 RTC rollover interrupt enabled 0 RTC rollover interrupt disabled
RTCVAL	RTC Compare Value The RTCVAL bits are compared to bits 10:21 of the RTC counter and if match sets RTCF. RTCVAL may only be updated when CNTEN is 0. **Note:** RTCVAL should not be set to 0.
APIEN	Autonomous Periodic Interrupt Enable The APIEN bit enables the autonomous periodic interrupt function. 1 API enabled 0 API disabled
APIIE	API Interrupt Enable The APIIE bit enables interrupts requests to the system if APIF is asserted. 1 API interrupts enabled 0 API interrupts disabled
CLKSEL	Clock Select The CLKSEL[0:1] bits select the clock source for the RTC. CLKSEL may only be updated when CNTEN is 0. The user should ensure that oscillator is enabled before selecting it as a clock source for RTC. 00 32 KHz SXOSC 01 128 kHz SIRC 10 16 MHz FIRC 11 16 MHz FXOSC
DIV512EN	Divide by 512 enable The DIV512EN bit enables the 512 clock divider. DIV512EN may only be updated when CNTEN is 0. 0 Divide by 512 is disabled. 1 Divide by 512 is enabled.
DIV32EN	Divide by 32 enable The DIV32EN bit enables the 32 clock divider. DIV32EN may only be updated when CNTEN is 0. 0 Divide by 32 is disabled. 1 Divide by 32 is enabled.
APIVAL	API Compare Value The APIVAL bits are compared to an offset value based on bits 22:31 of the RTC counter and if match asserts an interrupt/wakeup request. APIVAL may only be updated when APIEN is 0 or API function is undefined. **Note:** API functionality starts only when APIVAL is nonzero. The first API interrupt takes two more cycles because of synchronization of APIVAL to the RTC clock. After that, interrupts are periodic in nature. The minimum supported value of APIVAL is 4.

32.6.3 RTC Status Register (RTCS)

The RTCS register contains:

- RTC interrupt flag
- API interrupt flag
- ROLLOVR Flag

Figure 32-5. RTC Status Register (RTCS)

Offset RTC_BASE + 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R			RTCF													
W			w1c													
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			APIF			ROVRF										
W			w1c			w1c										
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-3. RTCS field descriptions

Field	Description
2 RTCF	RTC Interrupt Flag The RTCF bit indicates that the RTC counter has reached the counter value matching RTCVAL. RTCF is cleared by writing a 1 to RTCF. Writing a 0 to RTCF has no effect. 1 RTC interrupt 0 No RTC interrupt **Note:** RTCF is not set if RTCVAL is 12'b0.
18 APIF	API Interrupt Flag The APIF bit indicates that the RTC counter has reached the counter value matching API offset value. APIF is cleared by writing a 1 to APIF. Writing a 0 to APIF has no effect. 1 API interrupt 0 No API interrupt Note: The periodic interrupt comes after APIVAL[0:9] + 1'b1 RTC counts
21 ROVRF	Counter Roll Over Interrupt Flag The ROVRF bit indicates that the RTC has rolled over from 0xffff_ffff to 0x0000_0000. ROVRF is cleared by writing a 1 to ROVRF. 1 RTC has rolled over. 0 RTC has not rolled over.

32.6.4 RTC Counter Register (RTCCNT)

The RTCCNT register contains the current value of the RTC counter.

Figure 32-6. RTC Counter Register (RTCCNT)

Offset: RTC_BASE + 0x000C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTCCNT[0:31]																															
W																																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-4. RTCCNT Register Bit/field descriptions

Field	Description
0:31 RTCCNT[0:31]	RTC Counter Value Due to the clock synchronization, the RTCCNT value may actually represent a previous counter value.

32.7 RTC functional description

The RTC consists of a 32-bit free running counter enabled with the RTCC[CNTEN] bit (CNTEN when negated asynchronously resets the counter and synchronously enables the counter when enabled). The value of the counter may be read via the RTCCNT register. Note that due to the clock synchronization, the RTCCNT value may actually represent a previous counter value. The difference between the counter and the read value depends on ratio of counter clock and ipg_clk. Maximum possible difference between the two is 6 count values.

The clock source to the counter is selected with the RTCC[CLKSEL] field, which gives four options for clocking the RTC/API. The four clock sources are assumed to be two 16 MHz sources, one 32 KHz source and one 128 kHz source. The output of the clock mux can be optionally divided by combination of 512 and 32 to give a 1 ms RTC/API count period for different clock sources. Note that the RTCC[CNTEN] bit must be disabled when the RTC/API clock source is switched.

When the counter value for counter bits 10:21 match the 12-bit value in the RTCC[RTCVAL] field, then the RTCS[RTCF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[RTCIE] interrupt enable bit is set, then the RTC interrupt request is generated. The RTC supports interrupt requests in the range of 1s to 4096s (> 1 hr.) with a 1s resolution. The RTCC[RTCVAL] field may only be updated when the RTCC[CNTEN] bit is cleared to disable the counter. If there is a match while in low-power mode then the RTC will first generate a wakeup request to force a wakeup to run mode, then the RTCF flag will be set. RTCC[RTCVAL]=0x000 is invalid.

A rollover interrupt can be generated when the RTC transitions from a count of 0xFFFF_FFFF to 0x0000_0000. The rollover flag is enabled by setting the RTCC[ROVREN] bit. An interrupt request is generated for an RTC counter rollover when both the RTCC[ROVREN] and RTCC[RTCIE] bits are set.

All the flags and counter values are synchronized with ipg_clk. It is assumed that ipg_clk frequency is always more than or equal to the rtc_clk used to run the counter.

32.8 API functional description

Setting RTCC[APIEN] bit enables the autonomous interrupt function. The 10-bit RTCC[APIVAL] field selects the time interval for triggering an interrupt and/or wakeup event. Since the RTC is a free running counter, the APIVAL is added to the current count to calculate an offset. When the counter reaches (offset count + 1), a interrupt and/or wakeup request is generated. Then the offset value is recalculated and again retriggers a new request when the new value is reached. APIVAL may only be updated when APIEN is disabled. When a compare is reached, the RTCS[APIF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[APIIE] interrupt enable bit is set, then the API interrupt request is generated. If there is a match while in low-power mode, then the API will first generate a wakeup request to force a wakeup into normal operation, then the APIF flag will be set.

Chapter 33
Static RAM (SRAM)

33.1 Introduction

The MPC5606S family offers 24 KB or 48 KB of general purpose system SRAM (with ECC protection). Some family members also include 160 KB of graphics SRAM (without ECC protection). Details of the SRAM areas and their features for each family member are shown in

Table 33-1. SRAM memory map

Address range		Size [KB]	5602S	5604S	5606S	Region
0x40000000	0x40001FFF	8	Yes	Yes	Yes	SRAM (ECC protection, standby support)
0x40002000	0x40005FFF	16	Yes	Yes	Yes	SRAM (ECC protection, standby support is programmable)
0x40006000	0x4000BFFF	24	No	Yes	Yes	SRAM (ECC protection, standby support is programmable)
0x4000C000	0x5FFFFFFF	524240	—	—	—	Reserved
0x60000000	0x60027FFF	160	No	No	Yes	Graphics SRAM (no ECC protection, no standby support)
0x60028000	0x7FFFFFFF	524128	—	—	—	Reserved

33.2 General-purpose SRAM

The general-purpose SRAM comprises 8 KB of standby SRAM plus 16 KB or 40 KB of SRAM that may or may not be enabled in standby mode. The SRAM provides the following features:

- SRAM can be read/written from any bus master
- Byte, halfword and word addressable
- ECC protected with single-bit correction and double-bit detection

33.3 Graphics SRAM

The 160 KB of graphics SRAM has no ECC protection and is not powered during standby mode.

33.4 Low-power configuration

In order to reduce leakage, only an 8 KB portion of the SRAM remains powered by default during standby mode.

Table 33-2. Low-power configuration

Mode	Configuration
Run, Test, Safe and Stop	All general purpose and graphics SRAM is powered and operational.
Standby (1)	Only 8 KB of the SRAM remains powered. Upper RAM is disabled by the MC_PCU.
Standby (2)	All system SRAM remains powered. Upper RAM is enabled by the MC_PCU.

33.5 Register memory map

The internal SRAM has no registers. Registers for the SRAM ECC are located in the ECSM.

33.6 SRAM ECC mechanism

The SRAM ECC detects the following conditions and produces the following results:

- Detects and corrects all 1-bit errors
- Detects and flags all 2-bit errors as non-correctable errors
- Detects 39-bit reads (32-bit data bus plus the 7-bit ECC) that return all zeros or all ones, asserts an error indicator on the bus cycle, and sets the error flag

SRAM does not detect all errors greater than 2 bits.

Internal SRAM write operations are performed on the following byte boundaries:

- 1 byte (0:7 bits)
- 2 bytes (0:15 bits)
- 4 bytes or 1 word (0:31 bits)

If the entire 32 data bits are written to SRAM, no read operation is performed and the ECC is calculated across the 32-bit data bus. The 8-bit ECC is appended to the data segment and written to SRAM.

If the write operation is less than the entire 32-bit data width (1-, or 2-byte segment), the following occurs:

1. The ECC mechanism checks the entire 32-bit data bus for errors, detecting and either correcting or flagging errors.
2. The write data bytes (1-, or 2-byte segment) are merged with the corrected 32 bits on the data bus.
3. The ECC is then calculated on the resulting 32 bits formed in the previous step.
4. The 7-bit ECC result is appended to the 32 bits from the data bus, and the 39-bit value is then written to SRAM.

33.6.1 Access timing

The system bus is a two-stage pipelined bus, which makes the timing of any access dependent on the access during the previous clock. Table 33-3 lists the various combinations of read and write operations to SRAM and the number of wait states used for the each operation. The table columns contain the following information:

- Current operation: Lists the type of SRAM operation executing currently

- Previous operation: Lists the valid types of SRAM operations that can precede the current SRAM operation (valid operation during the preceding clock)
- Wait states: Lists the number of wait states (bus clocks) the operation requires which depends on the combination of the current and previous operation

Table 33-3. Number of wait states required for SRAM operations

	Current Operation	Previous Operation	Number of Wait States Required
Read Operation	Read	Idle	1
		Pipelined read	
		8-, 16-, or 32-bit write	0 (read from the same address)
			1 (read from a different address)
	Pipelined read	Read	0
Write Operation	8-, or 16-bit write	Idle	1
		Read	
		Pipelined 8-, or 16-bit write	2
		32-bit write	
		8-, or 16-bit write	0 (write to the same address)
	Pipelined 8-, 16-, or 32-bit write	8-, 16-, or 32-bit write	0
	32-bit write	Idle	0
		32-bit write	
		Read	

33.6.2 Reset effects on SRAM accesses

Asynchronous reset will possibly corrupt RAM if it asserts during a read or write operation to SRAM. The completion of that access depends on the cycle at which the reset occurs. Data read from or written to SRAM before the reset event occurred is retained, and no other address locations are accessed or changed. In case of no access ongoing when reset occurs, the RAM corruption does not happen.

Instead synchronous reset (SW reset) should be used in controlled function (without RAM accesses) in case initialization procedure is needed without RAM initialization.

33.7 Functional description

ECC checks are performed during the read portion of an SRAM ECC read/write (R/W) operation, and ECC calculations are performed during the write portion of a read/write (R/W) operation. Because the ECC bits can contain random data after the device is powered on, the SRAM must be initialized by

executing 32-bit write operations prior any read accesses. This is also true for implicit read accesses caused by any write accesses of less than 32-bit as discussed in Section 33.6, SRAM ECC mechanism.

33.8 Initialization and application information

To use the SRAM, the ECC must check all bits that require initialization after power on. All writes must specify an even number of registers performed on 32-bit word-aligned boundaries. If the write is not the entire 32 bits (8 or 16 bits), a read / modify / write operation is generated that checks the ECC value upon the read.

Chapter 34
Sound Generation Logic (SGL)

34.1 Introduction

This chapter describes the Sound Generation Logic (SGL) module.

Refer to Figure 34-1 for the detailed block diagram of the SGL.

34.1.1 Overview

The SGL provides a single output to the speaker or buzzer interface.

It selects outputs from the PWM channels being used for sound generation. In case of monophonic sound, two PWM channels are required while, in the case of polyphonic sound, only one PWM channel is required. Monophonic / polyphonic sound can be selected through the signal mono/poly_b.

A 32-bit counter value determines the duration for which sound will be played.

NOTE

In this document, the term “polyphonic” refers to PCM-based sound generation as described in Section 34.4, Functional description.

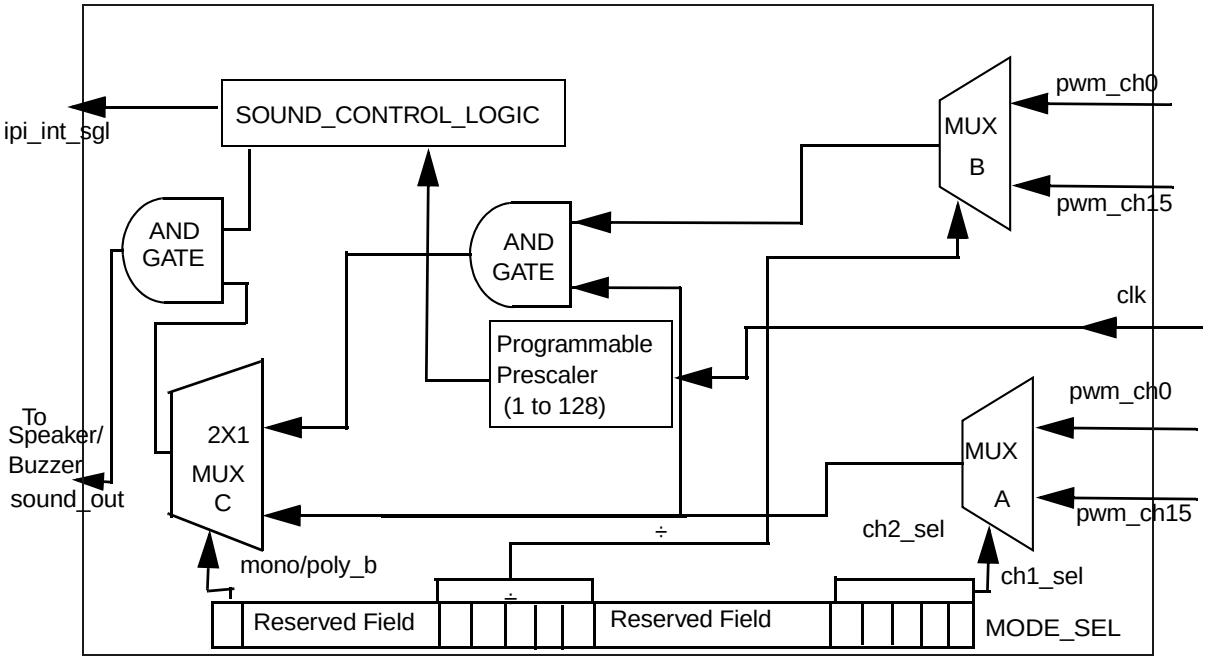


Figure 34-1. Block diagram

34.1.2 Features

The SGL has the following basic features:

- It can be used to produce two types of sounds — monophonic and polyphonic sound.
- It selects the PWM channels being used for sound generation and provides the PWM output from single channel or anded output from 2 PWM channels depending on whether polyphonic or monophonic sound is to be produced.
- It provides programmability to generate periodic or continuous sound, for desired duration which may be predefined.
- It has the facility for enabling/disabling the generation of interrupts.

34.2 External signal description

Figure 34-2. SGL External signal description

34.2.1 Detailed signal descriptions

Table 34-1. Detailed signal descriptions

Name	Function	I/O	Reset[1]
Clock	System clock	I	
pwm_ch0 to pwm_ch15	PWM Channel Output [15:0], • Signals coming from eMIOS0[23:16] correspond to pwm_ch[7:0] • Signals coming from eMIOS1[23:16] correspond to pwm_ch[15:8] Signals can have variable duty cycles as well as variable frequencies. See Table 34-4 for more information.	I	
sound_out	Output signal carrying the generated sound that is fed into the speaker interface.	O	0
ipi_int_sgl	Interrupt signal generated by SGL whenever the SOUND_DURATION counter reaches 0	O	0

1 This is the value that the output signals will have when the module is reset.

34.3 Memory map and register definition

34.3.1 Memory map

Table 34-2. SGL memory map

Address Offset	Register	Access	Reset Value
0x00	MODE_SEL register	R/W[1]	0x0000_0000
0X04	SOUND_DURATION register	R/W	0x0000_0000
0X08	HIGH_PERIOD register	R/W	0x0000_0000
0X0C	LOW_PERIOD register	R/W	0x0000_0000
0X10	SGL_STATUS register	R	0x00

[1] Note that read/write registers may contain some read-only or write-only bits.

34.3.2 Register descriptions

34.3.2.1 MODE_SEL register

Offset 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M_P	SOUND_CTRL			SDCIE								CH2_SEL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRE												CH1_SEL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-3. MODE_SEL REGISTER

Table 34-3. MODE_SEL field descriptions

Field	Description
0 M_P	Selects output corresponding to monophonic or polyphonic sound. 1 Output from and gate will be selected to produce monophonic sound 0 Output from mux A will be selected to produce polyphonic sound
1–3 SOUND_CTRL	Sound Control bits. See Table 34-10 for a detailed description.

Table 34-3. MODE_SEL field descriptions (continued)

Field	Description
4 SDCIE	Sound Duration Complete Interrrupt Enable Bit. Enables or disables the interrupt ipi_int_sgl output signal. 1 Interrupt ipi_int_sgl is enabled 0 Interrupt ipi_int_sgl is not enabled
12–15 CH2_SEL	PWM channel select (Mux B). Used to select specific PWM channel to be used by Mux B for sound generation. See Table 34-4 for a detailed description.
16–22 PRE	Clock divider value for the prescaler. See Table 34-5 for a detailed description.
28–31 CH1_SEL	PWM channel select (Mux A). Used to select specific PWM channel to be used by Mux A for sound generation. See Table 34-4 for a detailed description.

Table 34-4. eMIOS channel mapping

CHx_SEL	eMIOS channel selected	CHx_SEL	eMIOS channel selected
0x00	eMIOS_0 Channel 16	0x08	eMIOS_1 Channel 16
0x01	eMIOS_0 Channel 17	0x09	eMIOS_1 Channel 17
0x02	eMIOS_0 Channel 18	0x0A	eMIOS_1 Channel 18
0x03	eMIOS_0 Channel 19	0x0B	eMIOS_1 Channel 19
0x04	eMIOS_0 Channel 20	0x0C	eMIOS_1 Channel 20
0x05	eMIOS_0 Channel 21	0x0D	eMIOS_1 Channel 21
0x06	eMIOS_0 Channel 22	0x0E	eMIOS_1 Channel 22
0x07	eMIOS_0 Channel 23	0x0F	eMIOS_1 Channel 23

Table 34-5. Prescaler clock divider

PRE	Divide ratio
0000000	1
0000001	2
0000010	3
0000011–1111110	4–127
1111111	128

34.3.2.2 SOUND_DURATION register

Offset 0x04 Access: User read/write

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

Figure 34-4. SOUND_DURATION Register

Table 34-6. SOUND_DURATION field descriptions

Field	Description
0–31 SOUND_DURATION	32-bit value which is loaded into bits [0:31] of SOUND_DURATION counter

34.3.2.3 HIGH_PERIOD register

Offset 0x08 Access: User read/write

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

Figure 34-5. HIGH_PERIOD Register

Table 34-7. HIGH_PERIOD field descriptions

Field	Description
0–31 HIGH_PERIOD	32-bit value loaded into bits [0:31] of the HIGH_PERIOD counter. It defines the on period for sound when operating in Periodic mode (that is, when bit 2 of MODE_SEL[SOUND_CTRL] is 0b1).

34.3.2.4 LOW_PERIOD register

Offset 0x0C Access: User read/write

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

Figure 34-6. LOW_PERIOD Register

Table 34-8. LOW_PERIOD field descriptions

Field	Description
0–31 LOW_PERIOD	32-bit value loaded into bits [0:31] of the LOW_PERIOD counter. It defines the off period for sound when operating in Periodic mode (that is, when bit 2 of MODE_SEL[SOUND_CTRL] is 0b1).

34.3.2.5 SGL_STATUS register

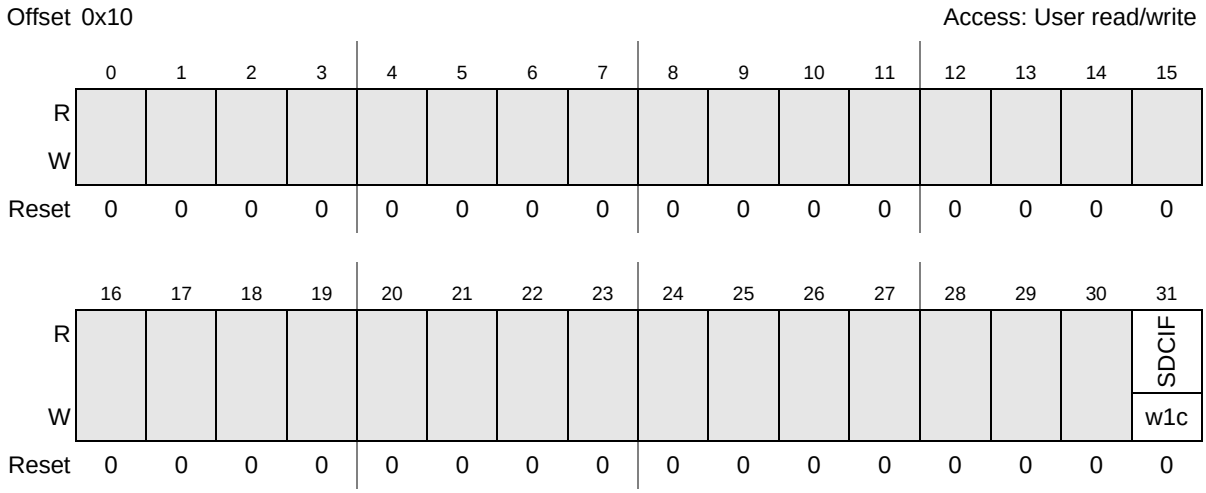


Figure 34-7. SGL_STATUS register

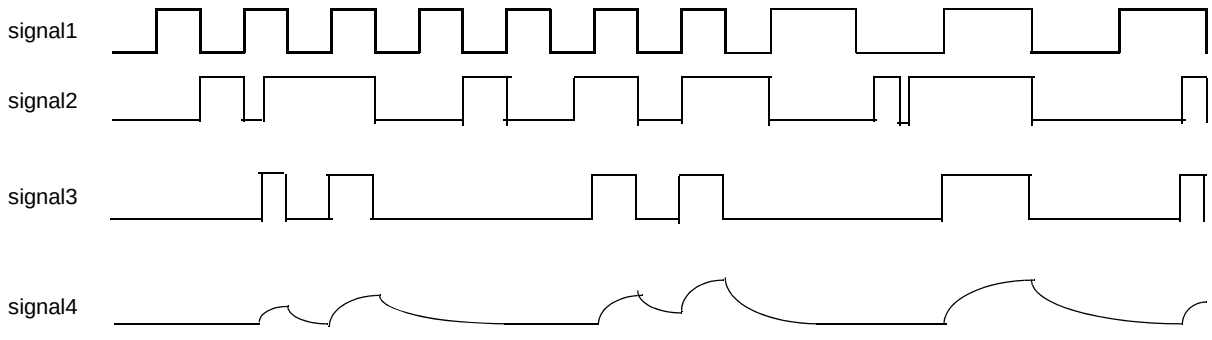
Table 34-9. SGL_STATUS field descriptions

Field	Description
SDCIF	Sound Duration Complete Interrrupt Flag bit. Reflects the status of the interrupt. 0 SDCIF cleared 1 SDCIF set event has occurred due to generation of interrupt

34.4 Functional description

The SGL can be used to produce two types of sounds as described below:

- Monophonic sound
  The tone amplitude modulation is based on two different mixed signals.The first of these two signals has a variable frequency and fixed duty cycle (signal1 in Figure 34-8). The second signal has a fixed frequency and a variable pulse width for generation of the amplitude (signal2 in Figure 34-8). The duty cycle of this signal represents the amplitude of the generated sound. The sound generator is generally a PWM that generates 2 different signals and a mixer to generate a tone at the output as shown in signal 3 of Figure 34-8. It is passed through a first-order low-pass filter to produce signal4 in Figure 34-8. This signal can be fed to the speaker interface.



signal1: Signal with fixed duty cycle and variable frequency
signal2: TONE Signal with fixed frequency (Refer to falling edges) and variable duty cycle
signal3: Anded output (sound_out) of Signal1 and Signal2 coming from the mixer
signal4: sound_out after passing through the low pass filter (external to the SGL)

Figure 34-8. Monophonic sound generation

- Polyphonic sound
  For polyphonic sound, PCM coded samples which reside in system memory, are loaded into the PWM registers at the PCM sample frequency (for example, 16 kHz). So, for a given time period (1/16 kHz), the PWM output has a fixed duty cycle and frequency. A new sample is loaded into the PWM register (at the rate of 16 kHz), which results in a corresponding change in the duty cycle and frequency (both of which will remain fixed until the loading of new PCM coded sample). The resulting PWM signal gives polyphonic sound as an output. The output must be filtered by a low-pass filter. The quality and the order of the low-pass filter will have a direct impact on the quality of the produced sound.

The SGL takes outputs of the 16 PWM channels being used for sound generation as inputs and processes them accordingly, depending on whether monophonic or polyphonic sound is to be generated, to give sound_out as the output signal.

The select lines for multiplexers come from a register (MODE_SEL) which is configured by software.

The type of sound output (monophonic or polyphonic) is controlled by MODE_SEL[M_P]. The value of MODE_SEL[SOUND_CTRL] determines the duration of sound and whether the sound output is periodic or continuous.

Each duration register (HIGH_PERIOD, LOW_PERIOD and SOUND_DURATION) has a total size of 32 bits. The clock on which their counters operate is provided by a programmable prescaler (MODE_SEL[PRE] can be configured to divide clk by values as shown in Figure 34-1), so that the duration for which sound can be produced becomes significant.

The fixed prescaler brings down the time base at 64 MHz system clock frequency to 1 microsecond, which can be further prescaled by using the programmable prescaler.

The HIGH_PERIOD, LOW_PERIOD and SOUND_DURATION registers must be reprogrammed each time sound needs to be generated, with a new configuration of MODE_SEL[SOUND_CTRL].

See Table 34-10 for a detailed description of how the SOUND_CTRL bits affect the generated sound.

Table 34-10. Truth table for MODE_SEL[SOUND_CTRL]

SOUND_ CTRL[2] (Periodic/ Continuous)	SOUND_ CTRL[1] (Begin/End)	SOUND_ CTRL[0] (Counter Enable)	Result
X	0	0	No sound generation
0	0	1	Continuous sound is generated until the counter loaded with the value in the SOUND_DURATION register reaches 0. There is no dependency on HIGH_PERIOD and LOW_PERIOD register values.
0	1	X	Continuous sound is generated. There is no dependency on SOUND_DURATION, HIGH_PERIOD, or LOW_PERIOD register values.
1	0	1	Sound High (i.e. Sound is on) and Sound Low (i.e. sound is off) are generated until the counter loaded with the value in the SOUND_DURATION register reaches 0. Duration for Sound High and Sound Low depend on HIGH_PERIOD and LOW_PERIOD_MSB Register values respectively. The HIGH_PERIOD counter and LOW_PERIOD counter are reloaded automatically as long as the SOUND_DURATION counter does not reach 0.
1	1	X	Sound High (i.e. Sound is on) and Sound Low (i.e. sound is off) are generated. Duration for Sound High and Sound Low depend on HIGH_PERIOD and LOW_PERIOD Register values respectively. The HIGH_PERIOD counter and LOW_PERIOD counter are reloaded automatically. There is no dependency on the value in the SOUND_DURATION register in this mode.

In case the user needs to control the sound duration through software, then SOUND_CTRL[1] needs to be set to 1. SOUND_CTRL[1] can be forced to 0 at any point of time by software through the IPS interface if the sound generation needs to be stopped. This feature handles the case in which sound generation needs to be stopped after a time that cannot be predetermined. An example of this is the warning sound that should be generated when a car door is left open, which should stop as soon as the car door is closed. The sound generation continues as long as SOUND_CTRL[1] is 1 (that is, when the car door is open) and stops as soon as it is set to zero (when the car door is closed).

NOTE

While the sound is active, it is not possible to stop the sound immediately by setting SOUND_CTR = 0 (no sound generation). The current programmed duration of the sound must complete before the sound can be stopped.

If a continuous sound is selected, then set SOUND_DURATION = 0, and select SOUND_CTRL = 5 (single duration sound). If SOUND_DURATION is previously configured, then the sound will stop when the value in SOUND_DURATION counts down to 0. If required, it is possible to set SOUND_CONTROL = 0 once the sound has stopped.

If a non-continuous sound is selected, then wait until the sound stops and then set SOUND_CTRL = 0.

It is advisable to choose the values for HIGH_PERIOD, LOW_PERIOD and SOUND_DURATION in such a way that SOUND_DURATION is an integral multiple of the sum of LOW_PERIOD and HIGH_PERIOD.

The HIGH_PERIOD and LOW_PERIOD registers can also be used as a volume control for monophonic sound when SOUND_CTRL[2] is 1.

In case where monophonic sound is to be generated, the software configures the MODE_SEL register through IPS interface such that the set of 4 bits (CH1_SEL[0:3]) can be used to select output from one of the PWM channels being used for sound generation. This output will be the one having variable duty cycle and fixed frequency.

Similarly, the 4 bits configured by the software comprise the signal ch2_sel[0:3], thus leading to selection of output from the second PWM channel being used for sound generation. This output from second channel will be the one having fixed duty cycle and variable frequency.

Simultaneously, mono/poly_b is configured such that for monophonic sound generation its value will be one, and hence the output from the and gate will be selected i.e. the ANDing of signals from MuxA (variable duty cycle, fixed frequency) and MuxB (fixed duty cycle, variable frequency) will produce monophonic sound.

In case where polyphonic sound is to be generated, the software configures the MODE_SEL register through the IPS interface such that the set of 4 bits (CH1_SEL[0:3]) can be used to select output from the PWM channel being used for sound generation.

Simultaneously, mono/poly_b is configured such that for polyphonic sound generation its value will be zero, and hence the output corresponding to the signal producing polyphonic sound will be selected.

The suggested sequence of events for sound generation controlled by SOUND_CTRL[0:2] is:

1. Initially SOUND_CTRL[0:2] is 3’b000.
2. Program the SOUND_DURATION/ HIGH_PERIOD/LOW_PERIOD registers’ values, depending on the mode of sound generation which is desired.
3. Program the SOUND_CTRL[0:2] bits to the new value depending on the mode desired.
4. Edge detector detects the change on SOUND_CTRL[0:2] bits.
5. On the next clock edge after the edge detection, the sound generation starts.
6. In cases where the sound duration depends on the value in SOUND_DURATION register, the sound generation stops when SOUND_DURATION counter reaches zero. An interrupt is raised as soon as the SOUND_DURATION counter reaches zero. In these cases, the ISR changes the SOUND_CTRL[0:2] to 3’b000 from the present value.

In other cases, where SOUND_DURATION counter has no role to play, the sound generation stops when the SOUND_CTRL[0:2] bits are changed asynchronously by the software.

34.4.1 Interrupts

SGL generates the interrupt signal ipi_int_sgl. The generation of this interrupt is enabled if the MODE_SEL[SDCIE] bit is 1. If this bit is 0, then the interrupt is disabled.

The interrupt is generated whenever the SOUND_DURATION counter reaches 0 in cases where sound is being generated for a predetermined duration of time (i.e., SOUND_DURATION register value is controlling sound duration).

The SGL_STATUS[SDCIF] bit indicates the generation of interrupt. It is set whenever an interrupt is generated irrespective of the value of the SDCIE bit. It can be cleared by writing 1 to it.

The interrupt signal ipi_int_sgl is asserted if the SDCIF flag is set and the SDCIE bit is set.

Chapter 35
Stepper Motor Controller (SMC)

35.1 Introduction

The SMC block is a PWM motor controller suitable for driving small stepper and air core motors used in instrumentation applications. The module can also be used for other motor control or PWM applications that match the frequency, resolution and output drive capabilities of the module. The SMC has 12 PWM channels associated with two pins each (24 pins in total).

35.1.1 Features

The SMC includes the following features:

- 10/11-bit PWM counter
- 11-bit resolution with selectable PWM dithering function
- Left, right, or center aligned PWM
- Short-circuit detection in each PWM channel with programmable timeout

35.1.2 Modes of operation

35.1.2.1 Functional modes

35.1.2.1.1 Dither Function

Dither function can be selected or deselected by setting or clearing the MCCTL0[DITH] bit. This bit influences all PWM channels. For details, please refer to Section 35.4.1.3.5, Dither Bit (MCCTL0[DITH]).

35.1.2.2 PWM channel configuration modes

The 12 PWM channels can operate in three functional modes. Those modes are, with some restrictions, selectable for each channel independently.

35.1.2.2.1 Dual full H-bridge mode

This mode is suitable to drive a stepper motor or a 360^{o} air gauge instrument. For details, please refer to Section 35.4.1.1.1, Dual full H-bridge mode. In this mode two adjacent PWM channels are combined, and two PWM channels drive four pins.

35.1.2.2.2 Full H-bridge mode

This mode is suitable to drive any load requiring a PWM signal in a H-bridge configuration using two pins. For details please refer to Section 35.4.1.1.2, Full H-bridge mode.

35.1.2.2.3 Half H-bridge mode

This mode is suitable to drive a 90^{o} instrument driven by one pin. For details, please refer to Section 35.4.1.1.3, Half H-bridge mode.

35.1.2.3 PWM alignment modes

Each PWM channel can operate independently in three different alignment modes. For details, please refer to Section 35.4.1.3.1, PWM alignment modes.

35.1.2.4 Low-power mode

The behavior of the SMC when it is disabled by the MC_ME module is programmable. For details, please refer to Section 35.4.5, Operation in SMC stop mode.

35.1.3 Block diagram

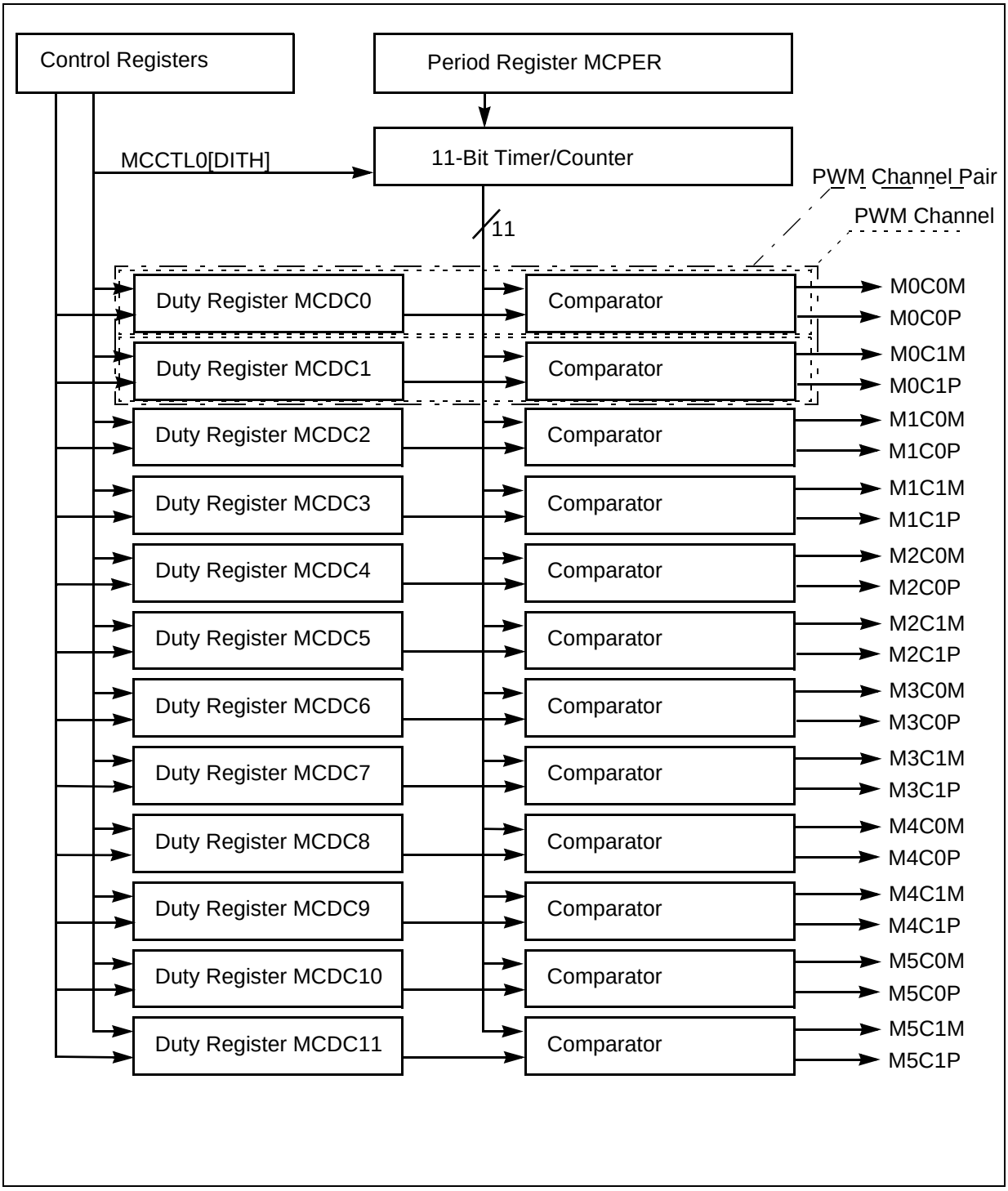


Figure 35-1. SMC block diagram

35.2 External signal description

The SMC is associated with 24 pins. Table 35-1 lists the relationship between the PWM channels, signal pins, PWM channel pairs (motor numbers), coils and nodes they are supposed to drive if all channels are set to dual full H-bridge configuration.

Table 35-1. PWM Channel and Pin Assignment

Pin Name	PWM Channel	PWM Channel Pair[1]	Coil	Node
M0C0M	0	0	0	Minus
M0C0P				Plus
M0C1M	1		1	Minus
M0C1P				Plus
M1C0M	2	1	0	Minus
M1C0P				Plus
M1C1M	3		1	Minus
M1C1P				Plus
M2C0M	4	2	0	Minus
M2C0P				Plus
M2C1M	5		1	Minus
M2C1P				Plus
M3C0M	6	3	0	Minus
M3C0P				Plus
M3C1M	7		1	Minus
M3C1P				Plus
M4C0M	8	4	0	Minus
M4C0P				Plus
M4C1M	9		1	Minus
M4C1P				Plus
M5C0M	10	5	0	Minus
M5C0P				Plus
M5C1M	11		1	Minus
M5C1P				Plus

1 A PWM Channel Pair always consists of PWM channel x and PWM channel x+1 (x = 2·n). The term “PWM Channel Pair” is equivalent to the term “Motor”. For example, Channel Pair 0 is equivalent to Motor 0.

35.2.1 M0C0M/M0C0P/M0C1M/M0C1P — PWM output pins for Motor 0

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 0. PWM output on M0C0M results in a positive current flow through coil 0 when M0C0P is driven to a logic high state. PWM output on M0C1M results in a positive current flow through coil 1 when M0C1P is driven to a logic high state (for details refer to Section 35.4.1, Modes of operation).

35.2.2 M1C0M/M1C0P/M1C1M/M1C1P — PWM output pins for Motor 1

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 1. PWM output on M1C0M results in a positive current flow through coil 0 when M1C0P is driven to a logic high state. PWM output on M1C1M results in a positive current flow through coil 1 when M1C1P is driven to a logic high state (for details refer to Section 35.4.1, Modes of operation).

35.2.3 M2C0M/M2C0P/M2C1M/M2C1P — PWM output pins for Motor 2

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 2. PWM output on M2C0M results in a positive current flow through coil 0 when M2C0P is driven to a logic high state. PWM output on M2C1M results in a positive current flow through coil 1 when M2C1P is driven to a logic high state (for details refer to Section 35.4.1, Modes of operation).

35.2.4 M3C0M/M3C0P/M3C1M/M3C1P — PWM output pins for Motor 3

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 3. PWM output on M3C0M results in a positive current flow through coil 0 when M3C0P is driven to a logic high state. PWM output on M3C1M results in a positive current flow through coil 1 when M3C1P is driven to a logic high state (for details refer to Section 35.4.1, Modes of operation).

35.2.5 M4C0M/M4C0P/M4C1M/M4C1P — PWM output pins for Motor 4

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 4. PWM output on M4C0M results in a positive current flow through coil 0 when M4C0P is driven to a logic high state. PWM output on M4C1M results in a positive current flow through coil 1 when M4C1P is driven to a logic high state (for details refer to Section 35.4.1, Modes of operation).

35.2.6 M5C0M/M5C0P/M5C1M/M5C1P — PWM output pins for Motor 5

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 5. PWM output on M5C0M results in a positive current flow through coil 0 when M5C0P is driven to a logic high state. PWM output on M5C1M results in a positive current flow through coil 1 when M5C1P is driven to a logic high state (for details refer to Section 35.4.1, Modes of operation).

35.3 Memory map and register definition

This section provides a detailed description of all registers of the 10-bit 12-channel SMC module.

35.3.1 Module memory map

Table 35-2 shows the memory map of the 10-bit 12-channel SMC module.

Access type can be

- RW: Read and Write

- Data access type is 8-, 16-, or 32-bit. It is recommended to access the various register using the access types shown in Table 35-2 for consistent write operations.

Table 35-2. SMC memory map

Address offset	Use	Recommended Access type	Location
0x00	Motor Controller Control Register 0 (MCCTL0)	RW, 8-bit	on page 1126
0x01	Motor Controller Control Register 1 (MCCTL1)	RW, 8-bit	on page 1127
0x02	Motor Controller Period Register (MCPER)	RW, 16-bit	on page 1128
0x04	Reserved	—	—
0x05	Reserved	—	—
0x06	Reserved	—	—
0x07	Reserved	—	—
0x08	Reserved	—	—
0x09	Reserved	—	—
0x0A	Reserved	—	—
0x0B	Reserved	—	—
0x0C	Reserved	—	—
0x0D	Reserved	—	—
0x0E	Reserved	—	—
0x0F	Reserved	—	—
0x10	Motor Controller Channel Control Register 0 (MCCC0)	RW, 8-bit	on page 1128
0x11	Motor Controller Channel Control Register 1 (MCCC1)	RW, 8-bit	on page 1128
0x12	Motor Controller Channel Control Register 2 (MCCC2)	RW, 8-bit	on page 1128
0x13	Motor Controller Channel Control Register 3 (MCCC3)	RW, 8-bit	on page 1128
0x14	Motor Controller Channel Control Register 4 (MCCC4)	RW, 8-bit	on page 1128
0x15	Motor Controller Channel Control Register 5 (MCCC5)	RW, 8-bit	on page 1128
0x16	Motor Controller Channel Control Register 6 (MCCC6)	RW, 8-bit	on page 1128
0x17	Motor Controller Channel Control Register 7 (MCCC7)	RW, 8-bit	on page 1128
0x18	Motor Controller Channel Control Register 8 (MCCC8)	RW, 8-bit	on page 1128
0x19	Motor Controller Channel Control Register 9 (MCCC9)	RW, 8-bit	on page 1128
0x1A	Motor Controller Channel Control Register 10 (MCCC10)	RW, 8-bit	on page 1128
0x1B	Motor Controller Channel Control Register 11 (MCCC11)	RW, 8-bit	on page 1128
0x1C	Reserved	—	—
0x1D	Reserved	—	—
0x1E	Reserved	—	—
0x1F	Reserved	—	—
0x20	Motor Controller Duty Cycle Register 0 (MCDC0)	RW, 16-bit	on page 1129
0x22	Motor Controller Duty Cycle Register 1 (MCDC1)	RW, 16-bit	on page 1129
0x24	Motor Controller Duty Cycle Register 2 (MCDC2)	RW, 16-bit	on page 1129
0x26	Motor Controller Duty Cycle Register 3 (MCDC3)	RW, 16-bit	on page 1129
0x28	Motor Controller Duty Cycle Register 4 (MCDC4)	RW, 16-bit	on page 1129
0x2A	Motor Controller Duty Cycle Register 5 (MCDC5)	RW, 16-bit	on page 1129
0x2C	Motor Controller Duty Cycle Register 6 (MCDC6)	RW, 16-bit	on page 1129
0x2E	Motor Controller Duty Cycle Register 7 (MCDC7)	RW, 16-bit	on page 1129

Table 35-2. SMC memory map (continued)

Address offset	Use	Recommended Access type	Location
0x30	Motor Controller Duty Cycle Register 8 (MCDC8)	RW, 16-bit	on page 1129
0x32	Motor Controller Duty Cycle Register 9 (MCDC9)	RW, 16-bit	on page 1129
0x34	Motor Controller Duty Cycle Register 10 (MCDC10)	RW, 16-bit	on page 1129
0x36	Motor Controller Duty Cycle Register 11 (MCDC11)	RW, 16-bit	on page 1129
0x38	Reserved	—	—
0x39	Reserved	—	—
0x3A	Reserved	—	—
0x3B	Reserved	—	—
0x3C	Reserved	—	—
0x3D	Reserved	—	—
0x3E	Reserved	—	—
0x3F	Reserved	—	—
0x40	Short-circuit Detector Timeout Register (MCSDTO)	RW, 8-bit	on page 1131
0x41	Reserved	—	—
0x42	Reserved	—	—
0x43	Reserved	—	—
0x44	Short-circuit Detector Enable Register 0 (MCSDE0)	RW, 8-bit	on page 1131
0x45	Short-circuit Detector Enable Register 1 (MCSDE1)	RW, 8-bit	on page 1132
0x46	Short-circuit Detector Enable Register 2 (MCSDE2)	RW, 8-bit	on page 1132
0x47	Reserved	—	—
0x48	Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0)	RW, 8-bit	on page 1133
0x49	Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1)	RW, 8-bit	on page 1133
0x4A	Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2)	RW, 8-bit	on page 1134
0x4B	Reserved	—	—
0x4C	Short-circuit Detector Interrupt Register 0 (MCSDI0)	R/MW, 8-bit	on page 1134
0x4D	Short-circuit Detector Interrupt Register 1 (MCSDI1)	R/MW, 8-bit	on page 1135
0x4E	Short-circuit Detector Interrupt Register 2 (MCSDI2)	R/MW, 8-bit	on page 1135
0x4F	Reserved	—	—

35.3.2 Register description

This section provides detailed descriptions of all registers in ascending address order. Table 35-3 provides a key for the register figures and register tables.

Table 35-3. Register Access Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	

Table 35-3. Register Access Conventions (continued)

Convention	Description
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A flag bit that can be read, is cleared by writing a one, writing 0 has no effect.
Reset Value	
0	Resets to zero.
1	Resets to one.

35.3.2.1 Motor Controller Control Register 0 (MCCTL0)

This register controls the operating mode of the SMC module.

Offset Module Base + 0x0000

	0	1	2	3	4	5	6	7
R	0	MCPRE		0	0	DITH	0	MCTOIF
W								w1c
Reset	0	0	0	0	0	0	0	0

Figure 35-2. Motor Controller Control Register 0 (MCCTL0)

Table 35-4. MCCTL0 field descriptions

Field	Description
MCPRE	**Motor Controller Prescaler Select** — MCPRE determines the prescaler value that sets the motor controller timer counter clock frequency (f_{TC}). The clock source for the prescaler is the peripheral bus clock (f_{BUS}) as shown in Figure 35-32. Writes to MCPRE will not affect the timer counter clock frequency f_{TC} until the start of the next PWM period. 00 $\mathbf{f_{TC}} = f_{Bus}$ 01 $\mathbf{f_{TC}} = f_{Bus}/2$ 10 $\mathbf{f_{TC}} = f_{Bus}/4$ 11 $\mathbf{f_{TC}} = f_{Bus}/8$
DITH	**Motor Control/Driver Dither Feature Enable** (refer to Section 35.4.1.3.5, Dither Bit (MCCTL0[DITH])) 0 Dither feature is disabled. 1 Dither feature is enabled.
MCTOIF	**Motor Controller Timer Counter Overflow Interrupt Flag** — This bit is set when a motor controller timer counter overflow occurs. The bit is cleared by writing a 1 to the bit. 0 A motor controller timer counter overflow has not occurred since the last reset or since the bit was cleared. 1 A motor controller timer counter overflow has occurred.

35.3.2.2 Motor Controller Control Register 1 (MCCTL1)

This register controls the behavior of the analog section of the SMC as well as the interrupt enables.

Offset Module Base + 0x0001

	0	1	2	3	4	5	6	7
R	RECIRC	0	0	0	0	0	0	MCTOIE
W								
Reset	0	0	0	0	0	0	0	0

Figure 35-3. Motor Controller Control Register 1 (MCCTL1)

Table 35-5. MCCTL1 field descriptions

Field	Description
RECIRC	**Recirculation in (Dual) Full H-Bridge Mode** (refer to Section 35.4.1.3.3, Recirculation Bit (MCCTL1[RECIRC])) — RECIRC only affects the outputs in (dual) full H-bridge modes. In half H-bridge mode, the PWM output is always active low. RECIRC = 1 will also invert the effect of the MCDCx[SIGN] bits (refer to Section 35.4.1.3.2, Sign Bit (MCDCx[SIGN])) in (dual) full H-bridge modes. RECIRC must be changed only while no PWM channel is operating in (dual) full H-bridge mode; otherwise, erroneous output pattern may occur. 0 Recirculation on the high side transistors. Active state for PWM output is logic low, the static channel will output logic high. 1 Recirculation on the low side transistors. Active state for PWM output is logic high, the static channel will output logic low.
MCTOIE	**Motor Controller Timer Counter Overflow Interrupt Enable** 0 Interrupt disabled. 1 Interrupt enabled. An interrupt will be generated when the motor controller timer counter overflow interrupt flag (MCCTL0[MCTOIF]) is set.

35.3.2.3 Motor Controller Period Register (MCPER)

Offset Module Base + 0x0002, 0x0003

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

Figure 35-4. Motor Controller Period Register (MCPER)

Table 35-6. MCPER field descriptions

Field	Description
PER	**PWM Period** — PER defines the number of motor controller timer counter clocks a PWM period lasts. The motor controller timer counter is clocked with the frequency f_{TC}. If dither mode is enabled (MCCTL0[DITH] = 1, refer to Section 35.4.1.3.5, Dither Bit (MCCTL0[DITH])), PER[0] is ignored and reads as a 0. In this case PER = 2 * MCDCx[DUTY[10:1]].

Setting PER to 0 will shut off all PWM channels as if MCCCx[MCAM] is set to 0 in all channel control registers after the next period timer counter overflow. In this case, the motor controller releases all pins.

NOTE

Programming PER to 1 and setting the MCCTL0[DITH] bit will be managed as if PER is programmed to 0. All PWM channels will be shut off after the next period timer counter overflow.

35.3.2.4 Motor Controller Channel Control Register (MCCC0..11)

Each PWM channel has one associated control register to control output delay, PWM alignment, and output mode. The number of each register refers directly the PWM channel it controls. The relation between channels, pin names and register names is shown in Table 35-19.

Offset Module Base + 0x0010 . . . 0x001B

	0	1	2	3	4	5	6	7
R	MCOM		MCAM		0	0	CD	
W								
Reset	0	0	0	0	0	0	0	0

Figure 35-5. Motor Controller Channel Control Register (MCCC0..11)

Table 35-7. MCCCx field descriptions

Field	Description
MCOM	**Output Mode** — MCOM controls the PWM channel's output mode. 00 Half H-bridge mode, PWM on pin MnCxM, pin MnCxP is released 01 Half H-bridge mode, PWM on pin MnCxP, pin MnCxM is released 10 Full H-bridge mode 11 Dual full H-bridge mode
MCAM	**PWM Channel Alignment Mode** — MCAM controls the PWM channel's PWM alignment mode and operation. MCAM and MCOM are double buffered. The values used for the generation of the output waveform will be copied to the working registers either at once (if all PWM channels are disabled or MCPER[PER] is set to 0) or if a timer counter overflow occurs. Reads of the register return the most recent written value, which are not necessarily the currently active values. 00 Channel disabled 01 Left aligned 10 Right aligned 11 Center aligned
CD	**PWM Channel Delay** — Each PWM channel can be individually delayed by a programmable number of PWM timer counter clocks. The delay will be n/f_{TC}. 00 Zero PWM clocks channel delay 01 One PWM clock channel delay 10 Two PWM clocks channel delay 11 Three PWM clocks channel delay

NOTE

The SMC will release the pins after the next PWM timer counter overflow without accommodating any channel delay if a single channel has been disabled or if the period register has been cleared or all channels have been disabled. Program one or more inactive PWM frames (duty cycle = 0) before writing a configuration that disables a single channel or the entire SMC.

35.3.2.5 Motor Controller Duty Cycle Register (MCDC0..11)

Each duty cycle register sets the sign and duty functionality for the respective PWM channel. The number of each register refires directly the PWM channel it controls. The relation between channels, pin names and register names is shown in Table 35-19.

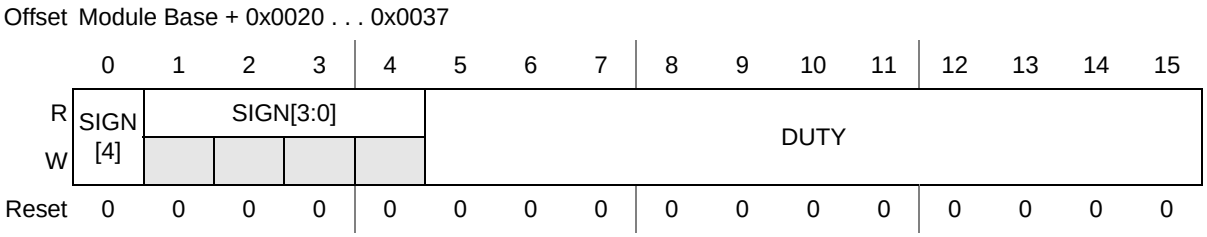


Figure 35-6. Motor Controller Duty Cycle Register (MCDC0..11)

Table 35-8. MCDCx field descriptions

Field	Description
SIGN[4]	Sign Bit. The SIGN[4] bit is used to define which output will drive the PWM signal in (dual) full-H-bridge modes. The SIGN[4] bit has no effect in half-bridge modes. See Section 35.4.1.3.2, Sign Bit (MCDCx[SIGN]) and Table 35-20 for detailed information about the impact of MCCTL1[RECIRC] and SIGN[4] bit on the PWM output.
SIGN[3:0]	Sign Bit Extension. Replicates the SIGN[4] bit towards the DUTY field to make the whole register a signed representation for the duty cycle length.
DUTY	Duty Cycle Length. DUTY defines the number of motor controller timer counter clocks the corresponding output is driven low (MCCTL1[RECIRC] = 0) or is driven high (MCCTL1[RECIRC] = 1). Setting all bits to 0 will give a static high output in case of MCCTL1[RECIRC] = 0; otherwise, a static low output. Values greater than or equal to the contents of the period register will generate a static low output in case of MCCTL1[RECIRC] = 0, or a static high output if MCCTL1[RECIRC] = 1.

To prevent the output from inconsistent signals, the duty cycle registers are double buffered. The SMC module will use working registers to generate the output signals. The working registers are copied from the bus accessible registers at the following conditions:

- MCPER[PER] is set to 0 (all channels are disabled in this case)
- MCCCx[MCAM] of the respective channel is set to 0 (channel is disabled)
- A PWM timer counter overflow occurs while in half H-bridge or full H-bridge mode
- A PWM channel pair is configured to work in Dual Full H-Bridge mode and a PWM timer counter overflow occurs after the odd[1] duty cycle register of the channel pair has been written.

In this way, the output of the PWM will always be either the old PWM waveform or the new PWM waveform, not some variation in between.

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active sign, duty cycle, and dither functionality due to the double buffering scheme.

1. Odd duty cycle register: MCDCx+1, x = 2·n

35.3.2.6 Short-circuit Detector Timeout Register (MCSDTO)

Offset Module Base + 0x0040

	0	1	2	3	5	7
R W	TOUT					
Reset	0	0	0	0	0	0	0	0

Figure 35-7. Short-circuit Detector Timeout Register (MCSDTO)

Table 35-9. MCSDTO field descriptions

Field	Description
TOUT	Timeout . The value TOUT is an unsigned 8-bit number. This value is used as load value for the short-circuit detection counters. This value is applied to all 24 short-circuit detection blocks. Due to synchronization and sampling, TOUT must always be larger than 2 (see also Section 35.4.6, Short-circuit detection).

35.3.2.7 Short-circuit Detector Enable Register 0 (MCSDE0)

Offset Module Base + 0x0044

	0	1	2	3	4	5	6	7
R W	SDEN							
Reset	0	0	0	0	0	0	0	0

Figure 35-8. Short-circuit Detector Enable Register 0 (MCSDE0)

Table 35-10. MCSDE0 field description

Field	Description
SDEN	Short-Circuit Detector Enable. Each short-circuit detector can be enabled or disabled according to the mapping described in Table 35-23. The short-circuit detector of a given pin is enabled if the related enable bit is set to 1.

35.3.2.8 Short-circuit Detector Enable Register 1 (MCSDE1)

Offset Module Base + 0x0045

	0	1	2	3	4	5	6	7
R W	SDEN							
Reset	0	0	0	0	0	0	0	0

Figure 35-9. Short-circuit Detector Enable Register 1 (MCSDE1)

Table 35-11. MCSDE1 field description

Field	Description
SDEN	**Short-Circuit Detector Enable** — Each short-circuit detector can be enabled or disabled according to the mapping described in Table 35-23. The short-circuit detector of a given pin is enabled if the related enable bit is set to 1.

35.3.2.9 Short-circuit Detector Enable Register 2 (MCSDE2)

Offset Module Base + 0x0046

	0	1	2	3	4	5	6	7
R W	SDEN							
Reset	0	0	0	0	0	0	0	0

Figure 35-10. Short-circuit Detector Enable Register 2 (MCSDE2)

Table 35-12. MCSDE2 field description

Field	Description
SDEN	**Short-Circuit Detector Enable** — Each short-circuit detector can be enabled or disabled according to the mapping described in Table 35-23. The short-circuit detector of a given pin is enabled if the related enable bit is set to 1.

35.3.2.10 Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0)

Offset Module Base + 0x0048

	0	1	2	3	4	5	6	7
R W	SDIE							
Reset	0	0	0	0	0	0	0	0

Figure 35-11. Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0)

Table 35-13. MCSDIEN0 field descriptions

Field	Description
SDIE	**Short-Circuit Detector Interrupt Enable** — The interrupt of each short-circuit detector can individually be enabled or disabled according to the mapping described in Table 35-23. The short-circuit detector interrupt of a given pin is enabled if the related interrupt enable bit is set to 1.

35.3.2.11 Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1)

Offset Module Base + 0x0049

	0	1	2	3	4	5	6	7
R W	SDIE							
Reset	0	0	0	0	0	0	0	0

Figure 35-12. Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1)

Table 35-14. MCSDIEN1 field descriptions

Field	Description
SDIE	**Short-Circuit Detector Interrupt Enable** — The interrupt of each short-circuit detector can individually be enabled or disabled according to the mapping described in Table 35-23. The short-circuit detector interrupt of a given pin is enabled if the related interrupt enable bit is set to 1.

35.3.2.12 Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2)

Offset Module Base + 0x004A

	0	1	2	3	4	5	6	7
R	SDIE							
W								
Reset	0	0	0	0	0	0	0	0

Figure 35-13. Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2)

Table 35-15. MCSDIEN2 field descriptions

Field	Description
SDIE	**Short-Circuit Detector Interrupt Enable** — The interrupt of each short-circuit detector can individually be enabled or disabled according to the mapping described in Table 35-23. The short-circuit detector interrupt of a given pin is enabled if the related interrupt enable bit is set to 1.

35.3.2.13 Short-circuit Detector Interrupt Register 0 (MCSDI0)

Offset Module Base + 0x004C

	0	1	2	3	4	5	6	7
R	SDIF							
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0

Figure 35-14. Short-circuit Detector Interrupt Register 0 (MCSDI0)

Table 35-16. MCSDI0 field descriptions

Field	Description
SDIF	**Short-circuit Detector Interrupt Flag** — In case of a detected short-circuit, the corresponding bit according to the mapping in Table 35-23 is set in the short-circuit detector interrupt register. If this specific interrupt is also enabled in the interrupt enable register MCSDIEN0, than this event will rise an external interrupt.

35.3.2.14 Short-circuit Detector Interrupt Register 1 (MCSDI1)

Offset Module Base + 0x004D

	0	1	2	3	4	5	6	7
R	SDIF							
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0

Figure 35-15. Short-circuit Detector Interrupt Register 1 (MCSDI1)

Table 35-17. MCSDI1 field descriptions

Field	Description
SDIF	**Short-circuit Detector Interrupt Flag** — In case of a detected short-circuit, the corresponding bit according to the mapping in Table 35-23 is set in the short-circuit detector interrupt register. If this specific interrupt is also enabled in the interrupt enable register MCSDIEN1, than this event will rise an external interrupt.

35.3.2.15 Short-circuit Detector Interrupt Register 2 (MCSDI2)

Offset Module Base + 0x004E

	0	1	2	3	4	5	6	7
R	SDIF							
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0

Figure 35-16. Short-circuit Detector Interrupt Register 2 (MCSDI2)

Table 35-18. MCSDI2 field descriptions

Field	Descriptions
SDIF	**Short-circuit Detector Interrupt Flag** — In case of a detected short-circuit, the corresponding bit according to the mapping in Table 35-23 is set in the short-circuit detector interrupt register. If this specific interrupt is also enabled in the interrupt enable register MCSDIEN2, than this event will rise an external interrupt.

35.4 Functional description

35.4.1 Modes of operation

35.4.1.1 PWM output modes

The SMC is configured between three output modes.

- Dual full H-bridge mode can be used to control either a stepper motor or a 360° air core instrument. In this case two PWM channels are combined.
- In full H-bridge mode, each PWM channel is updated independently.
- In half H-bridge mode, one pin of the PWM channel can generate a PWM signal to control a 90° air core instrument (or other load requiring a PWM signal) and the other pin is unused.

The mode of operation for PWM channel x is determined by the output mode bits MCCCx[MCOM]. After a reset occurs, each PWM channel will be disabled, the corresponding pins are released.

Each PWM channel consists of two pins. One output pin will generate a PWM signal. The other will operate as logic high or low output depending on the state of the recirculation bit MCCTL1[RECIRC] (refer to Section 35.4.1.3.3, Recirculation Bit (MCCTL1[RECIRC])), while in (dual) full H-bridge mode, or will be released, while in half H-bridge mode. The state of the sign bit MCDCx[SIGN[4]] in the duty cycle register determines the pin where the PWM signal is driven in full H-bridge mode. While in half H-bridge mode, the state of the released pin is determined by other modules associated with this pin.

Associated with each PWM channel pair n are two PWM channels, x and x + 1, where x = 2 * n and n (0,1,2... 5) is the PWM channel pair number. Duty cycle register x controls the sign of the PWM signal (which pin drives the PWM signal) and the duty cycle of the PWM signal for SMC channel x. The pins associated with PWM channel x are MnC0P and MnC0M. Similarly, duty cycle register x + 1 controls the sign of the PWM signal and the duty cycle of the PWM signal for channel x + 1. The pins associated with PWM channel x + 1 are MnC1P and MnC1M. This is summarized in Table 35-19.

Table 35-19. Corresponding Registers and Pin Names for each PWM Channel Pair

PWM Channel Pair Number	PWM Channel Control Register	Duty Cycle Register	Channel Number	Pin Names
n	MCCCx	MCDCx	PWM Channel x, x = 2·n	MnC0M
				MnC0P
	MCCCx+1	MCDCx+1	PWM Channel x+1, x = 2·n	MnC1M
				MnC1P
0	MCCC0	MCDC0	PWM Channel 0	M0C0M
				M0C0P
	MCCC1	MCDC1	PWM Channel 1	M0C1M
				M0C1P
1	MCCC2	MCDC2	PWM Channel 2	M1C0M
				M1C0P
	MCCC3	MCDC3	PWM Channel 3	M1C1M
				M1C1P
2	MCCC4	MCDC4	PWM Channel 4	M2C0M
				M2C0P
	MCCC5	MCDC5	PWM Channel 5	M2C1M
				M2C1P

Table 35-19. Corresponding Registers and Pin Names for each PWM Channel Pair (continued)

PWM Channel Pair Number	PWM Channel Control Register	Duty Cycle Register	Channel Number	Pin Names
3	MCCC6	MCDC6	PWM Channel 6	M3C0M
				M3C0P
	MCCC7	MCDC7	PWM Channel 7	M3C1M
				M3C1P
4	MCCC8	MCDC8	PWM Channel 8	M4C0M
				M4C0P
	MCCC9	MCDC9	PWM Channel 9	M4C1M
				M4C1P
5	MCCC10	MCDC10	PWM Channel 10	M5C0M
				M5C0P
	MCCC11	MCDC11	PWM Channel 11	M5C1M
				M5C1P

35.4.1.1.1 Dual full H-bridge mode

PWM channel pairs x and x + 1 operate in dual full H-bridge mode if both channels are enabled (MCCCx[MCAM]=0x1, 0x2, or 0x3) and the output mode bits MCCCx[MCOM] in both PWM channel control registers are set to 0x3.

A typical configuration in dual full H-bridge mode is shown in Figure 35-17. PWM channel x drives the PWM output signal on either MnC0P or MnC0M. If MnC0P drives the PWM signal, MnC0M will be output either high or low depending on the MCCTL1[RECIRC] bit. If MnC0M drives the PWM signal, MnC0P will be an output high or low. PWM channel x + 1 drives the PWM output signal on either MnC1P or MnC1M. If MnC1P drives the PWM signal, MnC1M will be an output high or low. If MnC1M drives the PWM signal, MnC1P will be an output high or low. This results in motor recirculation currents on the high side drivers (MCCTL1[RECIRC] = 0) while the PWM signal is at a logic high level, or motor recirculation currents on the low side drivers (MCCTL1[RECIRC] = 1) while the PWM signal is at a logic low level. The pin driving the PWM signal is determined by the sign bit MCDCx[SIGN[4]] for the corresponding channel and the state of the MCCTL1[RECIRC] bit. The value of the PWM duty cycle is determined by the value of the duty cycle bits MCDCx[DUTY] for the corresponding channel.

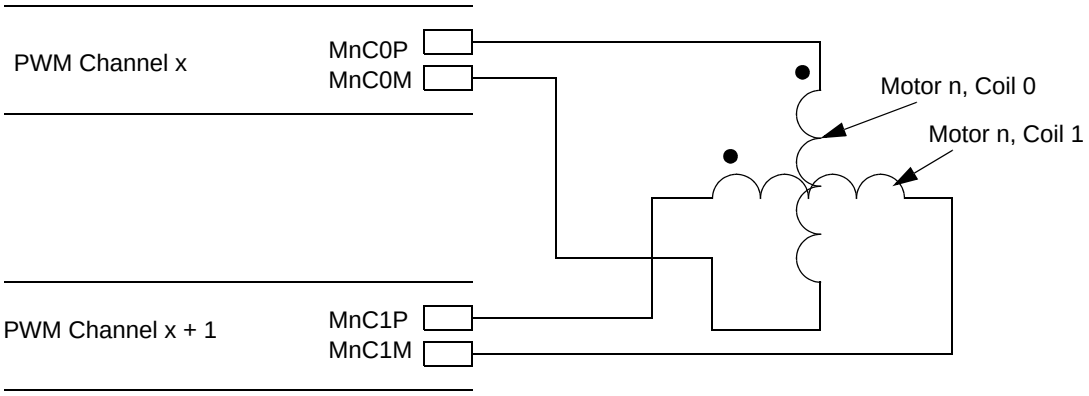


Figure 35-17. Typical dual full H-bridge mode configuration

16-bit write accesses to the duty cycle registers are allowed, 8-bit write accesses can lead to unpredictable duty cycles.

The following sequence should be used to update the current magnitude and direction for coil 0 and coil 1 of the motor to achieve consistent PWM output:

1. Write to duty cycle register x
2. Write to duty cycle register x + 1

At the next timer counter overflow, the duty cycle registers will be copied to the working duty cycle registers. Sequential writes to the duty cycle register x will result in the previous data being overwritten.

35.4.1.1.2 Full H-bridge mode

In full H-bridge mode (MCCCx[MCOM]=0x2), the PWM channels x and x + 1 operate independently. The duty cycle working registers are updated whenever a timer counter overflow occurs.

35.4.1.1.3 Half H-bridge mode

In half H-bridge mode (MCCCx[MCOM] = 0x0 or 0x1), the PWM channels x and x + 1 operate independently. In this mode, each PWM channel can be configured such that one pin is released and the other pin is a PWM output. Figure 35-18 shows a typical configuration in half H-bridge mode.

The two pins associated with each channel are switchable between released mode and PWM output dependent upon the state of the output mode bits MCCCx[MCOM]. See register description in Section 35.3.2.4, Motor Controller Channel Control Register (MCCC0..11). In half H-bridge mode, the state of the MCDCx[SIGN[4]] bit has no effect.

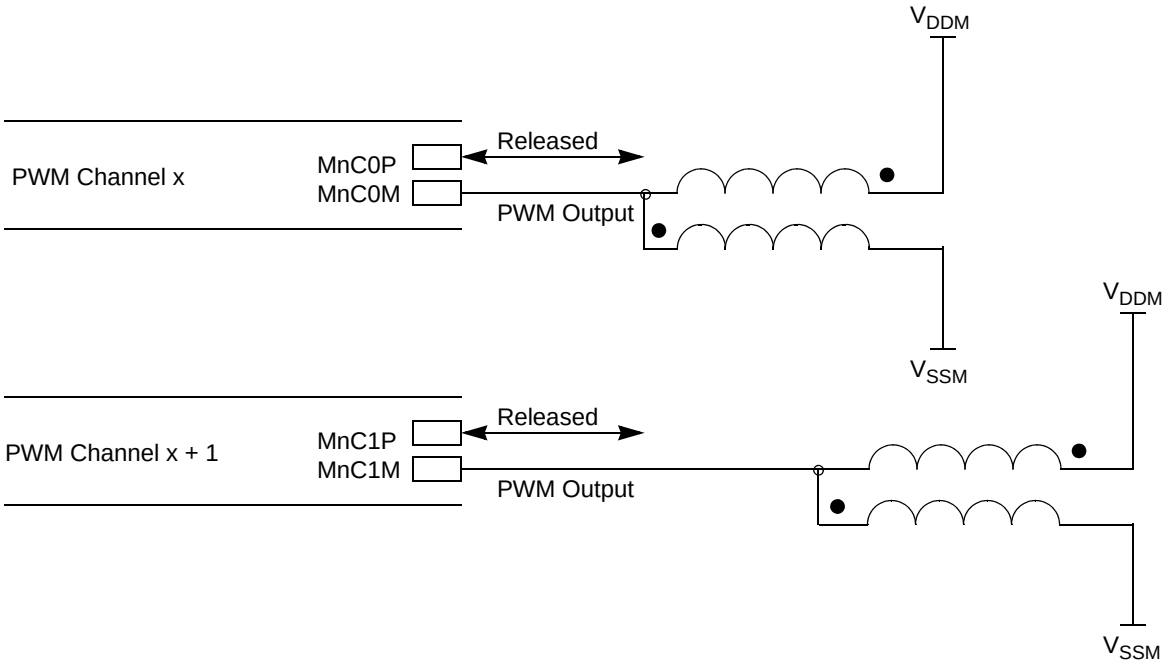


Figure 35-18. Typical quad half H-bridge mode configuration

35.4.1.2 Relationship between PWM mode and PWM channel enable

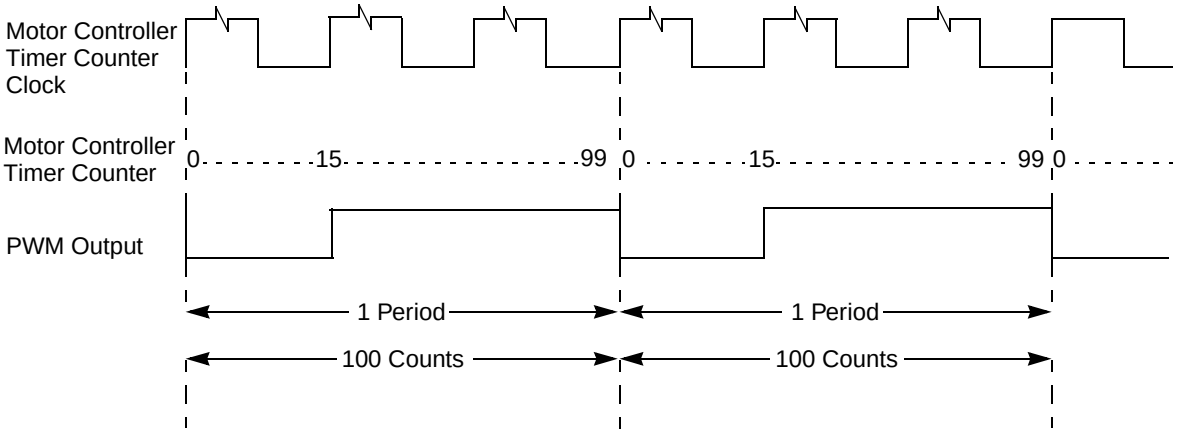
The pair of SMC channels cannot be placed into dual full H-bridge mode unless both SMC channels have been enabled (MCCCx[MCAM] not equal to 0) and dual full H-bridge mode is selected for both PWM channels (MCCCx[MCOM] = 0x3). If only one channel is set to dual full H-bridge mode, this channel will operate in full H-bridge mode, the other as programmed.

35.4.1.3 Relationship between Sign, Duty, Dither, RECIRC, Period, and PWM mode functions

35.4.1.3.1 PWM alignment modes

Each PWM channel can be programmed individually to three different alignment modes. The alignment mode is determined by the MCCCx[MCAM] bits in the corresponding channel control register.

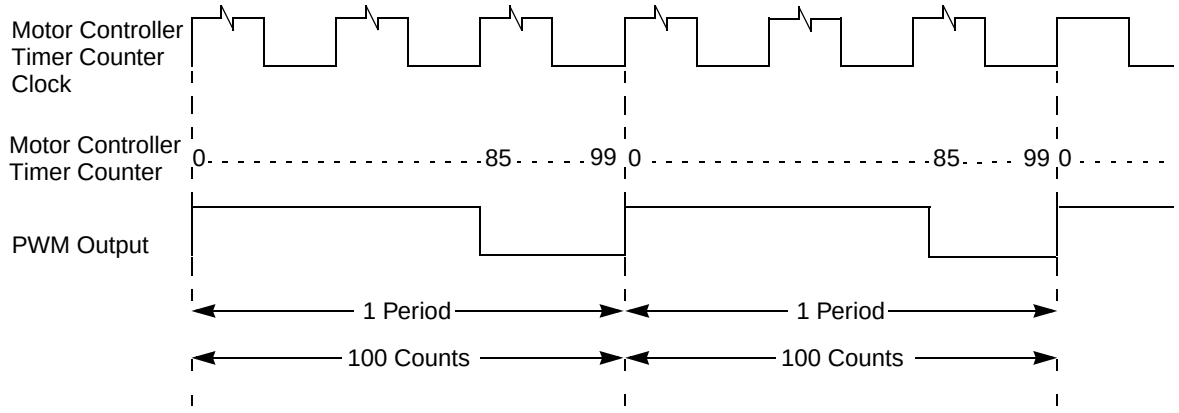
Left aligned (MCCCx[MCAM] = 0x1): The output will start active (low if MCCTL1[RECIRC] = 0 or high if MCCTL1[RECIRC] = 1) and will turn inactive (high if MCCTL1[RECIRC] = 0 or low if MCCTL1[RECIRC] = 1) after the number of counts specified by the corresponding duty cycle register.



MCCTL0[DITH] = 0, MCCCx[MCAM] = 0x1, MCDCx[DUTY] = 15, MCPER[PER] = 100, MCCTL1[RECIRC] = 0

Figure 35-19. Left aligned

Right aligned (MCCCx[MCAM] = 0x2): The output will start inactive (high if MCCTL1[RECIRC] = 0 and low if
MCCTL1[RECIRC] = 1) and will turn active after the number of counts specified by the difference of the contents of period register and the corresponding duty cycle register.



MCCTL0[DITH] = 0, MCCCx[MCAM] = 0x2, MCDCx[DUTY] = 15, MCPER[PER] = 100, MCCTL1[RECIRC] = 0

Figure 35-20. Right aligned

Center aligned (MCCCx[MCAM] = 0x3): Even periods will be output left aligned, odd periods will be output right aligned. PWM operation starts with the even period after the channel has been enabled. PWM operation in center aligned mode might start with the odd period if the channel has not been disabled before changing the alignment mode to center aligned.

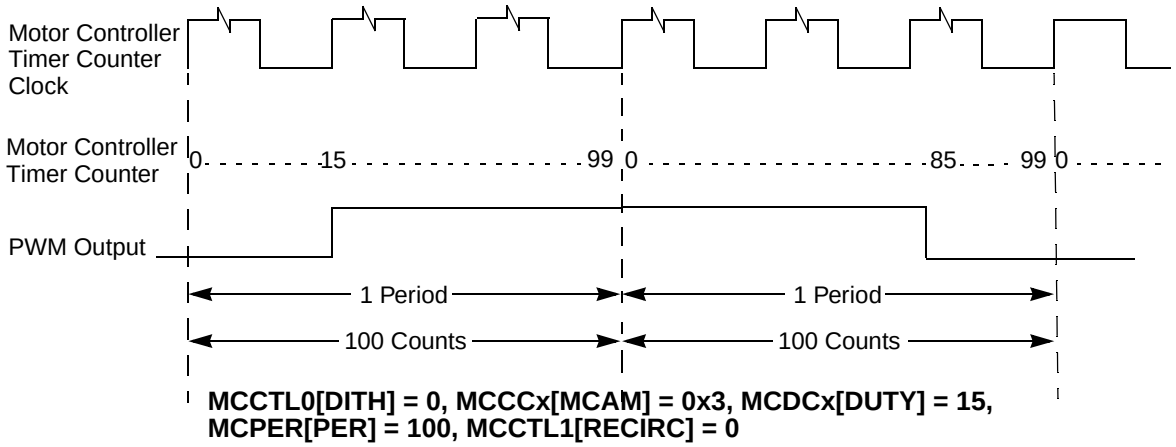


Figure 35-21. Center aligned

35.4.1.3.2 Sign Bit (MCDCx[SIGN])

Assuming MCCTL1[RECIRC] = 0 (the active state of the PWM signal is low), when the MCDCx[SIGN[4]] bit for the corresponding channel is cleared, MnC0P (if the PWM channel number is even, n = 0, 1, 2...5, see Table 35-19) or MnC1P (if the PWM channel number is odd, n = 0, 1, 2...5 see Table 35-19), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the MCDCx[SIGN[4]] bit has no effect. The PWM output signal is generated on MnC0M (if the PWM channel number is even, n = 0, 1, 2...5, see Table 35-19) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2...5).

Assuming MCCTL1[RECIRC] = 0 (the active state of the PWM signal is low), when the MCDCx[SIGN[4]] bit for the corresponding channel is set, MnC0M (if the PWM channel number is even, n = 0, 1, 2...5, see Table 35-19) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2...5, see Table 35-19), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the MCDCx[SIGN[4]] bit has no effect. The PWM output signal is generated on MnC0P (if the PWM channel number is even, n = 0, 1, 2...5, see Table 35-19) or MnC1P (if the PWM channel number is odd, n = 0, 1, 2...5).

Setting MCCTL1[RECIRC] = 1 will also invert the effect of the MCDCx[SIGN[4]] bit such that while MCDCx[SIGN[4]] = 0, MnC0P or MnC1P will generate the PWM signal and MnC0M or MnC1M will be a static low output. While MCDCx[SIGN[4]] = 1, MnC0M or MnC1M will generate the PWM signal and MnC0P or MnC1P will be a static low output. In this case the active state of the PWM signal will be high.

See Table 35-20 for detailed information about the impact of MCDCx[SIGN[4]] and MCCTL1[RECIRC] bit on the PWM output.

Table 35-20. Impact of MCCTL1[RECIRC] and MCDCx[SIGN[4]] bit on the PWM output

Output Mode	MCCTL1[RECIRC]	MCDCx[SIGN[4]]	MnCyM	MnCyP
(Dual) Full H-Bridge	0	0	$\overline{PWM}$[1]	1
(Dual) Full H-Bridge	0	1	1	$\overline{PWM}$

Table 35-20. Impact of MCCTL1[RECIRC] and MCDCx[SIGN[4]] bit on the PWM output (continued)

Output Mode	MCCTL1[RECIRC]	MCDCx[SIGN[4]]	MnCyM	MnCyP
(Dual) Full H-Bridge	1	0	0	PWM[2]
(Dual) Full H-Bridge	1	1	PWM	0
Half H-Bridge: PWM on MnCyM	Don't care	Don't care	$\overline{\text{PWM}}$	—[3]
Half H-Bridge: PWM on MnCyP	Don't care	Don't care	—	$\overline{\text{PWM}}$

1 $\overline{\text{PWM}}$: The PWM signal is low active. e.g., the waveform starts with 0 in left aligned mode. Output M generates the PWM signal. Output P is static high.

2 PWM: The PWM signal is high active. e.g., the waveform starts with 1 in left aligned mode. output P generates the PWM signal. Output M is static low.

3 The state of the output transistors is not controlled by the SMC.

35.4.1.3.3 Recirculation Bit (MCCTL1[RECIRC])

The MCCTL1[RECIRC] bit controls the flow of the recirculation current of the load. Setting MCCTL1[RECIRC] = 0 will cause recirculation current to flow through the high side transistors, and MCCTL1[RECIRC] = 1 will cause the recirculation current to flow through the low side transistors. The MCCTL1[RECIRC] bit is only active in (dual) full H-bridge modes.

Effectively, MCCTL1[RECIRC] = 0 will cause a static high output on the output terminal not driven by the PWM, MCCTL1[RECIRC] = 1 will cause a static low output on the output terminals not driven by the PWM. To achieve the same current direction, the MCDCx[SIGN[4]] bit behavior is inverted if MCCTL1[RECIRC] = 1. Figure 35-22, Figure 35-23, Figure 35-24, and Figure 35-25 illustrate the effect of the MCCTL1[RECIRC] bit in (dual) full H-bridge modes.

MCCTL1[RECIRC] bit must be changed only while no PWM channel is operated in (dual) full H-bridge mode.

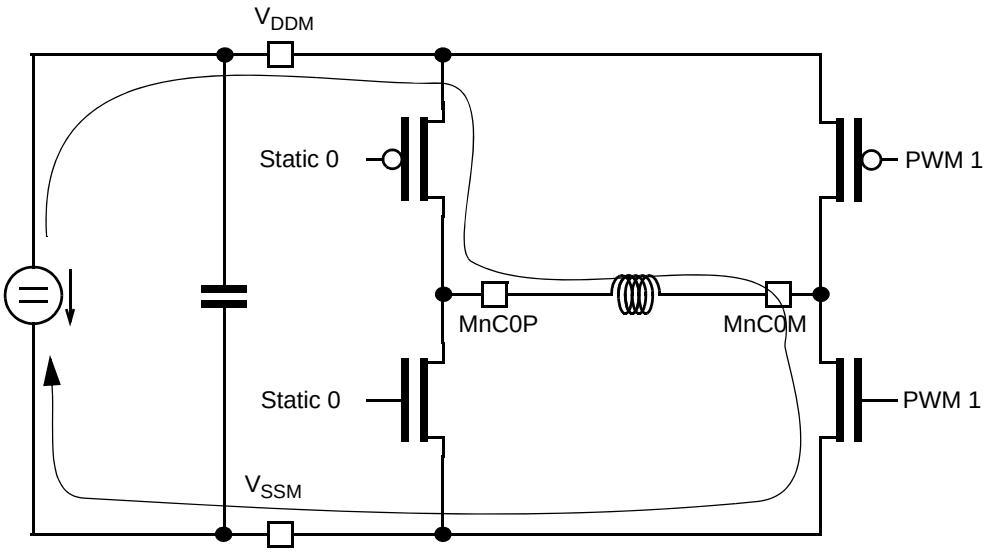


Figure 35-22. PWM Active Phase, MCCTL1[RECIRC] = 0, MCDCx[SIGN[4]] = 0

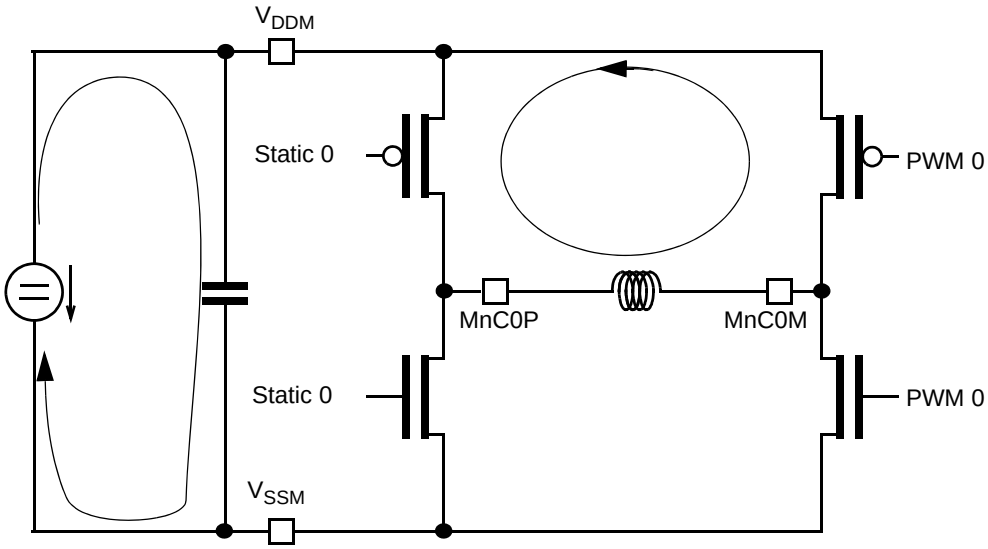


Figure 35-23. PWM Passive Phase, MCCTL1[RECIRC] = 0, MCDCx[SIGN[4]] = 0

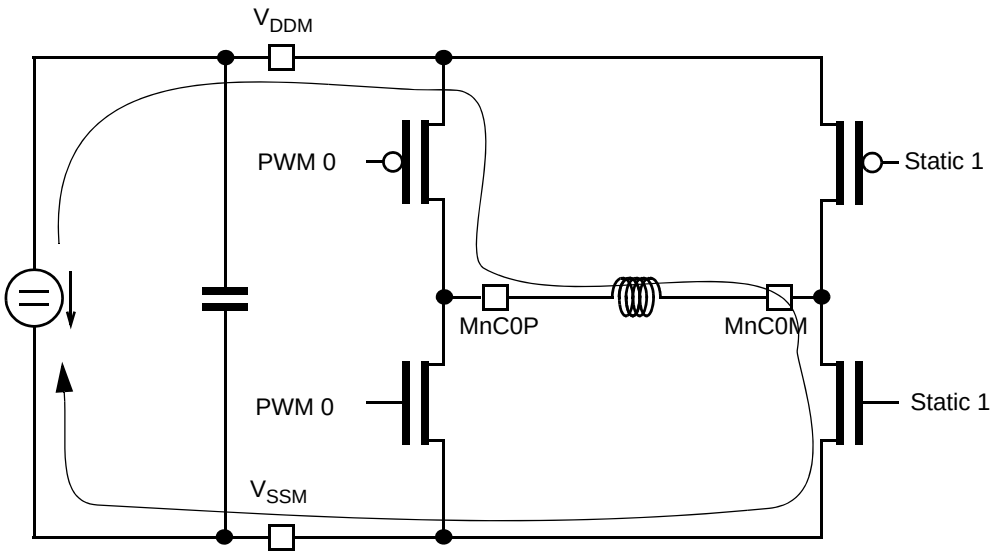


Figure 35-24. PWM Active Phase, MCCTL1[RECIRC] = 1, MCDCx[SIGN[4]] = 0

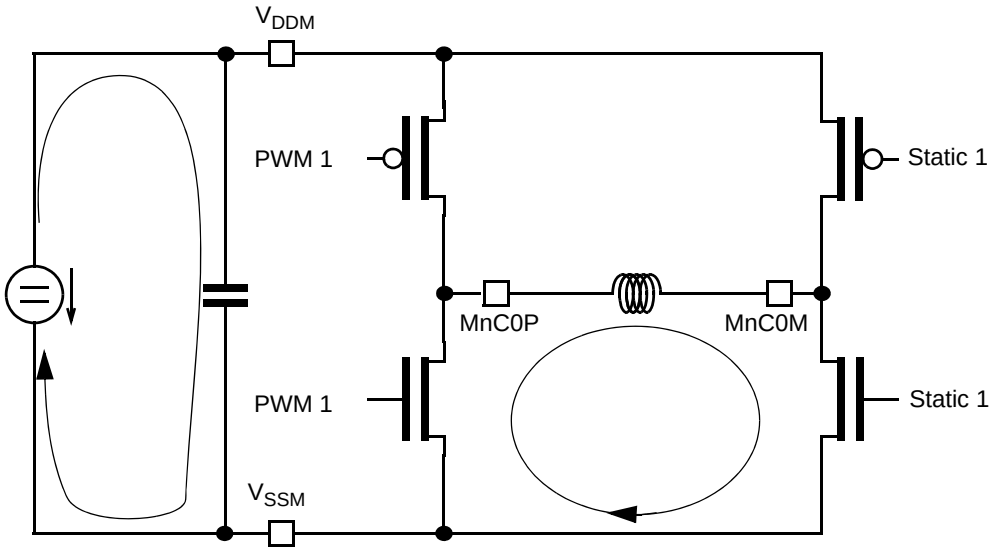


Figure 35-25. PWM Passive Phase, MCCTL1[RECIRC] = 1, MCDCx[SIGN[4]] = 0

35.4.1.3.4 Relationship between MCCTL1[RECIRC] bit, MCDCx[SIGN[4]] bit, MCCCx[MCOM] bits, PWM state, and output transistors

Please refer to Figure 35-26 for the output transistor assignment.

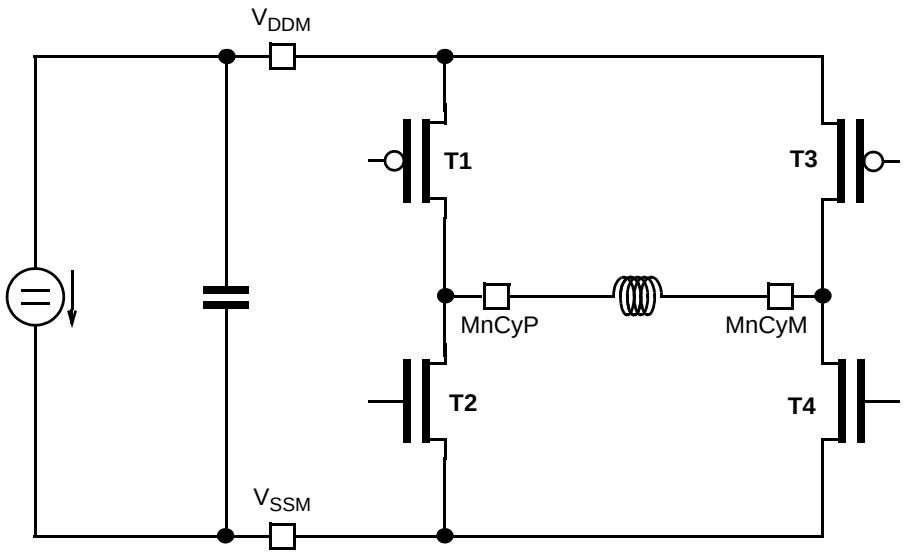


Figure 35-26. Output transistor assignment

Table 35-21 illustrates the state of the output transistors in different states of the SMC module. '—' means that the state of the output transistor is not controlled by the SMC.

Table 35-21. State of output transistors in various modes

Mode	MCCCx [MCOM]	PWM Duty[1]	MCCTL1 [RECIRC]	MCDCx [SIGN[4]]	T1	T2	T3	T4	MnCyM	MnCyP
Off	Don't care	—	Don't care	Don't care	—	—	—	—	—	—
Half H-Bridge	0x0	Active	Don't care	Don't care	—	—	OFF	ON	0	—
Half H-Bridge	0x0	Passive	Don't care	Don't care	—	—	ON	OFF	1	—
Half H-Bridge	0x1	Active	Don't care	Don't care	OFF	ON	—	—	—	0
Half H-Bridge	0x1	Passive	Don't care	Don't care	ON	OFF	—	—	—	1
(Dual) Full	0x2 or 0x3	Active	0	0	ON	OFF	OFF	ON	0	1
(Dual) Full	0x2 or 0x3	Passive	0	0	ON	OFF	ON	OFF	1	1
(Dual) Full	0x2 or 0x3	Active	0	1	OFF	ON	ON	OFF	1	0
(Dual) Full	0x2 or 0x3	Passive	0	1	ON	OFF	ON	OFF	1	1
(Dual) Full	0x2 or 0x3	Active	1	0	ON	OFF	OFF	ON	0	1
(Dual) Full	0x2 or 0x3	Passive	1	0	OFF	ON	OFF	ON	0	0
(Dual) Full	0x2 or 0x3	Active	1	1	OFF	ON	ON	OFF	1	0
(Dual) Full	0x2 or 0x3	Passive	1	1	OFF	ON	OFF	ON	0	0

1 When in (Dual) Full mode and RECIRC=0, the PWM is 0 when the duty cycle is active and 1 when it is passive. When RECIRC = 1, the opposite is true.

35.4.1.3.5 Dither Bit (MCCTL0[DITH])

The purpose of the dither mode is to increase the minimum length of output pulses without decreasing the PWM resolution, in order to limit the pulse distortion introduced by the slew rate control of the outputs. If dither mode is selected the output pattern will repeat after two timer counter overflows. For the same output frequency, the shortest output pulse will have twice the length while dither feature is selected. To achieve the same output frame frequency, the prescaler of the SMC module has to be set to twice the division rate if dither mode is selected; e.g., with the same prescaler division rate the repeat rate of the output pattern is the same as well as the shortest output pulse with or without dither mode selected.

The MCCTL0[DITH] bit enables or disables the dither function.

MCCTL0[DITH] = 0: dither function is disabled.

When MCCTL0[DITH] is cleared and assuming left aligned operation and MCCTL1[RECIRC] = 0, the PWM output will start at a logic low level at the beginning of the PWM period (motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 11-bit PWM duty cycle value MCDCx[DUTY]. When a match (output compare between motor controller timer counter and MCDCx[DUTY]) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the contents of MCPER[PER] – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes one PWM period. The PWM period repeats every MCPER[PER] counts of the motor controller timer counter. If MCDCx[DUTY] >= MCPER[PER], the output will be static low. If MCDCx[DUTY] = 0, the output will be continuously at a logic high level. The relationship between the motor controller timer counter clock, motor controller timer counter value, and PWM output while MCCTL0[DITH] = 0 is shown in Figure 35-27.

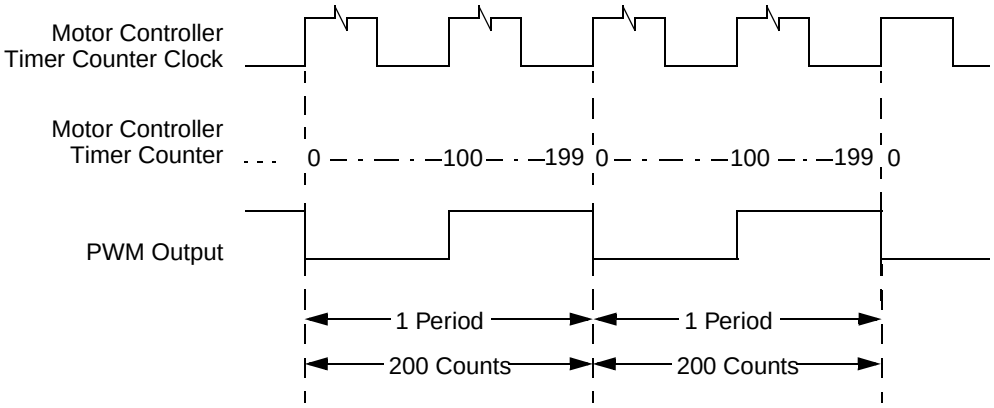


Figure 35-27. PWM Output: MCCTL0[DITH] = 0, MCCCx[MCAM] = 0x1, MCDCx[DUTY] = 100, MCPER[PER] = 200, MCCTL1[RECIRC] = 0

MCCTL0[DITH] = 1: dither function is enabled

Please note if MCCTL0[DITH] = 1, the bit MCPER[PER[0]] will be internally forced to 0 and read always as 0.

When MCCTL0[DITH] is set and assuming left aligned operation and MCCTL1[RECIRC] = 0, the PWM output will start at a logic low level at the beginning of the PWM period (when the motor controller timer counter = 0). The PWM output remains low until the motor controller timer counter matches the 10-bit

PWM duty cycle value MCDCx[DUTY]. When a match (output compare between motor controller timer counter and MCDCx[DUTY]) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by MCPER[PER[10:1]] – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes the first half of the PWM period. During the second half of the PWM period, the PWM output will remain at a logic low level until either the motor controller timer counter matches the 10-bit PWM duty cycle value MCDCx[DUTY] if MCDCx[DUTY[0]]= 0, or the motor controller timer counter matches the 10-bit PWM duty cycle value + 1 (the value of MCDCx[DUTY[10:1]] is incremented by 1 and is compared with the motor controller timer counter value) if MCDCx[DUTY[0]] = 1 for the corresponding channel. When a match occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by MCPER[PER[10:1]] – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level.

This process will repeat every number of counts of the motor controller timer counter defined by the period register contents (MCPER[PER]). If the output is neither set to 0% nor to 100% there will be four edges on the PWM output per PWM period in this case. Therefore, the PWM output compare function will alternate between MCDCx[DUTY] and MCDCx[DUTY] + 1 every half PWM period if MCDCx[DUTY[0]] for the corresponding channel is set to 1. The relationship between the motor controller timer counter clock (f_{TC}), motor controller timer counter value, and left aligned PWM output if MCCTL0[DITH] = 1 is shown in Figure 35-28 and Figure 35-29. Figure 35-30 and Figure 35-31 show right aligned and center aligned PWM operation respectively, with dither feature enabled and MCDCx[DUTY[0]] = 1. Please note: In the following examples, the MCPER[PER] value, which is, if MCCTL0[DITH] = 1, always an even number.

NOTE

The MCCTL0[DITH] bit must be changed only if the SMC is disabled (all channels disabled or period register cleared) to avoid erroneous waveforms.

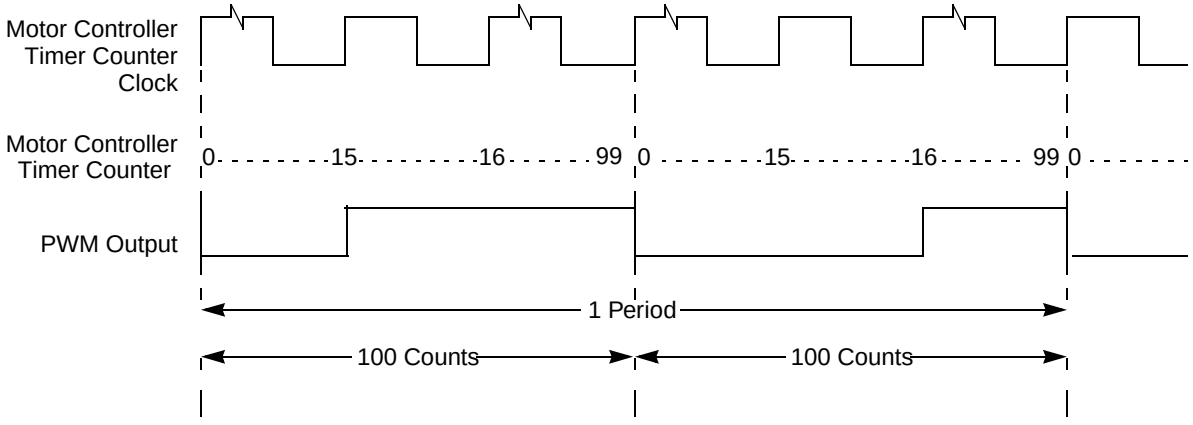


Figure 35-28. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x1, MCDCx[DUTY] = 31, MCPER[PER] = 200, MCCTL1[RECIRC] = 0

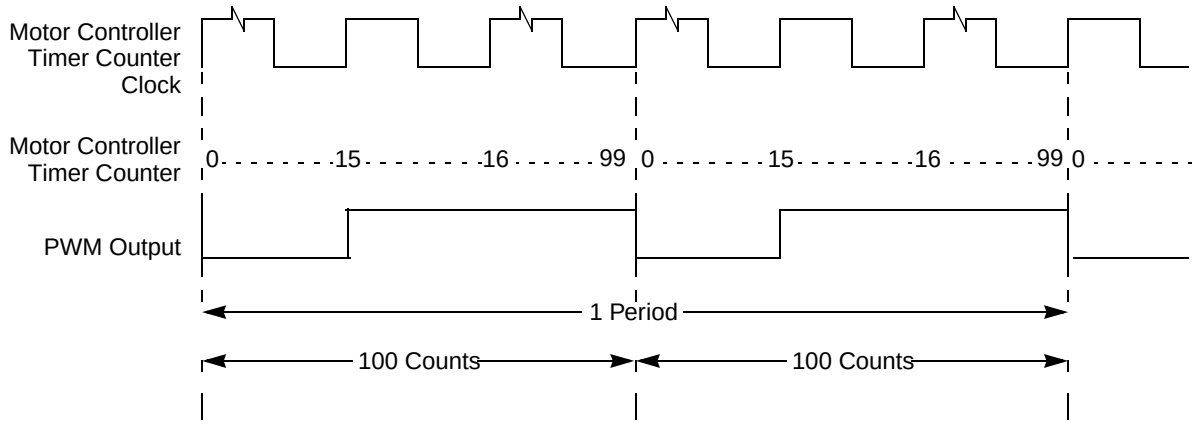


Figure 35-29. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x1, MCDCx[DUTY] = 30, MCPER[PER] = 200, MCCTL1[RECIRC] = 0

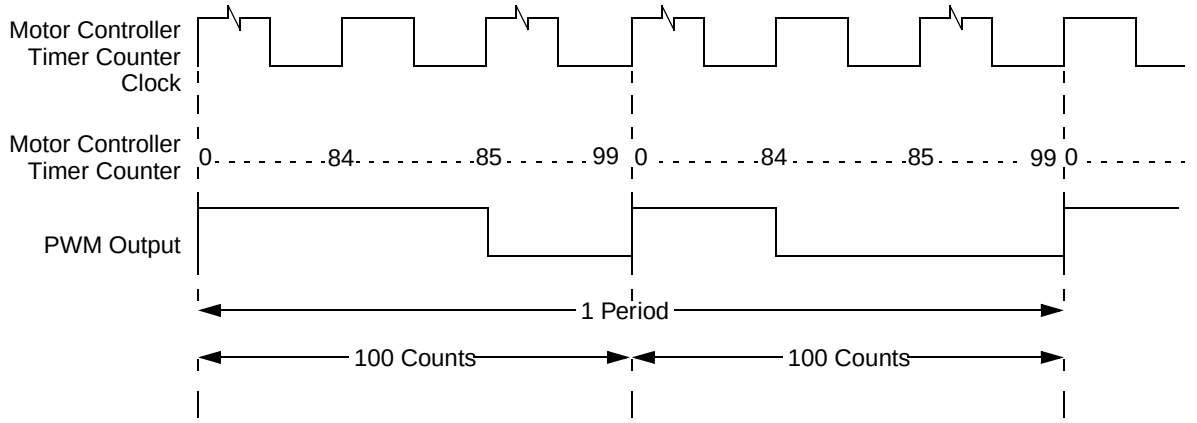


Figure 35-30. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x2, MCDCx[DUTY] = 31, MCPER[PER] = 200, MCCTL1[RECIRC] = 0

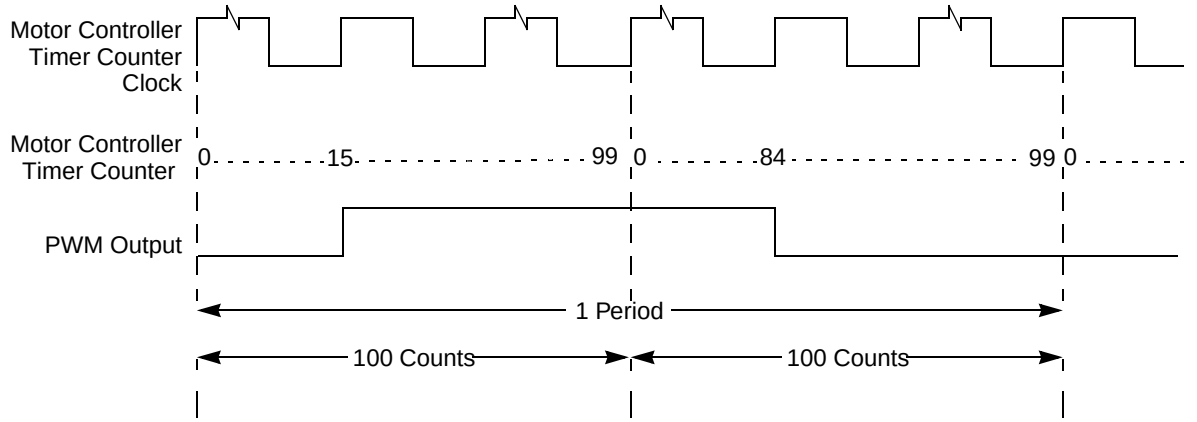


Figure 35-31. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x3, MCDCx[DUTY] = 31, MCPER[PER] = 200, MCCTL1[RECIRC] = 0

35.4.2 PWM Duty Cycle

The PWM duty cycle for the SMC channel x can be determined by dividing the decimal representation of the bits MCDCx[DUTY] by the decimal representation of the bits MCPER[PER] and multiplying the result by 100% as shown in Equation 35-1.

$$\text{Effective PWM Channel X \% Duty Cycle} = \frac{\text{DUTY}}{\text{MCPER}} \cdot 100\% \qquad \textit{Eqn. 35-1}$$

NOTE

x = PWM Channel Number = 0, 1, 2, 3 ... 11. This equation is only valid if MCDCx[DUTY] <= MCPER[PER] and MCPER[PER] is not equal to 0.

Whenever MCDCx[DUTY] >= MCPER[PER], a constant low level (MCCTL1[RECIRC] = 0) or high level (MCCTL1[RECIRC] = 1) will be output.

35.4.3 Motor Controller Counter Clock Source

Figure 35-32 shows how the PWM motor controller timer counter clock source is selected.

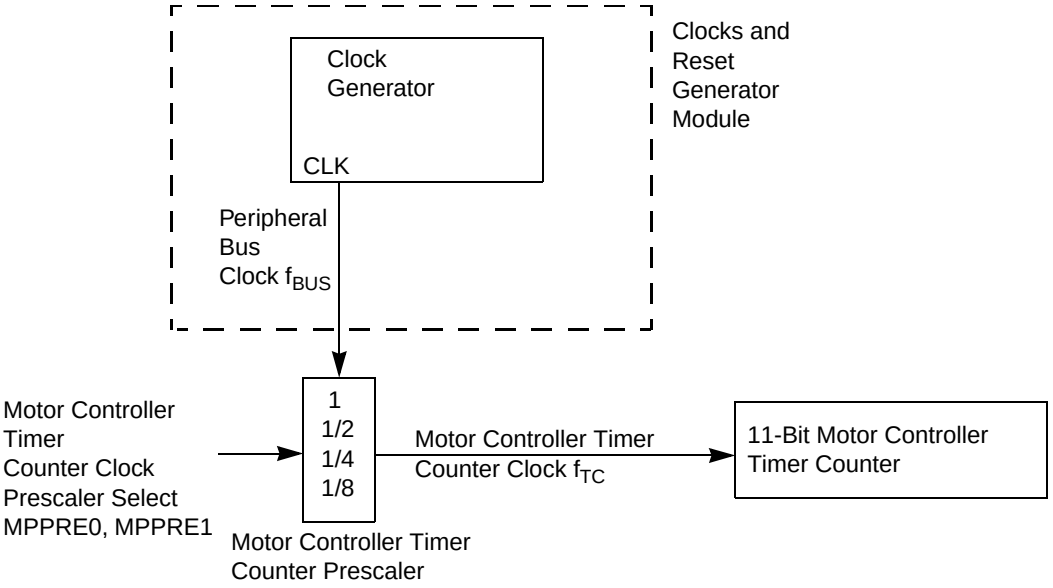


Figure 35-32. Motor Controller Counter Clock Selection

The peripheral bus clock is the source for the motor controller counter prescaler. The motor controller counter clock rate, f_{TC}, is set by selecting the appropriate prescaler value. The prescaler is selected with the MCCTL0[MCPRE] bits. The SMC channel frequency of operation can be calculated using Equation 35-2 if MCCTL0[DITH] = 0.

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{\text{MCPER} \cdot \text{M}} \qquad \textit{Eqn. 35-2}$$

The SMC channel frequency of operation can be calculated using Equation 35-3 if MCCTL0[DITH] = 1.

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{\text{MCPER} \cdot \text{M}/2} \qquad \textit{Eqn. 35-3}$$

NOTE

Both equations are only valid if MCPER[PER] is not equal to 0. M = 1 for left or right aligned mode, M = 2 for center aligned mode.

Table 35-22 shows examples of the SMC channel frequencies that can be generated based on different peripheral bus clock frequencies and the prescaler value.

Table 35-22. SMC Channel Frequencies (Hz), MCPER[PER] = 256, MCCTL0[DITH] = 0, MCCCx[MCAM] = 0x2, 0x1

Prescaler	Peripheral Bus Clock Frequency				
	16 MHz	10 MHz	8 MHz	5 MHz	4 MHz
1	62500	39063	31250	19531	15625
1/2	31250	19531	15625	9766	7813
1/4	15625	9766	7813	4883	3906
1/8	7813	4883	3906	2441	1953

NOTE

Due to the selectable slew rate control of the outputs, clipping may occur on short output pulses.

35.4.4 Output switching delay

In order to prevent large peak current draw from the motor power supply, selectable delays can be used to stagger the high logic level to low logic level transitions on the SMC outputs. The timing delay, t_d, is determined by the MCCCx[CD] bits in the corresponding channel control register and is selectable between 0, 1, 2, or 3 motor controller timer counter clock cycles.

NOTE

A PWM channel gets disabled at the next timer counter overflow without notice of the switching delay.

35.4.5 Operation in SMC stop mode

All module clocks are stopped and the associated port pins are set to their inactive state, which is defined by the state of the MCCTL1[RECIRC] bit. The SMC module registers stay the same as they were prior to entering stop mode. Therefore, after exiting from stop mode, the associated port pins will resume to the same functionality they had prior to entering stop mode.

35.4.6 Short-circuit detection

Each PWM pin is equipped with a short-circuit detection function. Hence, 24 instances (4 for each PWM module) of the short-circuit detector exist.

Table 35-23. Cross-reference PWM Signal to Short-circuit Detector Register Bits

Short-Circuit Detector Index *sd*	PWM Channel	Pin Name	Related SD Enable Bit (see 35.3.2.7)	Related SD Int Enable Bit (see 35.3.2.10)	Related SD Int Bit (see 35.3.2.13)
23	10	M5C0M	MCSDE2[SDEN[7]]	MCSDIEN2[SDIE[7]]	MCSDI2[SDIF[7]]
22	8	M4C0M	MCSDE2[SDEN[6]]	MCSDIEN2[SDIE[6]]	MCSDI2[SDIF[6]]
21	6	M3C0M	MCSDE2[SDEN[5]]	MCSDIEN2[SDIE[5]]	MCSDI2[SDIF[5]]
20	4	M2C0M	MCSDE2[SDEN[4]]	MCSDIEN2[SDIE[4]]	MCSDI2[SDIF[4]]
19	2	M1C0M	MCSDE2[SDEN[3]]	MCSDIEN2[SDIE[3]]	MCSDI2[SDIF[3]]
18	0	M0C0M	MCSDE2[SDEN[2]]	MCSDIEN2[SDIE[2]]	MCSDI2[SDIF[2]]
17	11	M5C1M	MCSDE2[SDEN[1]]	MCSDIEN2[SDIE[1]]	MCSDI2[SDIF[1]]
16	9	M4C1M	MCSDE2[SDEN[0]]	MCSDIEN2[SDIE[0]]	MCSDI2[SDIF[0]]
15	7	M3C1M	MCSDE1[SDEN[7]]	MCSDIEN1[SDIE[7]]	MCSDI1[SDIF[7]]
14	5	M2C1M	MCSDE1[SDEN[6]]	MCSDIEN1[SDIE[6]]	MCSDI1[SDIF[6]]
13	3	M1C1M	MCSDE1[SDEN[5]]	MCSDIEN1[SDIE[5]]	MCSDI1[SDIF[5]]
12	1	M0C1M	MCSDE1[SDEN[4]]	MCSDIEN1[SDIE[4]]	MCSDI1[SDIF[4]]
11	10	M5C0P	MCSDE1[SDEN[3]]	MCSDIEN1[SDIE[3]]	MCSDI1[SDIF[3]]
10	8	M4C0P	MCSDE1[SDEN[2]]	MCSDIEN1[SDIE[2]]	MCSDI1[SDIF[2]]
9	6	M3C0P	MCSDE1[SDEN[1]]	MCSDIEN1[SDIE[1]]	MCSDI1[SDIF[1]]
8	4	M2C0P	MCSDE1[SDEN[0]]	MCSDIEN1[SDIE[0]]	MCSDI1[SDIF[0]]
7	2	M1C0P	MCSDE0[SDEN[7]]	MCSDIEN0[SDIE[7]]	MCSDI0[SDIF[7]]
6	0	M0C0P	MCSDE0[SDEN[6]]	MCSDIEN0[SDIE[6]]	MCSDI0[SDIF[6]]
5	11	M5C1P	MCSDE0[SDEN[5]]	MCSDIEN0[SDIE[5]]	MCSDI0[SDIF[5]]
4	9	M4C1P	MCSDE0[SDEN[4]]	MCSDIEN0[SDIE[4]]	MCSDI0[SDIF[4]]
3	7	M3C1P	MCSDE0[SDEN[3]]	MCSDIEN0[SDIE[3]]	MCSDI0[SDIF[3]]
2	5	M2C1P	MCSDE0[SDEN[2]]	MCSDIEN0[SDIE[2]]	MCSDI0[SDIF[2]]
1	3	M1C1P	MCSDE0[SDEN[1]]	MCSDIEN0[SDIE[1]]	MCSDI0[SDIF[1]]
0	1	M0C1P	MCSDE0[SDEN[0]]	MCSDIEN0[SDIE[0]]	MCSDI0[SDIF[0]]

Each single short-circuit detector is a timer, measuring the time during which the signals PWM and FB are not equal (see Figure 35-33). If this time is greater than or equal to the time represented by MCSDTO[TOUT], then a short-circuit is assumed.

To enable the short-circuit detector on a pin, ensure that the pin's Input Buffer is enabled in the corresponding Pad Configuration Register in the SIU module.

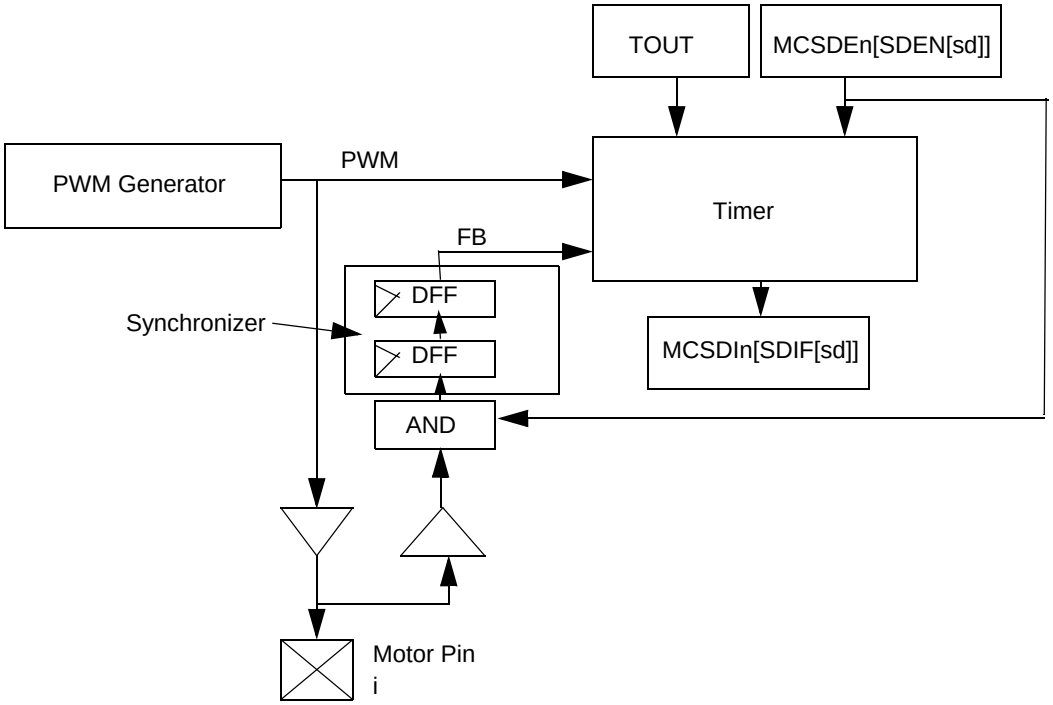
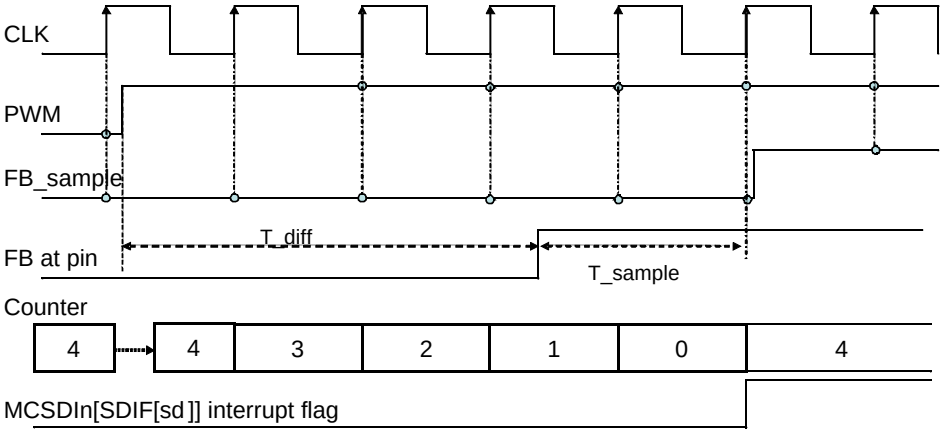


Figure 35-33. Short-circuit Detector Overview



T_sample	:Sample and synchronization delay
T_diff	:Delay between FB at pin and the internal PWM signal
PWM	:Signal from motor controller
FB at pin	:This is the signal directly after the input driver of the pad s ignal
FB_Sample	:Pad signal after sampling and synchronization

Figure 35-34. Example for MCSDTO[TOUT] = 4

- Hence, if PWM ≠ FB, the timer starts counting if the short-circuit detector is enabled at least one clock cycle before (see MCSDE0, MCSDE1 or MCSDE2)
- If PWM = FB, the timer stops and is reset to the timeout value MCSDTO[TOUT] in order to be ready for the next transaction

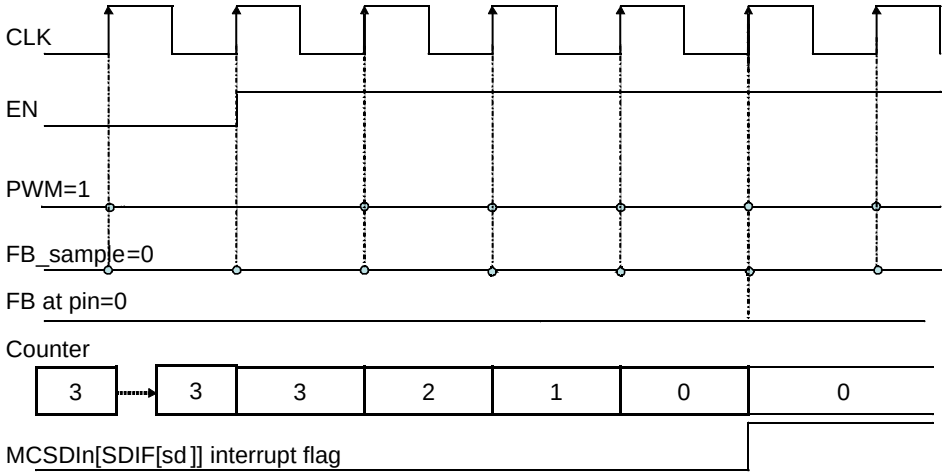
- IF PWM ≠ FB and the timer state is larger or equal MCSDTO[TOUT], than
  - One of the interrupt flags MCSDI0[SDIF], MCSDE1[SDIF] or MCSDI2[SDIF] is set in order to flag a short-circuit.
  - The interrupt flag is cleared by writing one, by reset or by disabling the short-circuit detector
  - The timer is stopped and reloaded with the timeout value MCSDTO[TOUT] in order to be ready for the next transaction
- If a short-circuit detector is disabled (MCSDEn[SDEN[*sd*]] = 0), the related short-circuit detector counter is halted and preloaded with the register value of MCSDTO[TOUT]. The related bit in MCSDIn[SDIF[*sd*]] of this specific short-circuit detector is set to 0. This means that, if all short-circuit detectors are disabled, all bits of MCSDIn stay at 0. No interrupt from the detector can be generated independently of the interrupt mask in MCSDIENn.
- In case of low-power modes, the state of the short-circuit detector is frozen. After exit of the low-power mode, the short-circuit detector will resume operation from the previous state. If the short-circuit detector should restart with defined state (counter value = MCSDTO[TOUT], than the related detector shall be disabled and enabled again. This will reload the counter with the MCSDTO[TOUT] value and restart the short-circuit detector.

The maximum time span which the timer can cover depends on the clock frequency F of the main clock. The maximum delay D covered by the counter is D = MCSDTO[TOUT]* F. Due to sampling and synchronization of the feedback signal, the value of MCSDTO[TOUT] must always be larger than 2.

The two synchronizer stages imply also, that a a short-circuit with a duration of less than or equal to 2 clock cycles cannot be detected.

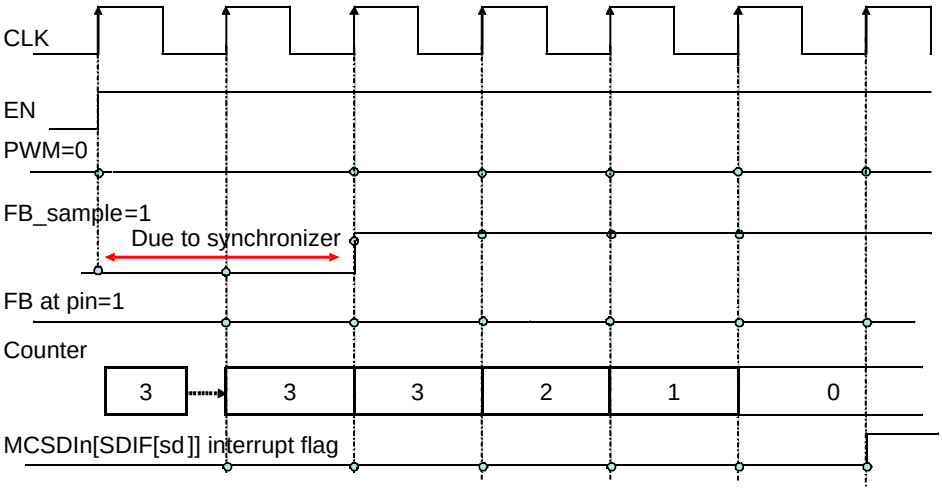
Two special cases shall be highlighted:

- Static short-circuit to ground and PWM signal = 1 see Figure 35-35: In this case, the enable of the short-circuit detector starts the sampling process and the interrupt bit is set MCSDTO[TOUT]+1 cycles after enabling the short-circuit detector
- Static short-circuit to VDD and PWM signal = 0 see Figure 35-36: In this case. the enable of the short-circuit detector starts the sampling process and the interrupt bit is set MCSDTO[TOUT]+3 cycles after enabling the short-circuit detector due to the synchronizer which has been cleared during disable of the short-circuit detector



T_sample :Sample and synchronization delay
T_diff :Delay between FB at pin and the internal PWM signal
PWM :Signal from motor controller
FB at pin :This is the signal directly after the input driver of the pad s ignal
FB_Sample :Pad signal after sampling and synchronization

Figure 35-35. Static Short-circuit, PWM signal always at 1 and FB always at 0, MCSDTO[TOUT]=3



T_sample :Sample and synchronization delay
T_diff :Delay between FB at pin and the internal PWM signal
PWM :Signal from motor controller
FB at pin :This is the signal directly after the input driver of the pad s ignal
FB_Sample :Pad signal after sampling and synchronization

Figure 35-36. Static Short-circuit, PWM signal always at 0 and FB always at 1, MCSDTO[TOUT]=3

NOTE

The short-circuit detection block does not disable a port in case of a short-circuit. It is task of the microcontroller to manage the event of a short-circuit.

35.5 Reset

The SMC is reset by system reset. All associated ports are released, all registers of the SMC module will switch to their reset state as defined in Section 35.3.2, Register description.

35.6 Interrupts

The SMC has one interrupt output which is the bitwise OR function of 25 individual interrupt request sources:

- One time counter overflow interrupt: An interrupt will be requested when the MCCTL1[MCTOIE] bit in is set and the running PWM frame is finished. The interrupt is cleared by either setting the MCCTL1[MCTOIE] bit to 0 or to write a one to the MCCTL0[MCTOIF] bit
- 24 Interrupts for the short-circuit detection, one for each PWM pin: Whenever a short-circuit is detected on one PWM pin and the short-circuit detector enable bit MCSDEn[SDEN[*sd*]] is set, than the related interrupt flag MCSDIn[SDIF[*sd*]] is set according to the mapping shown inTable 35-23. The interrupt flag in MCSDIn[SDIF[*sd*]] will also rise an external interrupt if the interrupt enable bit MCSDIENn[SDIE[*sd*]] is set. To clear the interrupt flag, either write a one into the related bit position MCSDIn[SDIF[*sd*]] or disable the related short-circuit detector by writing zero to MCSDEn[SDEN[*sd*]]. If the short-circuit detector is enabled and a static short-circuit exists, then the MCSDIn[SDIF[*sd*]] flag will be asserted directly after clearing it, because the short-circuit detector is still enabled and will detect the short-circuit again. To avoid this behavior, disable the short-circuit detector channel after detection of the short-circuit.

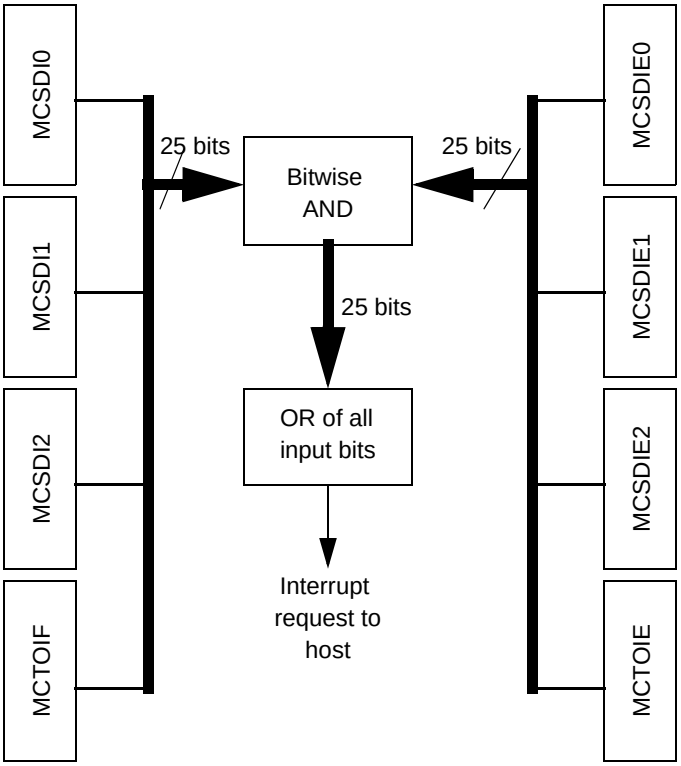


Figure 35-37. SMC Interrupt Generation

Chapter 36
Stepper Stall Detect (SSD)

36.1 Introduction

36.1.1 Overview

The SSD block connects to one stepper motor (SM) with two coils. It can be used to monitor the movement of the SM to detect that the attached gauge pointer has reached the stall position of the scale.

Basis of the movement detection is to drive one of the coils and to integrate the back EMF (electromotive force) induced in the other coil. This back EMF is present only if the SM is rotating. Therefore, if the integral of the back EMF exceeds a certain threshold, the SM is still rotating; otherwise it can be regarded as being stalled.

The SSD block shares the pins connected to the SM coils together with the motor controller block responsible for driving the SM in the main application (e.g. moving the gauge pointer to a certain position of the scale).

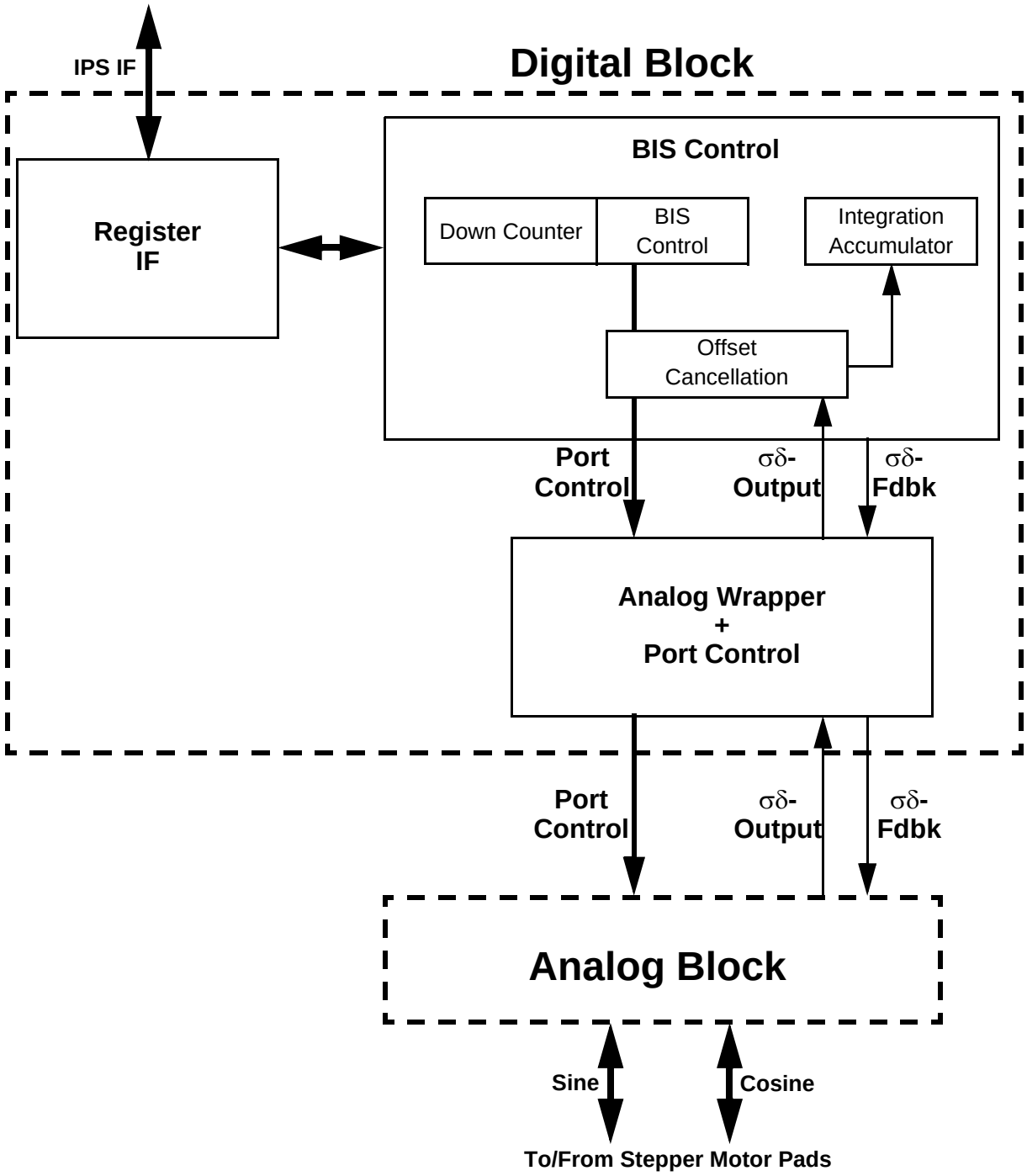


Figure 36-1. SSD overall block diagram

The names of the sub blocks given in the diagram above relate to the descriptions given in Section 36.4.1, Main building blocks of the SSD:

- 'Analog Block' relates to Section 36.4.1.1, Analog block.
- 'Analog Wrapper and Port Control' relates to Section 36.4.1.2, Analog Wrapper + Port Control.

- ‘Register Interface’ relates to Section 36.4.1.3, Register Interface.
- ‘BIS (blanking-integration sequence) Control Logic’ relates to Section 36.4.1.4, BIS control.

36.1.2 Features

The most important features of the SSD block are listed below:

- Programmable Full Step States according to 2-coil stepper motors (4 states)
- Programmable Integration Polarity
- Integration Accumulator of 16 bits with programmable clock divider.
- Programmable Down Counter (16 bit timer)
  - 64 MHz bus clock: Finest resolution in time down to 125 ns, maximum length of blanking or integration phase for this case is ~8.2 ms
  - 64 MHz bus clock: Maximum length of blanking or integration phase is 1.05 s, resolution in time reduced to 16 µs for this case
- Automatic sequence of blanking followed by integration (BIS) triggered by the user.
- Full flexibility over blanking and integration phase of the BIS:
  - Separate down counter initialization values and divider factors
  - Separate interrupt flags and interrupt enable bits
  - Separate coil drive enable bits
- Polarity switching to cancel DC offset errors programmable. The down counter value for the integration phase can be divided by 2, 4, or 8 to switch the polarity during the integration phase. Additionally the offset cancellation can be switched off completely
- Seamless changeover into or from SSD (stepper stall detect) mode: The coil control signals outside of the BIS are fully programmable.

36.1.3 Modes of operation

This section describes the different modes of operation of the SSD block.

36.1.3.1 Disabled mode

In this mode the SSD block is disabled. None of the SM coils is driven nor one of the coils sensed. This is the case if the RTZE bit in the CONTROL register is cleared.

36.1.3.2 Normal mode

In this mode the SSD block needs exclusive control over the SM coils. This must be ensured on at the device level. Setting the RTZE bit in the CONTROL register is not sufficient since the SSD does not provide a global port enable signal. Refer to Section 36.4, Functional description, for more details.

36.1.3.3 Stop mode

The CPU can activate Stop mode. In Stop mode, the system clocks driving the SSD block are stopped. In Stop mode, the analog block is also disabled.

36.2 External signal description

Each of the four analog I/Os of the SSD block is used either as the output of a half-bridge sourcing or sinking the current of the SM coil driven currently or —in case of acting as an input—the back EMF of the non-driven coil with respect to an internally generated reference voltage is supplied to the σδ-modulator of the analog block. The signal properties are given in Table 36-1.

Table 36-1. Signal properties

Name	Port	Coil	Coil Node	I/O	Reset
COSP	COSP	Cosine	Plus	Analog I/O	Z
COSM	COSM		Minus	Analog I/O	Z
$\overline{\text{SINP}}$	SINP	Sine	Plus	Analog I/O	Z
SINM	SINM		Minus	Analog I/O	Z

36.3 Memory map and register definition

This section provides a detailed description of the registers of the SSD block. Note that all registers are 16 bits in width. There is no access on byte level.

36.3.1 Memory map

Table 36-2 lists the registers of the SSD block.

Table 36-2. Block memory map

Offset	Register Name (Long)	Register Name (Short)	Access[1]	Location
0x00	SSD Control and Status Register	CONTROL	R/W	on page 1161
0x02	SSD Interrupt Flag and Enable Register	IRQ	R/W	on page 1162
0x04	SSD Integrator Accumulator Register	ITGACC	R	on page 1163
0x06	SSD Down Counter Count register	DCNT	R	on page 1163
0x08	SSD Blanking Counter Load Register	BLNCNTLD	R/W	on page 1164
0x0A	SSD Integration Counter Load Register	ITGCNTLD	R/W	on page 1164
0x0C	SSD Prescaler Register	PRESCALE	R/W	on page 1165
0x0E	RESERVED[2]		n/a	—

1 Note that read/write registers may contain some read-only or write-only bits.

2 Read access provides 0x0000. No write allowed.

36.3.2 Register descriptions

This section describes the individual bits of all the SSD registers. Note that the details of the functional description linked to these bits is given in Section 36.4, Functional description.

36.3.2.1 SSD Control and Status Register (CONTROL)

Figure 36-2 below describes the fields of the main control (CONTROL) register:

Offset: 0x00 Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	STEP		RCIR	ITGDIR	BLNDCL	ITGDCL	RTZE	0	BLNST	ITGST	0	0	0	SDCPU	0
W	TRIG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-2. SSD Control and Status Register (CONTROL)

The function of the CONTROL register bits is shown in Table 36-3.

Table 36-3. CONTROL Register field description

Field	Description
15 TRIG	Trigger Blanking → Integration sequence (BIS). 0 No effect. 1 Sequence of blanking → integration is triggered.
14–13 STEP	Full Step State. These bits determine which coil is driven for SM movement,. Refer to Table 36-10 for details of the step states. 00 Select 0° angle (east pole) state for the electromagnetic field in the SM. 01 Select 90° angle (north pole) state for the electromagnetic field in the SM. 10 Select 180° angle (west pole) state for the electromagnetic field in the SM. 11 Select 270° angle (south pole) state for the electromagnetic field in the SM.
12 RCIR	Blanking Polarity for coil recirculation. Refer to Section 36.4, Functional description, for details of the recirculation mode. 0 Coil recirculation via high side transistors (VDDM, analog supply voltage). 1 Coil recirculation via low side transistors (VSSM, analog GND).
11 ITGDIR	Direction (polarity) of integration. Refer to Section 36.4.1.4.3, DC Offset Cancellation, for details
10 BLNDCL	Drive Coil during Blanking. 0 During the BIS blanking phase the other coils is not driven by the SSD. The SM will not move during blanking. 1 During the BIS blanking phase the other coil is actually driven by the SSD block (genuine use case).
9 ITGDCL	Drive Coil during Integration and outside of any BIS. 1 During the BIS integration phase and outside of any BIS the other coil is actually driven by the SSD block (genuine use case). Outside of any BIS the same coil is driven. 0 During the BIS integration phase the other coils is not driven by the SSD. Outside of any BIS no coil is driven. The SM will not move during integration (not useful for SSD).

Table 36-3. CONTROL Register field description (continued)

Field	Description
8 RTZE	Return to Zero Enable. This is in fact the enable bit of the SSD logic to take over control of the SM coils[1]. 1 Control of the SM coils by the SSD block is enabled. 0 Control of the SM coils by the SSD block is disabled.
6 BLNST	Blanking Status. Refer to Section 36.1.3.2, Normal mode, for details. 1 The SSD block is currently in the blanking phase of an ongoing BIS. 0 The SSD block is not in the blanking phase of an ongoing BIS.
5 ITGST	Integration Status. Refer to Section 36.1.3.2, Normal mode, for details. 1 The SSD block is currently in the integration phase of an ongoing BIS. 0 The SSD block is not in the integration phase of an ongoing BIS.
1 SDCPU	Sigma-Delta modulator Power Up. Setting this bit enables the analog block of the SSD and enables the clocking of the port control logic of the digital part. 1 Analog block of the SSD is enabled. 0 Analog block of the SSD is not enabled
0	Reserved

1 The application must switch off any other blocks possibly interfering with port control of the SSD block.

36.3.2.2 Interrupt Enable and Flag Register (IRQ)

Figure 36-3 below describes the fields of the interrupt enable and flag (IRQ) register:

Offset 0x02 Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BLNIF	ITGIF	0	0	0	0	0	ACOVIF	BLNIE	ITGIE	0	0	0	0	0	ACOVIE
W	w1c	w1c						w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-3. SSD Interrupt Flag and Enable Register (IRQ)

The function of the IRQ register bits is shown in Table 36-4.

Table 36-4. IRQ Register field description

Field	Description
15 BLNIF	Blanking expired Interrupt Flag. 1 This flag is set when the BIS blanking phase has expired. 0 No such event.
14 ITGIF	Integration expired Interrupt Flag. 1 This flag is set when the BIS integration phase has expired. 0 No such event.
8 ACOVIF	Accumulator Overflow Interrupt Flag. 1 This flag is set when during the BIS integration phase the integration logic attempted either to increment the ITGACC register above 0x7FFF or to decrement it below 0x8000. 0 No such event.

Table 36-4. IRQ Register field description (continued)

Field	Description
7 BLNIE	Blanking expired Interrupt Enable. 1 A module interrupt will occur if the BLNIF bit is set. 0 The BLNIF flag will not trigger an interrupt on the ips_int output.
6 ITGIE	Integration expired Interrupt Enable. 1 A module interrupt will occur if the ITGIF bit is set. 0 The ITGIF flag will not trigger an interrupt on the ips_int output.
0 ACOVIE	Accumulator Interrupt Enable. 1 A module interrupt will occur if the ACOVIF bit is set. 0 The ACOVIF flag will not trigger an interrupt on the ips_int output.

36.3.2.3 Integration Accumulator Register (ITGACC)

Figure 36-4 below describes the fields of the integration accumulator (ITGACC) register:

Offset 0x04 Access: User read/write

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

Figure 36-4. SSD Integration Accumulator Register (ITGACC)

The function of the ITGACC register bits is shown in Table 36-5.

Table 36-5. ITGACC Register field description

Field	Description
15–0 ITGACC	Integration Accumulator readout value. This 2's complement register represents the accumulator register of the back EMF integrator of the SSD block. Refer to the functional description of the integrator for further details.

36.3.2.4 Down Counter Register (DCNT)

Figure 36-5 below describes the fields of the down counter (DCNT) register:

Offset 0x06 Access: User read/write

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

Figure 36-5. SSD Down Counter Register (DCNT)

The function of the DCNT register bits is shown in Table 36-6.

Table 36-6. DCNT Register field description

Field	Description
15–0 DCNT	Down Counter value. This register represents the actual value of the down counter in unsigned format. Refer to the functional description of the integrator for further details.

36.3.2.5 Blanking Counter Load Register (BLNCNTLD)

Figure 36-6 below describes the fields of the blanking counter load (BLNCNTLD) register:

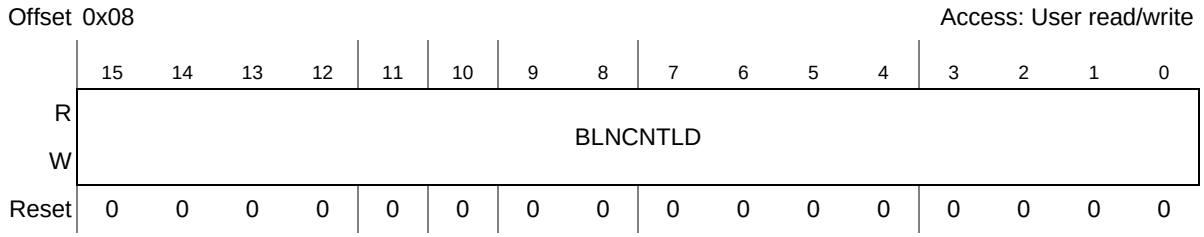


Figure 36-6. SSD Blanking Counter Load Register (BLNCNTLD)

The function of the BLNCNTLD register bits is shown in Table 36-7.

Table 36-7. BLNCNTLD Register field description

Field	Description
15–0 BLNCNTLD	Blanking Count Load value. This register is programmed with the number of down counter periods belonging to the blanking phase of the following BISs. Number format is unsigned. Refer to the Functional Description of the integrator for further details. Programming all 0's into the BLNCNTLD register bits disables blanking completely.

36.3.2.6 Integration Counter Load Register (ITGCNTLD)

Figure 36-7 below describes the fields of the integration counter load (ITGCNTLD) register:

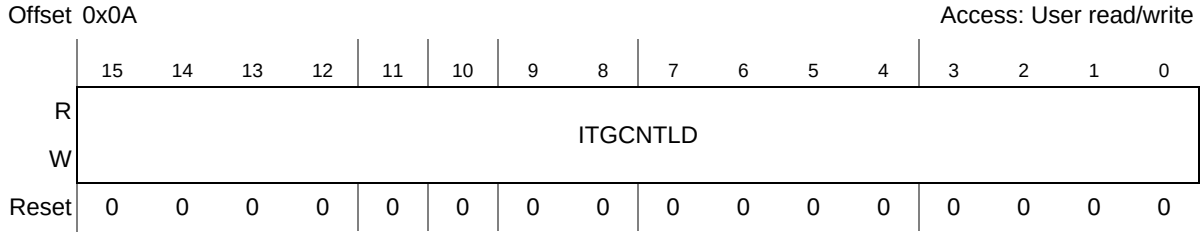


Figure 36-7. SSD Integration Counter Load Register (ITGCNTLD)

The function of the ITGCNTLD register bits is shown in Table 36-8.

Table 36-8. ITGCNTLD Register field description

Field	Description
15–0 ITGCNTLD	Integration Count Load value. This register is programmed with the number of down counter periods belonging to the integration phase of the following BISs. Number format is unsigned. Refer to the functional description of the integrator for further details. Programming all 0's into the ITGCNTLD register bits disables integration completely.

36.3.2.7 SSD Prescale and Divider Register (PRESCALE)

Figure 36-8 below describes the fields of the prescale and divider factor (PRESCALE) register:

Offset 0x0C — Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		BLNDIV			0	ITGDIV			0	0	OFFCNC		0	ACDIV		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-8. SSD Prescale and Divider Factor Register (PRESCALE)

The function of the PRESCALE register bits is shown in Table 36-9 below:

Table 36-9. PRESCALE Register field description

Field	Description
14–12 BLNDIV	Blanking Counter Clock Divider Select. The frequency for updating the down counter in the blanking phase of the next BISs is derived from the bus clock according to the formula <down counter clock> = <bus clock> / ($8 * 2^{BLNDIV}$) According to this formula the divider factors are: 000 8 001 16 010 32 011 64 100 128 101 256 110 512 111 1024
10–8 ITGDIV	Integration Counter Clock Divider Select. The frequency for updating the down counter in the integration phase of the next BISs is derived from the bus clock according to the formula <down counter clock> = <bus clock> / ($8 * 2^{ITGDIV}$) According to this formula the divider factors are: 000 8 001 16 010 32 011 64 100 128 101 256 110 512 111 1024
5–4 OFFCNC	Offset Cancellation polarity flip select. Refer to Section 36.4.1.4.3, DC Offset Cancellation, for details of the offset cancellation mechanism. The OFFCNC bits set the preset value of the internal counter which determines the polarity flips during the integration phase. The preset value is derived from the ITGCNTLD register value with the following divider factor: 00 0: Selected polarity remains unchanged for all the time of the integration phase. 01 2: First polarity switch (and possibly a succeeding one) occurs after [ITGCNTLD div 2] DCNT ticks. 10 4: First polarity switch and succeeding ones occur after [ITGCNTLD div 4] DCNT ticks. 11 8: First polarity switch and succeeding ones occur after [ITGCNTLD div 8] DCNT ticks. If the ITGCNTLD register value cannot be divided by the required factor, an additional polarity flip occurs with a duration corresponding to the bits shifted out.

Table 36-9. PRESCALE Register field description (continued)

Field	Description
2–0 ACDIV	Accumulator Sample Clock Divider Select. The accumulator sample clock is derived from the bus clock according to the formula <accumulator sample clock> = <bus clock> / (8 * 2^{ACDIV}) According to this formula the divider factors are: 000 8 001 16 010 32 011 64 100 128 101 256 110 512 111 1024 The first ITGACC register update occurs when (8 * 2^{ACDIV}) bus clocks have expired after the ITGST bit has been set by the SSD block.

36.4 Functional description

For all the descriptions given here it is assumed that the SSD block has gained exclusive control of the SM coils and the analog block is enabled appropriately.

36.4.1 Main building blocks of the SSD

The functional description given in this chapter deals with the main functional blocks. It concentrates on the description of the implemented functionality. Refer to Figure 36-1 for details.

36.4.1.1 Analog block

An overview of the analog block of the SSD block is given in Figure 36-9 below. Additionally the most important sub blocks of the digital part which are connected to the analog blocks are shown in order to clarify the joint operation of the analog block and the digital part.

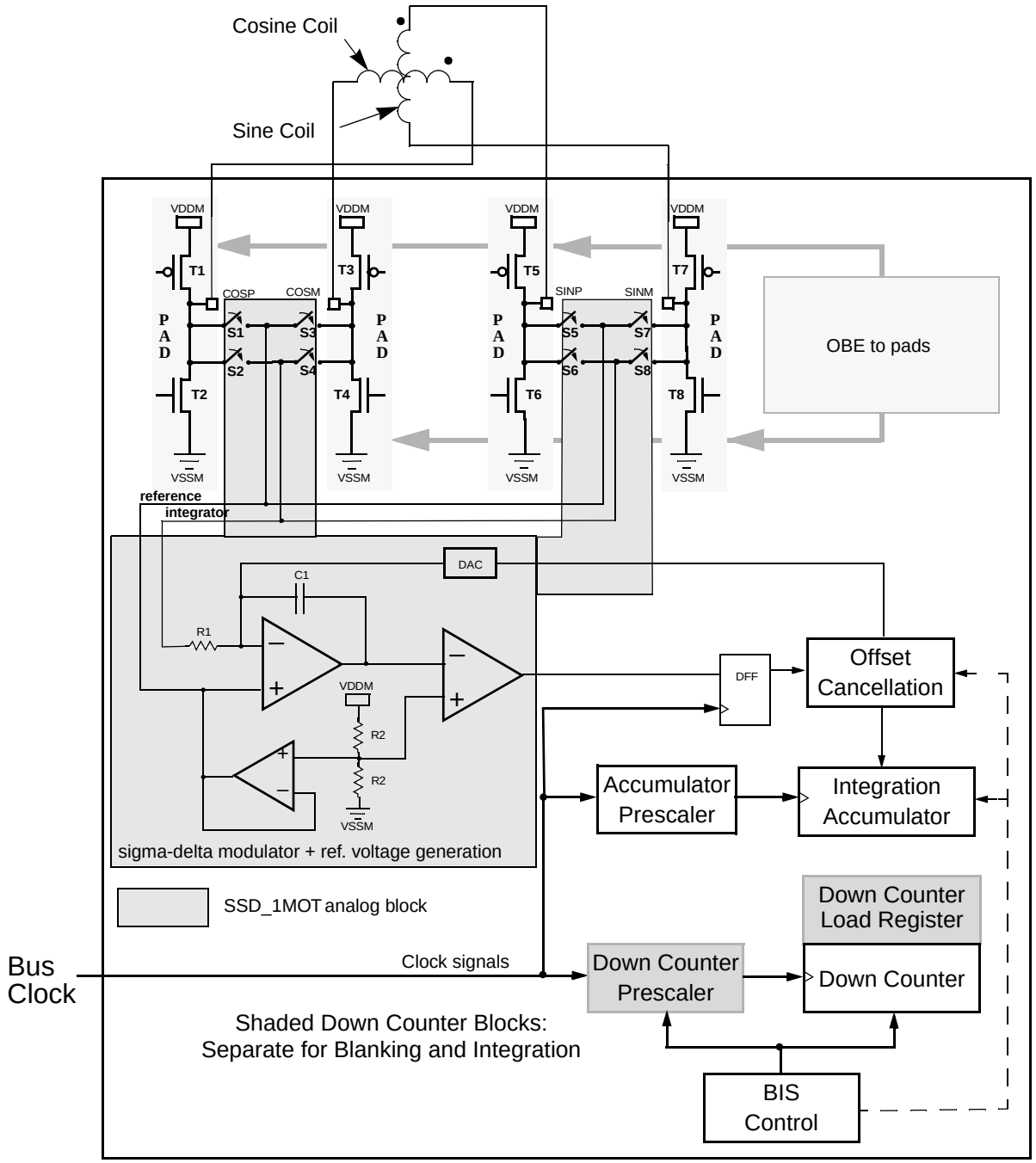


Figure 36-9. SSD block diagram, analog block

Main part of the analog block is the σδ-modulator. It is operational during the BIS integration phase only (step 5 in Section 36.4.2.2, Details of the SSD Measurement). The clock to update the feedback path is derived from the bus clock using the ACDIV setting. The 1-bit output value provided to the digital part is used to increment or decrement the ITGACC register.

For the correct movement of the SM the sine and cosine coil connections to VDDM and VDDS need to be set properly, depending from the angular position. This is achieved by enabling and disabling of the pad transistors T1 to T8. These pad transistors are not part of the SSD block, only the OBE (output buffer enable) signals for the pads are provided. For the switch characteristics refer to Section 36.4.1.2.1, Transistor Condition States.

Aside from the pad transistors the switches S1 to S8 determine which coil provides the back EMF to the integrator in which polarity w.r.t. the reference voltage. The switches are implemented in the analog block of the SSD (denoted by shading them in the same manner like the analog block).

The σδ-modulator is enabled by setting the SDCPU bit in the CONTROL register. To compensate for switching effects of the analog circuitry the user must take into account sufficient startup time, described in Section 36.5.1, Analog Block Startup Time, prior to starting the BIS.

36.4.1.2 Analog Wrapper + Port Control

This sub block controls the outputs to the coils and the inputs to the analog block. Refer to Figure 36-9 for details.

The most relevant bits of the CONTROL register belonging to that functional block are:

- STEP bits: These 2 bit vector has two functions.

  One function is to determine which coil is driven and which coil is connected to the σδ-modulator. Additionally the direction of the current flow is selected with these bits. For clockwise direction of the SM movement the value must be decremented and for counter-clockwise movement it must be incremented when advancing from one STEP setting to the next.
- BLNDCL: This bit is the enable of the supply voltage to be routed to the coil driven in the appropriate STEP setting during the blanking phase of an ongoing BIS.
- ITGDCL: This bit is the enable of the supply voltage to be routed to the coil driven in the appropriate STEP setting during the integration phase of an ongoing BIS. Additionally it determines the coil drive setting outside of any BIS.
- ITGDIR: This bit is relevant only in the integration phase. Together with the STEP bits the polarity of the integration is determined by enabling or disabling the appropriate analog switches. Refer to Section 36.4.1.4.3, DC Offset Cancellation.

These control bits are translated into the appropriate switching scheme of the pad transistors and the σδ-modulator switches described below. Note that it must be ensured at the device level that the SSD block has exclusive control over the analog pads connected to the SM coils.

Additionally it is a precondition that the RTZE bit in the CONTROL register is set.

36.4.1.2.1 Transistor Condition States

The pad transistors T1 to T8 are responsible for connecting the SM coils to the analog supply voltages VDDM and VDDS. Table 36-10 below shows the pad transistor condition states implemented in the analog block. In the table the columns have the following meaning:

- STEP denotes the setting of the corresponding bits in the CONTROL register.
- ITGST reflects the status indicator of the BIS integration phase.

- ‘Resulting DCOIL’ is the (BIS state dependent) resulting coil drive setting. See Table 36-11 for how this setting is derived from the value of the CONTROL register.
- RCIR denotes the setting of the corresponding bit in the CONTROL register.

Table 36-10. Transistor Condition States[1]

STEP	ITGST	Resulting DCOIL	RCIR	T1	T2	T3	T4	T5	T6	T7	T8	Remarks
xx	1	0	x	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	Integration with no drive
00	0	0	0	OFF	OFF	OFF	OFF	ON	OFF	ON	OFF	Blanking with no drive (RCIR bit determines supply voltage for recirculation)
00	0	0	1	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	
01	0	0	0	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF	
01	0	0	1	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF	
10	0	0	0	OFF	OFF	OFF	OFF	ON	OFF	ON	OFF	
10	0	0	1	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	
11	0	0	0	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF	
11	0	0	1	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF	
00	0	1	0	ON	OFF	OFF	ON	ON	OFF	ON	OFF	Cosine coil driven P → M
00	0	1	1	ON	OFF	OFF	ON	OFF	ON	OFF	ON	
00	1	1	x	ON	OFF	OFF	ON	OFF	OFF	OFF	OFF	
01	0	1	0	ON	OFF	ON	OFF	ON	OFF	OFF	ON	Sine coil driven P → M
01	0	1	1	OFF	ON	OFF	ON	ON	OFF	OFF	ON	
01	1	1	x	OFF	OFF	OFF	OFF	ON	OFF	OFF	ON	
10	0	1	0	OFF	ON	ON	OFF	ON	OFF	ON	OFF	Cosine coil driven M → P
10	0	1	1	OFF	ON	ON	OFF	OFF	ON	OFF	ON	
10	1	1	x	OFF	ON	ON	OFF	OFF	OFF	OFF	OFF	
11	0	1	0	ON	OFF	ON	OFF	OFF	ON	ON	OFF	Sine coil driven M → P
11	0	1	1	OFF	ON	OFF	ON	OFF	ON	ON	OFF	
11	1	1	x	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF	

1 ‘x’ means ‘Don’t care’

Table 36-11 below shows the logic dependency of the resulting coil drive from the internal state of the SSD block and from the setting of the different coil control bits in the CONTROL register:

Table 36-11. Generation of the Resulting DCOIL

BLNST	BLNDCL	ITGST	ITGDCL	Resulting DCOIL	Remarks
0	x	0	0	0	No running BIS ITGDCL determines result
			1	1	
1	0	0	x	0	Running BIS in blanking phase BLNDCL determines result
	1			1	
0	x	1	0	0	Running BIS in integration phase ITGDCL determines result
			1	1	

36.4.1.2.2 Switch condition states

The analog switches S1 to S8 are used to select the appropriate source and polarity of the non-driven coil into the σδ-modulator for integration during the integration phase of an ongoing BIS. Outside of the integration phase none of the switches is enabled. Refer to Table 36-12 below for details.

Table 36-12. Switch Condition States

ITGST	STEP (Register Bit)	Integration Polarity (Input to analog block)	S1	S2	S3	S4	S5	S6	S7	S8
0	xx	x	Open	Open	Open	Open	Open	Open	Open	Open
1	00	0	Open	Open	Open	Open	Close	Open	Open	Close
1	00	1	Open	Open	Open	Open	Open	Close	Close	Open
1	01	0	Open	Close	Close	Open	Open	Open	Open	Open
1	01	1	Close	Open	Open	Close	Open	Open	Open	Open
1	10	0	Open	Open	Open	Open	Open	Close	Close	Open
1	10	1	Open	Open	Open	Open	Close	Open	Open	Close
1	11	0	Close	Open	Open	Close	Open	Open	Open	Open
1	11	1	Open	Close	Close	Open	Open	Open	Open	Open

NOTE

The value given in the column ‘Integration Polarity’ in Table 36-12 is linked to the ITGDIR bit in the CONTROL register in the way described in Section 36.4.1.4.3, DC Offset Cancellation, below.

36.4.1.3 Register Interface

The register interface processes the IPS accesses from the device level. Access size is 32 bits, the SSD block supports 16- and 32-bit accesses.

Any write access on byte-level is ignored.

All reserved registers provide 0x0000 on read. Write access is not allowed.

36.4.1.4 BIS control

Once triggered the sequence control logic walks through a single individual BIS. In the normal application one BIS corresponds to a single step (90° movement of the SM). In detail the BIS is implemented in the SSD block in the following way:

- Each BIS starts with setting the TRIG bit. Ending a running BIS manually is only possibly by clearing the RTZE bit.
- If the BLNCNTLD register is set to a value other than 0x0000 the DCNT is loaded with (BLNCNTLD - 1) and is started using the BLNDIV bit setting for the clock divider. The BLNST is set.

  The blanking phase of the BIS is executed, the BLNDCL bit is used to determine whether one of the coils is driven during the blanking phase.

  If the appropriate number of down counter periods (equal to the BLNCNTLD register value) expires the BLNIF is set, the interrupt is triggered according to the BLNIE bit and the BLNST bit is cleared.
- If the ITGCNTLD register is set to a value other than 0x0000 the DCNT is loaded with (ITGCNTLD - 1) and is started using the ITGDIV bit setting for the clock divider. The ITGST is set and the ITGACC register is initialized with 0x0000.

  The integration phase of the BIS is executed, the ITGDCL bit is used to determine whether one of the coils is driven during the integration phase. The σδ-modulator of the analog block is functional and the ITGACC register is updated. During the integration phase the polarity is switched according to the OFFCNC bits.

  If the appropriate number of down counter periods (equal to the ITGCNTLD register value) expires the ITGIF is set, the interrupt is triggered according to the ITGIE bit and the ITGST bit is cleared.

The state of the ongoing BIS can be monitored by the following status bits:

- The BLNST bit is set during the blanking phase exclusively.
- The ITGST bit is set during the integration phase exclusively. When it is set, the BIS control enables the σδ-modulator in the analog block together with the integration circuitry. As long as the integration phase is active the ITGACC register content is modified depending from the output of the σδ-modulator.

In the normal use case the end of the BIS is the end of the integration phase. Independent from the time required by the software to detect and act upon the end of the BIS the ITGACC register is not changed after the end of the integration phase.

The sequence control logic itself makes use of the following sub blocks:

36.4.1.4.1 Down Counter

The down counter is basically a timer with the divider factor (BLNDIV or ITGDIV) and length (BLNCNTLD or ITGCNTLD) determined by the current state of the BIS. Additionally to defining the

length of the integration phase the ITGCNTLD setting determines the DCNT value to switch the integration polarity for DC Offset Cancellation depending from the OFFCNC bit setting.

When the down counter reaches 0x0000 and the associated divider period has expired the appropriate flag is set and the corresponding interrupt is triggered depending from the interrupt enable bit. Note that polling the DCNT register for 0x0000 is misleading since the DCNT time out is reached at the **end** of the divider period belonging to the value of 0x0000. Refer to Section 36.6.3, Watching Internal States of the SSD.

Table 36-13 below shows details how the BLNDIV/ITGDIV and BLNCNTLD/ITGCNTLD settings determine the length and granularity of the blanking and integration phase depending from the bus clock frequency.

Table 36-13. Blanking and Integration Phase Length Vs. Bus Clock[1]

Bus Clock	40 MHz		64 MHz		80 MHz	
BLNDIV/ITGDIV	**0**	**7**	**0**	**7**	**0**	**7**
Timing Granularity	0.2 µs	25.6 µs	0.125 µs	16 µs	0.1 µs	12.8 µs
Max. length of BLN/ITG	13.107 ms	1.678 s	8.192 ms	1.049 s	6.554 ms	0.839 s

1 Numbers rounded appropriately

36.4.1.4.2 Integration Accumulator

This is the fundamental sub block of the SSD, it is responsible for collecting the result of the back EMF integration from the σδ-modulator located in the analog block. The only time when the value of the accumulator can change is during the integration phase of a BIS.

In terms of signal processing the ITGACC register is the counterpart of the σδ-modulator in the analog block, working as the σδ-demodulator: Depending from the ACDIV bits in the PRESCALE register the output of the analog block is sampled periodically and the content of the accumulator incremented or decremented. Therefore the ITGACC register in fact counts the ‘imbalance’ between 1 and 0 output samples from the analog block.

The value of the ITGACC register can change only during the integration phase of an ongoing BIS. Before the first update the content is initialized to 0x0000 and starting from that it is incremented or decremented according to the σδ-modulator output.

Number format is two’s complement, if an overflow (attempt to increment ITGACC register value of 0x7FFF) or an underflow (attempt to decrement ITGACC register value of 0x8000) the ACOVIF bit indicates the over-/underflow condition, the SSD interrupt is triggered if the ACOVIE bit is set and the ITGACC register values is not changed. For the rest of the integration phase of the current BIS the ITGACC register value does not change. Reaching the accumulator end values **without** an over-/underflow condition does not prevent the ITGACC register from incrementing 0x8000 (–32768) or decrementing 0x7FFF (+32767).

Table 36-14 below shows details how the ACDIV setting determines the σδ-demodulator sampling clock w.r.t. the bus clock. The recommended setting for the sampling is a resulting clock between 500 kHz and 2 MHz. Therefore the ACDIV values for sampling clock values in this recommended range are given:

Table 36-14. ITGACC update clock vs. bus clock—recommended settings[1]

Bus Clock	40 MHz		64 MHz		80 MHz	
ACDIV	3'b010	3'b011	3'b011	3'b100	3'b011	3'b100
Divider factor	32	64	64	128	64	128
Sampling frequency	1.25 MHz	625 kHz	1.00 MHz	500 kHz	1.25 MHz	625 kHz
Update interval	0.8 µs	1.6 µs	1.00 µs	2.00 µs	0.8 µs	1.6 µs

1 Numbers rounded appropriately

36.4.1.4.3 DC Offset Cancellation

Due to deviations from the mid point of the analog supply voltages and other effects in the hardware of the analog blocks a DC offset may be introduced into the output of the σδ-modulator. As a consequence of such a DC offset the value obtained in the integration accumulator would depend from the 'direction' of the integration (e.g. accumulator increment for positive back EMF in clockwise movement). The DC offset cancellation implemented in the SSD block can eliminate (or at least reduce) the influence of such a DC offset:

When active the DC offset cancellation reverts two internal settings in the SSD block during the integration phase of the current BIS:

- The input into the analog block controlling the integration polarity which sets the switch condition state (third column in Table 36-12):

  Initial value (when the integration starts at step 5 in Section 36.4.2.2, Details of the SSD Measurement) is the bit value given in the ITGDIR register in the CONTROL register.
- The output of the σδ-modulator being applied to the integration accumulator (ITGACC register):

  Initially it is applied without change to the integration accumulator.

As a result the switch conditions in the analog circuitry change the direction of the voltage representing the back EMF measured by the σδ-modulator. But the change direction of the ITGACC register is maintained because the interpretation of the σδ-modulator output is reverted, too.

The offset cancellation is implemented as an additional counter running during the integration phase with the same clock setting like the DCNT register. The preset value for this counter is derived from the ITGCNTLD register by shifting right by 0, 1, 2 or 3 bits, depending from the OFFCNC bit setting. Clearing all the OFFCNC bits obviously disables the offset cancellation completely. Note that increasing the number of flips improves the offset cancellation because the different polarities are distributed equally over the complete integration phase (if ITGCNTLD can be divided by the appropriate number). Refer to Figure 36-10 for more details.

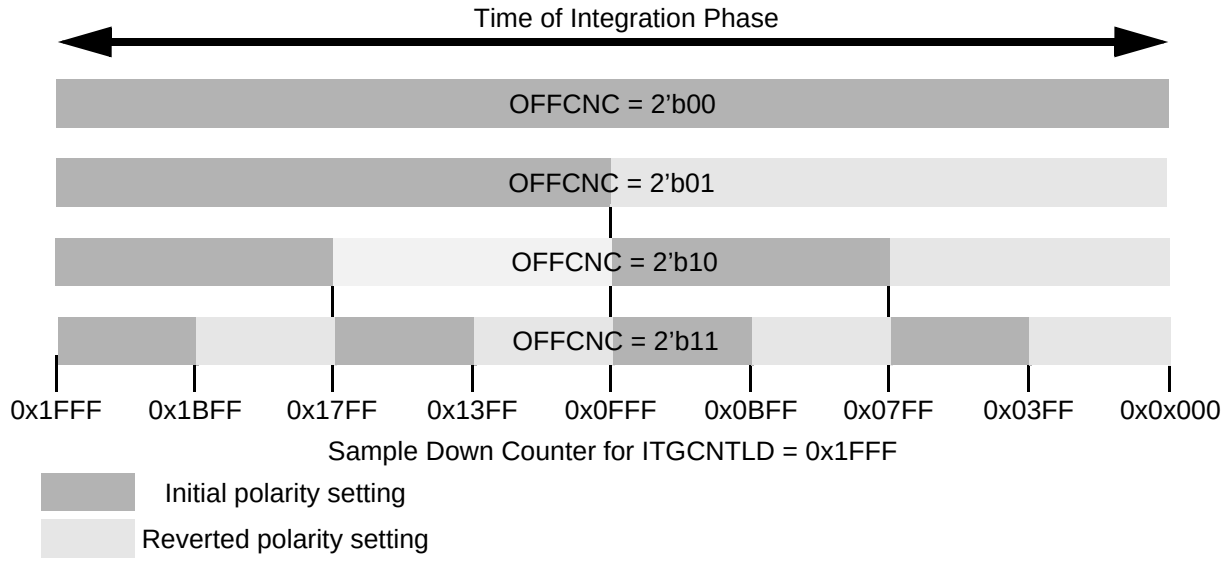


Figure 36-10. Offset Cancellation Polarity Distribution

If the shift process shifts out LS bits from the ITGCNTLD register (non-integer divide) the number shifted out creates an additional polarity flip which lasts the appropriate number of DCNT update periods.

36.4.2 Stepper Stall Detection Measurement

This part of the functional description deals with the main intended use case of the SSD block, this is the detection of the scale end boundaries of the gauge pointer moved by the SM which, in turn, is driven by the SSD block. For details of the related sub blocks refer to Section 36.4.1, Main building blocks of the SSD.

36.4.2.1 Overview of the SSD Measurement

The generic flow of SSD measurement is given in Figure 36-11 below, the numbers denoted at each step belong to the detailed explanations given in Section 36.4.2.2, Details of the SSD Measurement.

The two phases of the BIS are executed in sequence:

1. Blanking phase: Since the non-driven coil used for measurement was driven in the previous step switching transients are induced when changing from the driven into the non-driven state. Therefore both pins of the non-driven coil are connected to one of the analog supply voltages VDDM or VSSM (depending from the RCIR bit) to allow recirculation of these transient currents.
2. Integration phase: This is the actual measurement where the ITGACC register is changed according to the results of the σδ-modulator of the analog block.

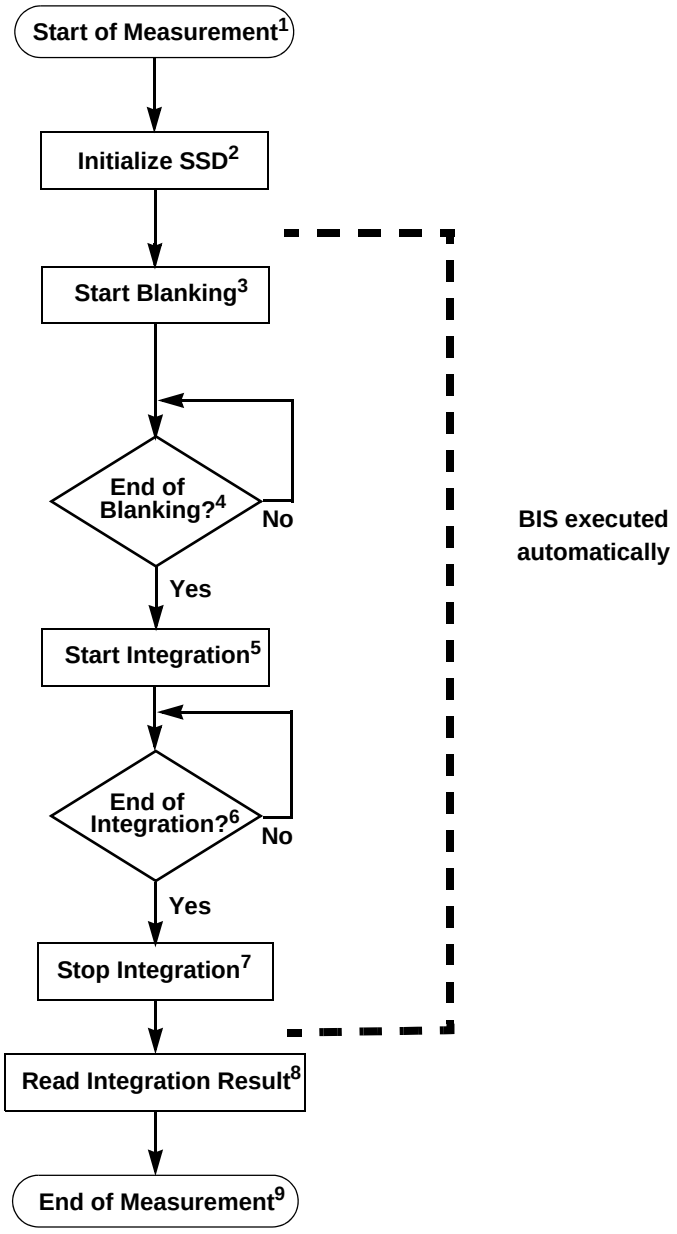


Figure 36-11. Generic SSD Flow

36.4.2.2 Details of the SSD Measurement

This section describes in detail the steps introduced in Figure 36-11. Note that it describes only the measurement process. The decision whether the stall position has been reached or another SM step is required must be made by the controlling CPU depending from the measurement result. All control bits are assumed to have their (inactive) reset values prior to entering the SSD measurement flow. The following paragraphs relate to one complete BIS including the (optional) blanking phase followed by the integration phase:

1. Start of Measurement

For proper usage of the SSD block it must have exclusive control over the coils belonging to the SM whose stall position must be detected. This must be ensured at the device level. Following this the SSD must be enabled by setting the RTZE bit in the CONTROL register, this bit must be left asserted for the complete SSD flow.

It is at this point in time that application-dependent control settings are set which remain constant over the complete SSD flow. These settings consist of the integrate direction of the ITGACC register, which is set by the ITGDIR bit and the direction to advance the STEP setting (increment or decrement influence clockwise or counter-clockwise movement of the SM). Additionally the PRESCALE and the IRQ register must be programmed (it is not recommended to change the content of the PRESCALE register during a running BIS).

2. Initialize SSD

   At this point the STEP bits are set according to the angular position of the SM for the current position. After programming the STEP bits the analog block can be enabled by setting the SDCPU bit.

3. Start Blanking

   This step starts with setting the TRIG bit together with the STEP bits initializing the complete BIS for the next step. Depending from the direction of the rotation the previous step setting is either decremented or incremented, wrapping from 2’b11 to 2’b00 or vice versa. If the BLNDCL bit is set this step marks the start of the SM movement, During blanking both pins of the non-driven coil are connected either to VDDM or VSSM for recirculation, depending from the RCIR bit. The bus clock is divided accordingly to the BLNDIV bits to decrement the DCNT. The BLNST bit is set to allow the user to monitor the status.

4. End of Blanking?

   The end of the blanking phase is automatically detected. If the DCNT reaches 0x0000 and the complete blanking time is expired the BLNIF flag is set and the interrupt triggered according to the BLNIE bit. The BLNST bit is cleared.

5. Start Integration

   After the end of the blanking phase the SSD block continues automatically with the integration phase:

   The ITGCNTLD register is used to initialize the DCNT and is decremented according to the ITGDIV bits setting. The driving coils is powered according to the ITGDCL bit. During the integration phase the polarity flip for offset cancellation is triggered according to the OFFCNC bits. The ITGST bit is set to allow the user to monitor the status.

6. End of Integration?

   The down counter is monitored in the same way like in step 4. The ITGIF flag is set and the interrupt is triggered according to the ITGIE bit. The ITGST bit is cleared.

7. Stop Integration

   On the expiration of the current BIS the integration is stopped, the σδ-modulator is disabled, the ITGACC register is frozen. Note that the current to the coil driven by the SSD block continues according to the ITGDCL and the STEP setting.

8. Read Integration Result

Now the result of the back EMF integration over the time set with the BLNCNTLD register value can be read from the ITGACC register.

9. End of Measurement

   Depending from the result of the measurement the SSD block now can be disabled by clearing the RTZE bit or another measurement can be started. Any additional measurement should start from step 2.

36.4.3 Additional modes of operation

There are several additional modes how the SSD block can operate with the SM coils. They are aside from the main use case.

36.4.3.1 Blanking with No Drive

During blanking of one coil (refer to step 3 in Section 36.4.2, Stepper Stall Detection Measurement) the user can disable the drive of the other coil by not setting the BLNDCL bit. Since the SM has moved in the integration phase of the previous BIS this means that the SM movement is interrupted during blanking. This mode is not useful for SSD.

36.4.3.2 Integration with No Drive

If the ITGDCL bit is switched off the driving coil of the SM is not powered during the integration phase, the SM will not move. Only the σδ-modulator output is integrated with the reference voltage setting belonging to that step. This mode makes no sense for stall detection, but it can be used for certain measurements of the analog sub blocks.

36.5 Initialization Information

36.5.1 Analog Block Startup Time

No specific initialization after a hard reset or after leaving one of the power down modes is necessary, but the user should allow a sufficient startup time of the analog block after hard reset or enabling of the SSD block to compensate switching transients. The startup time specified for the analog block is 20 µs.

36.5.2 Analog Block Polarity Switching Time

If the offset cancellation is used the polarity of the back EMF measurement in the analog block is reverted during the integration phase. Note that the time for the σδ-modulator to achieve a stable output depends strongly from the external circuitry, e.g. coil inductance, parasitic capacitance of the leads.

As an initial estimation for typical applications a time of maximum 10 µs should be taken into account by the user.

36.5.3 SSD startup

The SSD block takes care of smooth transitions between the different steps to avoid current glitches when the coil driven by the block (sine or cosine) changes. Single-cycle glitches in the coil drive current at startup of the SSD flow can be avoided in the following way:

- Program the STEP bits reflecting the current position of the SM prior to the SSD flow with the SDCPU bit still cleared.
- After programming the STEP bits enable the analog block by setting the SDCPU bit.

36.6 Application information

This is additional information intended for use by the customer.

36.6.1 Current flow examples

Figure 36-12 below shows the current flow for a complete sequence of the STEP bits for counter-clockwise movement including the recirculation time:

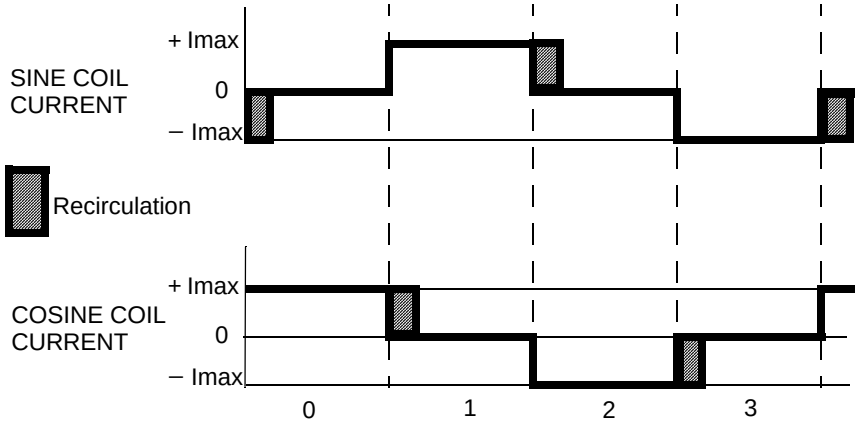


Figure 36-12. Full step sequence for counter-clockwise movement

Different Examples of the current flow in the SM coils for different settings of the control bits are given in Figure 36-13 to Figure 36-16 below:

Figure 36-13 below shows the recirculation at the beginning of STEP = 0. The cosine coil is driven in the direction P → M and the sine coil is recirculated against VDDM.

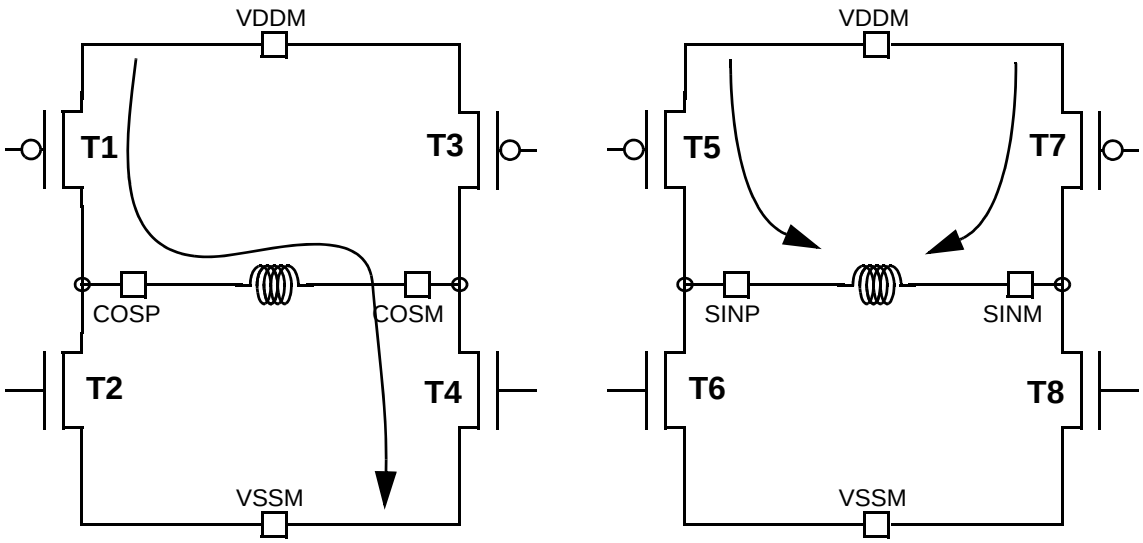


Figure 36-13. Current flow for Blanking (STEP = 0, BLNDCL = 1, RCIR = 0)

In Figure 36-14 below the recirculation at the beginning of the next BIS for STEP = 1 with changed setting of the RCIR bit is shown. The sine coil is driven again in P → M direction and the cosine coil is recirculated against VSSM.

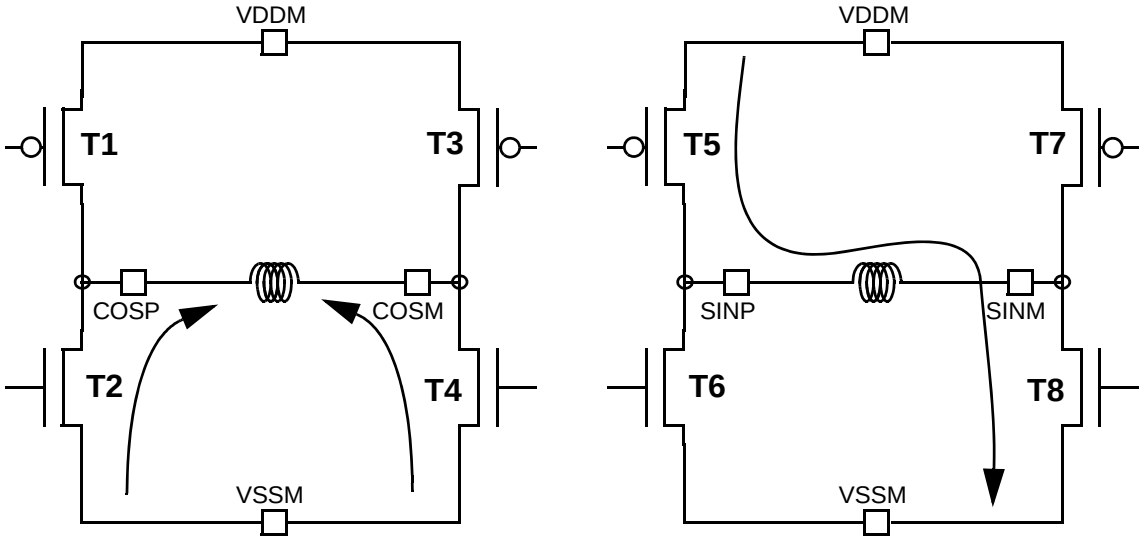


Figure 36-14. Current flow for Blanking (STEP = 1, BLNDCL = 1, RCIR = 1)

In Figure 36-15 below it is shown that for the next step (STEP = 2) the cosine coil is driven in reverse direction with respect to STEP = 0 (M → P direction). Because it is the integration phase of the BIS, the sine coil is isolated from the analog supply voltages. Instead, it is connected to the σδ-modulator (not shown).

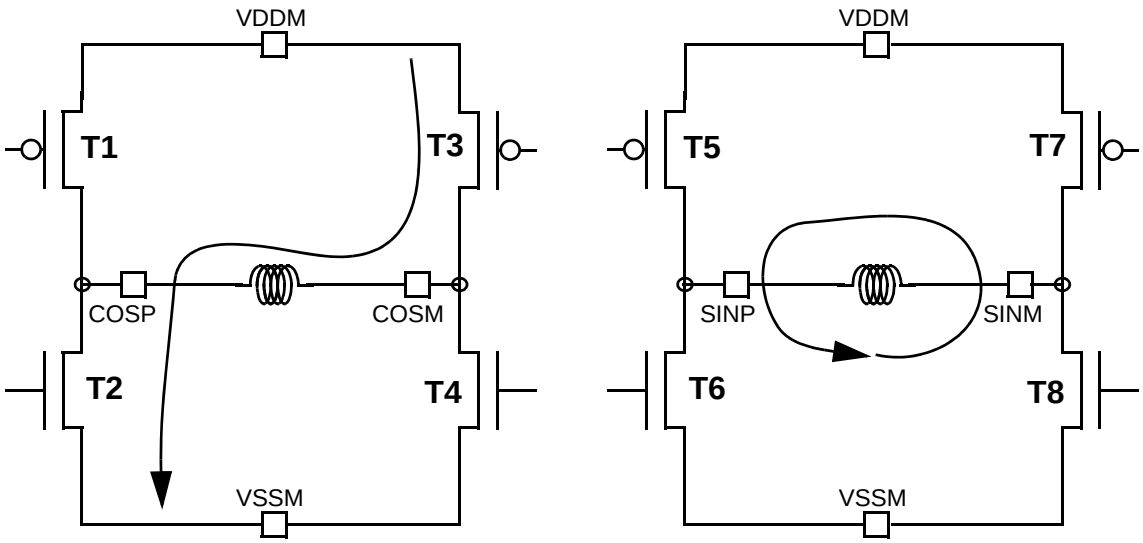


Figure 36-15. Current flow for Integration (STEP = 2, ITGDCL = 1)

Figure 36-16 below shows that the sine coil is driven for STEP = 3 in reverse direction with respect to STEP = 1 (M → P direction). Again the other coil (cosine) is isolated from the analog supply voltages because it is the integration phase of the current BIS.

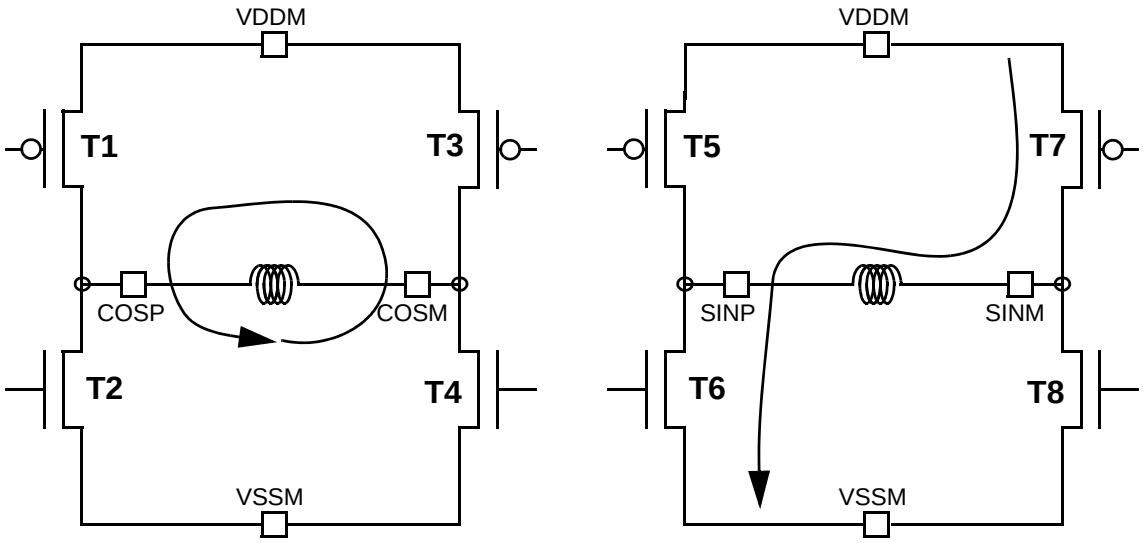


Figure 36-16. Current flow for Integration (STEP = 3, ITGDCL = 1)

36.6.2 Setting of the PRESCALE Register

36.6.2.1 Timing Resolution Considerations

Set the ACDIV bits to the lowest division factor possible, resulting in the highest possible clock frequency for the integration accumulator. This will give the most precise result.

Setting the BLNDIV or ITGDIV bits will influence the resolution of the down counter in the corresponding phase of the BIS (fine resolution required for blanking) as well as the available (absolute) time interval that can be covered by the length of the DCNT register (must be long enough to cover almost one SM step movement for integration). Due to the different prescaler settings for blanking and integration no compromise is necessary between fine resolution for blanking and long time for integration when using high bus frequencies.

It is recommended to select the setting with the best timing resolution for the blanking phase for the BLNDIV bit setting (lowest value). Most likely a different value must be chosen for the ITGDIV bit setting, nevertheless the lowest possible value should be chosen, too.

Note that in normal operation it should not occur that the ACOVIF bit in the IRQ register reads out to be set during the integration. If this happens the bit indicates that either an overflow or an underflow occurred in the ITGACC register. The result should be discarded because the setup of the SSD block was wrong for the current SSD attempt.

36.6.2.2 Offset Cancellation Considerations

Note that the polarity switching for offset cancellation like depicted in Figure 36-10 is controlled by the DCNT register update which is updated depending from the ITGDIV settings. All the divider settings are powers of 2, so the distance in time between two DCNT register updates is always an integer multiple or divider of the ITGACC register update, depending which divider factor is greater than the other one.

If the offset cancellation is used the measurement polarity in the analog block is reverted at least once during the integration phase. As a consequence the σδ-modulator needs some time after each polarity flip to achieve a stable output. An estimation for that time is given in Section 36.5.2, Analog Block Polarity Switching Time.

If the ITGACC register is updated **before** the σδ-modulator output is stable at least one count is incorrect. Since the next polarity flip takes place in the opposite direction this incorrect count will be compensated for by the following polarity flip. Therefore it may be useful to add a small number of DCNT register updates to the integration phase to have an even number of polarity flips in the offset cancellation.

Another mean to improve accuracy is to adjust the DCNT register updates where the polarity switches occur with respect to the following ITGACC register update to allow enough settling time for the σδ-modulator output.

36.6.3 Watching Internal States of the SSD

For some applications it might be required by the application to know the current state of an ongoing BIS. It is recommended to do that by reading the BLNST and the ITGST bits of the CONTROL register. If necessary the DCNT register value may be read to know how far the DCNT register period has expired.

Do **not** poll the DCNT register value for 0x0000 to find out the end of the blanking or integration phase. Aside from generating more CPU load than required this introduces an inaccuracy. Not earlier than the DCNT register divider period belonging to the 0x0000 value has expired the complete blanking or integration phase has been processed.

It should be kept in mind that the recommended way of operation is to enable the appropriate interrupt flag instead of polling the SSD registers.

36.6.4 Stepper Motor Transition Considerations

36.6.4.1 SSD Phase-In and Phase-Out

Prior to starting the SSD flow the SM gauge is usually moved in the proximity of the expected stall position using another SM driver module. To change the driving source of the SM without visible interruption to the SSD block it is essential for the user to replicate the coil drive setting valid at the end of the SM drive into the SSD CONTROL register. Given that the transfer of the port control from the previous SM driver module to the SSD block can be done sufficiently fast the SM will not move but will start at a know position with the SSD flow.

The same applies to the end of the SSD flow when the SM control is transferred to another module. It is essential for the application in this use case that the coil drive setting at the end of the SSD flow is replicated in this other module.

Basically the seamless hand over of the SM coils to another block can be handled by appropriately programming the STEP bits and the ITGDCL bit in the CONTROL register prior to pass pad control to the SSD block. This ensures that one of the SM coils is driven and the SM retains its position when the SSD block gets pad control.

Vice versa the STEP and ITGDCL setting at the end of the last BIS where the gauge stall was detected allow the user to replicate this specific coil drive setting to the module taking over pad control from the SSD block.

For more details refer to specific application notes describing the usage of the SSD block.

36.6.4.2 Changing of SSD Internal States

Depending from the application it may be required to lock the SM into the current angular position to prevent occasional movement prior to the next step. This is especially useful after the integration phase of a BIS. To achieve this the internal logic of the SSD block does the following after the integration phase has expired:

- The coil drive setting of the integration phase is held active: The ITGDCL bit determines the coil drive; the coil and coil direction is determined by the STEP bits.
- The undriven coil is set (back to) recirculation like in the blanking phase.

This leaves both coils in a well-defined state, nevertheless the user should keep this time outside of the BIS as short as possible to avoid visible interruptions of the gauge movement.

When changing to the next step within the SSD flow it should be noted that the update of the STEP bits should coincide (done in the same write to the CONTROL register) like the trigger of the BIS by writing 1 into the TRIG bit. This ensures seamless changeover from one step to the next as well as immediate start of the BIS when the coil driven has changed.

36.6.5 Legacy modes—separate blanking and integration phase

Despite the automatic BIS it is still possible to use the SSD block in a way similar to the old design. Separate blanking and integration phases can be obtained very easily by setting the down counter preload value for the undesired phase to 0x0000. When the corresponding BIS is executed this phase is simply skipped.

Note that in this case the BLNIF bit will be important for the user because its assertion marks the end of the blanking phase.

To ease the programming of separate BLNCNTLD and ITGCNTLD register values on-the-fly the two adjacent registers are placed into one single 32-bit access. Therefore the user who wants to implement programmed-control switching between blanking and integration needs only one single 32-bit register write (to switch the down counter preload values) prior to the execution of the blanking or integration phase.

Chapter 37
System Integration Unit Lite (SIUL)

37.1 Introduction

This chapter describes the System Integration Unit Lite (SIUL), which is used for the management of the pads and their configuration. It controls the multiplexing of the alternate functions used on all pads as well as being responsible for the management of the external interrupts to the device.

37.2 Overview

The SIUL controls the MCU pad configuration, ports, general-purpose input and output (GPIO) signals and external interrupts with trigger event configuration. Figure 37-1 is a block diagram of the SIUL and its interfaces to other system components.

The module provides dedicated general-purpose pads that can be configured as either inputs or outputs. When configured as an output, you can write to an internal register to control the state driven on the associated output pad. When configured as an input, you can detect the state of the associated pad by reading the value from an internal register. When configured as an input and output, the pad value can be read back, which can be used a method of checking if the written value appeared on the pad.

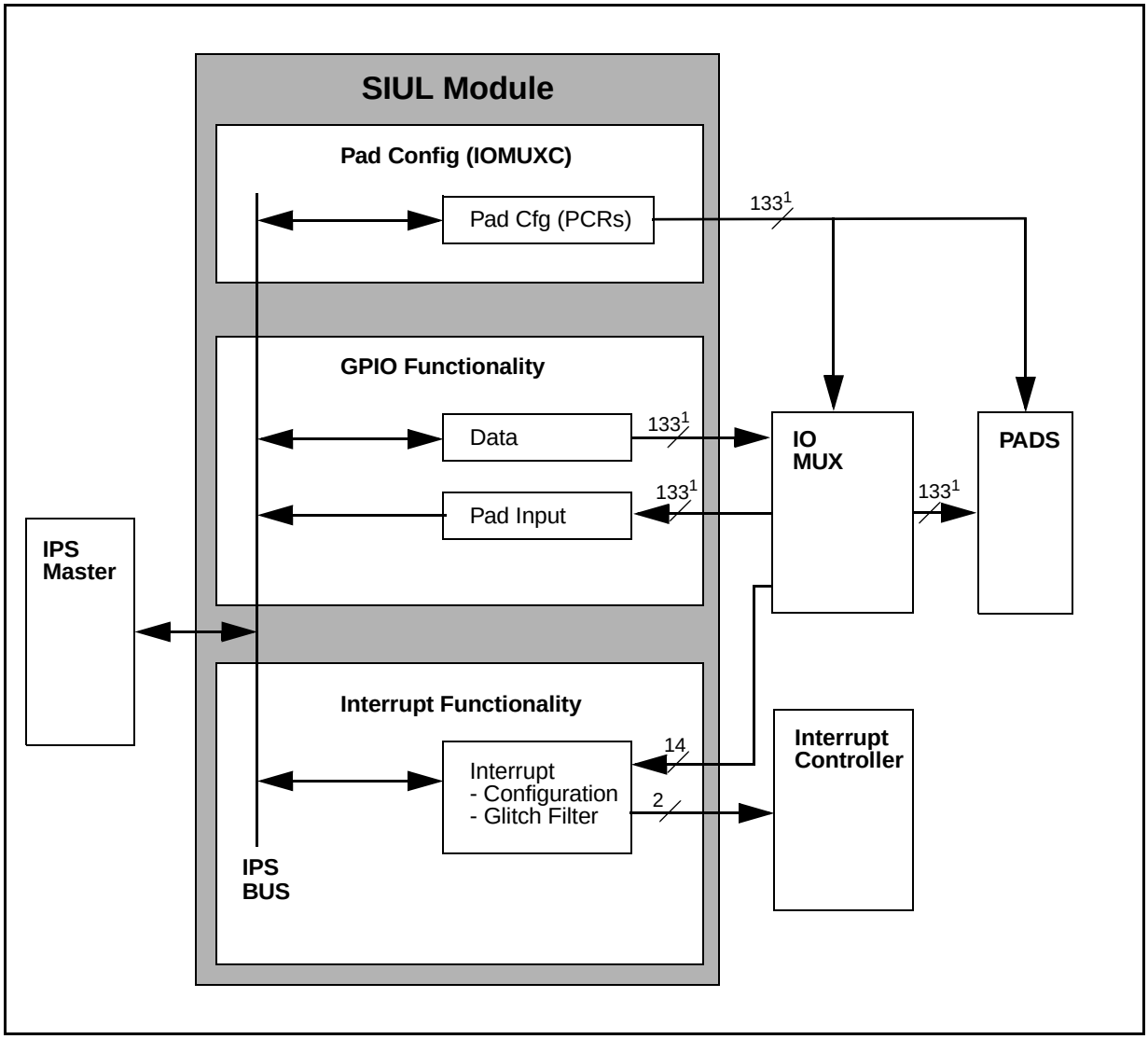


Figure 37-1. SIUL block diagram

[1] Up to 105 I/O pins in the 144-pin packages; up to 133 I/O pins in the 176- and 208-pin packages

37.3 Features

The SIUL supports these distinctive features:

- GPIO
  - GPIO function on up to 133 I/O pins
  - Dedicated input and output registers for each GPIO pin
- External interrupts
  - 2 system interrupt vectors for up to 14 interrupt sources
  - 14 programmable digital glitch filters
  - Independent interrupt mask
  - Edge detection
- System configuration
  - Pad configuration control

37.4 External signal description

The pad configuration allows flexible, centralized control of the pin electrical characteristics of the MCU with the GPIO control providing centralized general purpose I/O for an MCU that multiplexes GPIO with other signals at the I/O pads. These other signals, or alternate functions, will normally be the peripherals functions. The internal multiplexing allows user selection of the input to chip-level signal multiplexors. Each GPIO port communicates via 16 I/O channels. In order to use the pad as a GPIO, the corresponding Pad Configuration Registers (PCR) for all pads used in the port must be configured as GPIO rather than as the alternate pad function.

Table 37-1 lists the external pins used by the SIUL.

Table 37-1. SIUL signal properties

External IRQ	Flag	PCR	Port	Package		
				144	176	208
IRQ_0	EIF[0]	PCR[1]	PA[1]	x	x	x
	EIF[1]	PCR[8]	PA[8]	x	x	x
	EIF[2]	PCR[9]	PA[9]	x	x	x
	EIF[3]	PCR[16]	PB[0]	x	x	x
	EIF[4]	PCR[18]	PB[2]	x	x	x
	EIF[5]	PCR[27]	PB[11]	x	x	x
	EIF[6]	PCR[29]	PB[13]	x	x	x
	EIF[7]	PCR[71]	PF[1]	x	x	x

Table 37-1. SIUL signal properties (continued)

External IRQ	Flag	PCR	Port	Package		
				144	176	208
IRQ_1	EIF[8]	PCR[80]	PF[10]	x	x	x
	EIF[9]	PCR[86]	PG[0]	x	x	x
	EIF[10]	PCR[87]	PG[1]	x	x	x
	EIF[11]	PCR[98]	PG[12]	x	x	x
	EIF[12]	PCR[131]	PK[10]	—	x	x
	EIF[13]	PCR[132]	PK[11]	—	x	x

37.4.1 Detailed signal descriptions

37.4.1.1 General-purpose I/O pins (GPIO[0:132])

The GPIO pins provide general-purpose input and output function. The GPIO pins are generally multiplexed with other I/O pin functions. Each GPIO input and output is separately controlled by an input (GPDI*n_n*) or output (GPDO*n_n*) register.

37.4.1.2 External interrupt request input pins (EIRQ[0:13])[1]

The EIRQ[0:13] are connected to the SIU inputs. Rising or falling edge events are enabled by setting the corresponding bits in the SIU_IREER or the SIU_IFEER register.

37.5 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL module.

37.5.1 SIUL memory map

Table 37-2 gives an overview on the SIUL registers implemented.

Table 37-2. SIUL memory map

Address	Name	Description	Size (bits)	Location
Base (0xC3F9_0000)	—	Reserved	—	—
Base + 0x0004	MIDR1	MCU ID Register #1	32-bit	on page 1190
Base + 0x0008	MIDR2	MCU ID Register #2	32-bit	on page 1191
Base + (0x000C–0x0013)	—	Reserved	—	—

1. EIRQ[0:11] in the 144-pin LQFP; EIRQ[0:13] in all other packages

Table 37-2. SIUL memory map (continued)

Address	Name	Description	Size (bits)	Location
Base + 0x0014	ISR	Interrupt Status Flag Register	32-bit	on page 1193
Base + 0x0018	IRER	Interrupt Request Enable Register	32-bit	on page 1193
Base + (0x001C–0x0027)	—	Reserved	—	—
Base + 0x0028	IREER	Interrupt Rising Edge Event Enable	32-bit	on page 1194
Base + 0x002C	IFEER	Interrupt Falling-Edge Event Enable	32-bit	on page 1194
Base + 0x0030	IFER	IFER Interrupt Filter Enable Register	32-bit	on page 1195
Base + (0x0034–0x003F)	—	Reserved	—	—
Base + 0x0040— Base + 0x0148	PCR0–PCR132[1]	Pad Configuration Registers 0–132	16-bit	on page 1195
Base + (0x014C–0x04FF)	—	Reserved	—	—
Base + 0x0500– Base + 0x0528	PSMI0_3– PSMI40_42	Pad Selection for Multiplexed Inputs	32-bit	on page 1198
Base + (0x052C–0x05FF)	—	Reserved	—	—
Base + 0x0600– Base + 0x0684	GPDO0_3– GPDO132_135[1]	GPIO Pad Data Output Register	32-bit	on page 1202
Base + (0x0688–07FF)	—	Reserved	—	—
Base + 0x0800– Base + 0x0884	GPDI0_3– GPDI132_135[1]	GPIO Pad Data Input Register	32-bit	on page 1202
Base + (0x088C–0x0BFF)	—	Reserved	—	
Base + 0x0C00– Base + 0x0C10	PGPDO0– PGPDO4	Parallel GPIO Pad Data Out Register	32-bit	on page 1203
Base + (0x0C14–0x0C3F)	—	Reserved	—	—
Base + 0x0C40– Base + 0x0C50	PGPDI0–PGPDI4	Parallel GPIO Pad Data In Register	32-bit	on page 1205
Base + (0x0C54–0x0C7F)	—	Reserved	—	—
Base + 0x0C80– Base + 0x0CA4	MPGPDO0– MPGPDO8	Masked Parallel GPIO Pad Data Out Register	32-bit	on page 1206
Base + (0x0CA8–0x0FFF)	—	Reserved	—	—
Base + 0x1000– Base + 0x103C	IFMC0–IFMC15[1]	Interrupt Filter Maximum Counter Register	32-bit	on page 1207
Base + (0x1040–0x107C)	—	Reserved	—	—

Table 37-2. SIUL memory map (continued)

Address	Name	Description	Size (bits)	Location
Base + 0x1080	IFCP	Interrupt Filter Clock Prescaler Register	32-bit	on page 1207
Base + (0x1084–0x3FFF)	—	Reserved	—	—

[1] Not all port pins are available in all packages.

NOTE

A transfer error will be issued when trying to access reserved register space.

37.5.2 Register protection

The individual registers of SIUL are protected from accidental writes. The following registers are protected:

- IRER (*Interrupt Request Enable Register*)
- IREER (*Interrupt Rising-Edge Event Enable Register*)
- IFEER (*Interrupt Falling-Edge Event Enable Register*)
- IFER (*Interrupt Filter Enable Register*), entire PortA, PortB[0:3] and PortC[2:15]
- PSMI[*n*] (*Pad Selection for Multiplexed Inputs*)
- IFMC[*n*] (*Interrupt Filter Maximum Counter*)
- IFPC (*Interrupt Filter Clock Prescaler*)

Refer to Appendix A, Registers Under Protection, for details.

37.5.3 Register description

This section describes in address order all the SIUL registers. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order. The numbering convention of register is MSB=0, however the numbering of internal field is LSB=0, e.g. PARTNUM[5] = MIDR1[10].

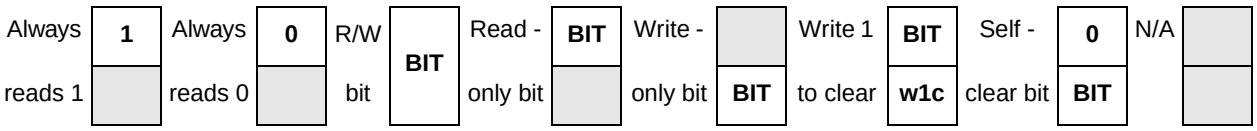


Figure 37-2. Key to Register Fields

37.5.3.1 MCU ID Register #1 (MIDR1)

This register holds identification information about the device.

Address: Base + 0x0004 Access: None

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSP	PKG[4:0]							MAJOR_MASK[3:0]				MINOR_MASK[3:0]			
W																
Reset	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0

Figure 37-3. MCU ID Register #1 (MIDR1)

Table 37-3. MIDR1 field descriptions

Field	Description
PARTNUM [15:0]	MCU Part Number Device part number of the MCU. 0101_0110_0000_0001: 128K 0101_0110_0000_0010: 256K 0101_0110_0000_0011: 320/384K 0101_0110_0000_0100: 512K For the full part number this field needs to be combined with MIDR2.PARTNUM[0:7]
CSP	Always reads back 0
PKG[4:0]	Package Settings Can be read by software to determine the package type that is used for the particular device: 0b01001: 100-pin QFP 0b01101: 144-pin QFP
MAJOR_MASK[3:0]	Major Mask Revision Counter starting at 0x0. Incremented each time when there is a resynthesis.
MINOR_MASK[3:0]	Minor Mask Revision Counter starting at 0x0. Incremented each time a mask change is done.

37.5.3.2 MCU ID Register #2 (MIDR2)

This register holds identification information about the device.

Address: Base + 0x0008 Access: None

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1[3:0]				FLASH_SIZE_2[3:0]										
W																
Reset	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0

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

Figure 37-4. MCU ID Register #2 (MIDR2)

[1] Static bit fixed in hardware

Table 37-4. MIDR2 field descriptions

Field	Description
SF	Manufacturer 0 Freescale Semiconductor 1 Reserved
FLASH_SIZE_1	Coarse granularity for Flash memory size Needs to be added to the memory size indicated by FLASH_SIZE_2 to calculate the actual memory size. 0011 128K 0100 256K 0101 512K
FLASH_SIZE_2	Fine granularity for Flash memory size Needs to be added to the memory size indicated by FLASH_SIZE_1 to calculate the actual memory size. 0000 0 x (FLASH_SIZE_1 / 8) 0010 2 x (FLASH_SIZE_1 / 8) 0100 4 x (FLASH_SIZE_1 / 8)
PARTNUM	ASCII character in MCU Part Number 0x52: Character ‘R’ (this device)
EE	Data Flash present 0 No Data Flash is present 1 Data Flash is present

37.5.3.3 Interrupt Status Flag Register (ISR)

This register holds the external interrupt flags.

Address: Base + 0x0014 Access: User read/write (write 1 to clear)

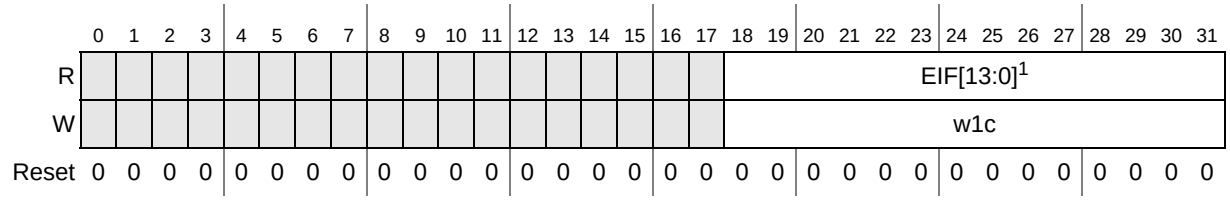


Figure 37-5. Interrupt Status Flag Register (ISR)

1 EIF[11:0] is valid in the 144-pin LQFP.

Table 37-5. ISR field descriptions

Field	Description
EIF[x]	External Interrupt Status Flag x This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by IREER[x] and IFEER[x] has occurred

37.5.3.4 Interrupt Request Enable Register (IRER)

This register is used to enable the external interrupt messaging to the interrupt controller.

Address: Base + 0x0018 Access: User read/write

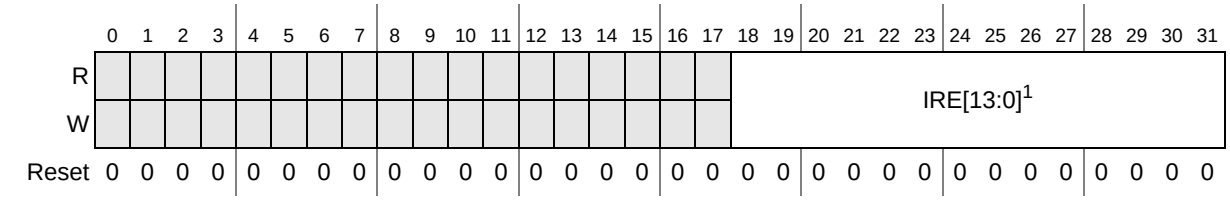


Figure 37-6. Interrupt Request Enable Register (IRER)

1 IRE[11:0] is valid in the 144-pin LQFP.

Table 37-6. IRER field descriptions

Field	Description
IRE[x]	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 A set EIF[x] bit causes an interrupt request

37.5.3.5 Interrupt Rising-Edge Event Enable Register (IREER)

This register is used to enable rising-edge triggered events to be enabled on the corresponding external interrupt pads.

Address: Base + 0x0028 Access: User read/write

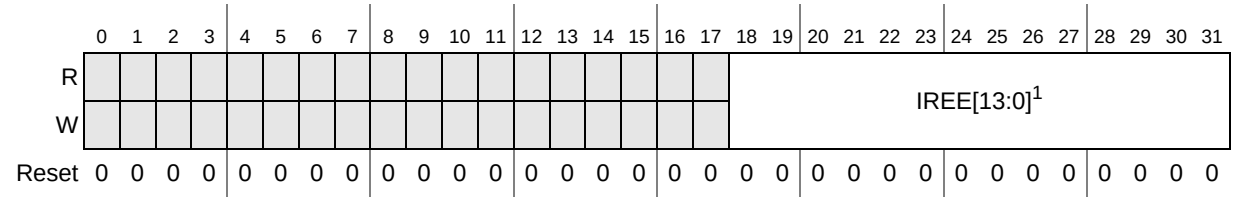


Figure 37-7. Interrupt Rising-Edge Event Enable Register (IREER)

1 IREE[11:0] is valid in the 144-pin LQFP.

Table 37-7. IREER field descriptions

Field	Description
IREE[x]	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

37.5.3.6 Interrupt Falling-Edge Event Enable Register (IFEER)

This register is used to enable falling-edge triggered events to be enabled on the corresponding external interrupt pads.

Address: Base + 0x002C Access: User read/write

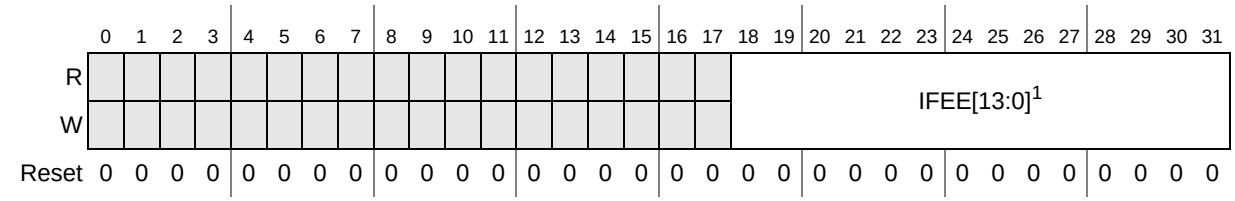


Figure 37-8. Interrupt Falling-Edge Event Enable Register (IFEER)

1 IFEE[11:0] is valid in the 144-pin LQFP.

Table 37-8. IFEER field descriptions

Field	Description
IFEE[x]	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

NOTE

If both the IREE and IFEE bit is cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

37.5.3.7 Interrupt Filter Enable Register (IFER)

This register is used to enable a digital filter counter on the corresponding external interrupt pads to filter out glitches on the inputs.

Address: Base + 0x0030 Access: User read/write

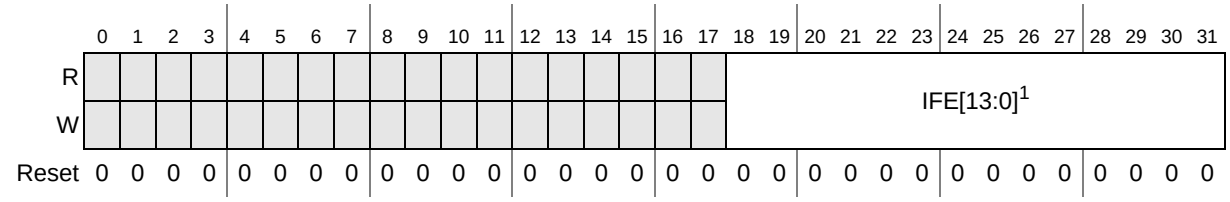


Figure 37-9. Interrupt Filter Enable Register (IFER)

1 IFE[11:0] is valid in the 144-pin LQFP.

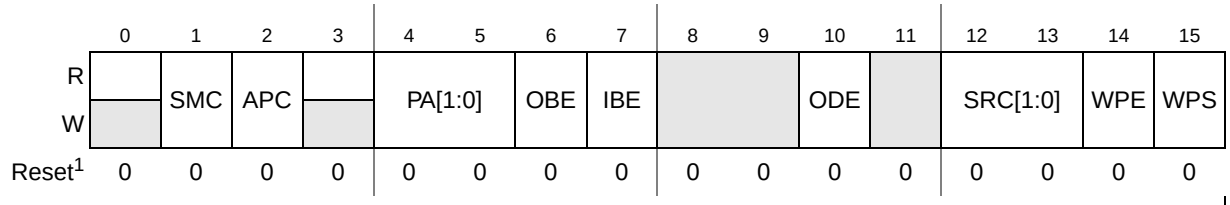
Table 37-9. IFER field descriptions

Field	Description
IFE[x]	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

37.5.3.8 Pad Configuration Registers (PCR0–PCR132)

The Pad Configuration Registers allow configuration of the static electrical and functional characteristics associated with I/O pads. Each PCR controls the characteristics of a single pad.

Address: Base + 0x0040 (PCR0)(133 registers) Access: User read/write
Base + 0x0042 (PCR1)
...
Base + 0x0148 (PCR132)



1 Reset value shown is for the most of the PCRs, however, some PCRs are initialized to different values dependent on the requirements of the device. See Chapter 3, Signal Description, for the reset configurations of each PCR on this device.

Figure 37-10. Pad Configuration Registers (PCRx)

NOTE

16/32-bit access is supported.

In addition to the bit map above, the following Table 37-11 describes the PCR register depending on the pad type. The bits in shaded fields are not implemented for the particular I/O type. The PA field selecting the number of alternate functions may or may not be present depending on the number of alternate functions actually mapped on the pad.

Figure 37-11. Pad Configuration Register (PCR) for the different pad types in MPC5606S

Pad type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pad with GPIO and digital alternate function		SMC	APC		PA[1:0]		OBE	IBE			ODE		SRC[1:0]		WPE	WPS
Pad with slew rate control		SMC	APC		PA[1:0]		OBE	IBE			ODE			SRC[1]	WPE	WPS
Pad with GPIO and analog functionality		SMC	APC		PA[1:0]		OBE	IBE			ODE		SRC[1:0]		WPE	WPS

Table 37-10. PCRx field descriptions

Field	Description
SMC	Safe Mode Control This bit supports the overriding of the automatic deactivation of the output buffer of the associated pad upon entering Safe mode of the SoC. 0 In SoC Safe mode, the output buffer of the pad is disabled. 1 In SoC Safe mode, the output buffer remains functional.
APC	Analog Pad Control This bit enables the usage of the pad as analog input. 0 Analog input path from the pad is gated and cannot be used. 1 Analog input path switch can be enabled by the ADC.
PA[1:0]	Pad Output Assignment This field is used to select the function that is allowed to drive the output of a multiplexed pad. 00 Alternative Mode 0: GPIO. 01 Alternative Mode 1: see Chapter 3, Signal Description 10 Alternative Mode 2: see Chapter 3, Signal Description 11 Alternative Mode 3: see Chapter 3, Signal Description **Note:** Number of bit depending of the number of actual alternate function. Please refer to the *MPC5606S Microcontroller Data Sheet*.
OBE	Output Buffer Enable This bit enables the output buffer of the pad in case the pad is in GPIO mode. 0 Output Buffer of the pad is disabled when PA = 00. 1 Output Buffer of the pad is enabled when PA = 00.
IBE	Input Buffer Enable This bit enables the input buffer of the pad. 0 Input Buffer of the pad is disabled. 1 Input Buffer of the pad is enabled.
ODE	Open Drain Output Enable This bit controls output driver configuration for the pads connected to this signal. Either open drain or push/pull driver configurations can be selected. This feature applies to output pads only. 0 Open drain enable signal is negated for the pad. 1 Open drain enable signal is asserted for the pad.

Table 37-10. PCRx field descriptions (continued)

Field	Description
SRC[1:0]	Slew Rate Control This field controls the slew rate control output signals from the SIUL. The output signals are driven to the value of this field. The actual slew rates are defined by the implementation of the pad devices for a given SoC. **Note:** For low-power modes, keeping these bits asserted may result in more leakage. It is recommended to not drive these bits during low-power modes.
WPE	Weak Pullup/Pulldown Enable This bit controls whether the weak pullup/pulldown devices are enabled/disabled for the pad connected to this signal. **Note:** When a pin is configured as an output, the weak internal pull up/down is disabled regardless of the WPE or WPS settings in the PCR. 0 Weak pull device disabled for the pad. 1 Weak pull device enabled for the pad.
WPS	Weak Pullup/Pulldown Select This bit controls whether weak pullup or weak pulldown devices are used for the pads connected to this signal when weak pullup/pulldown devices are enabled. 0 Weak pulldown selected 1 Weak pullup selected

37.5.3.9 Pad Selection for Multiplexed Inputs Registers (PSMI0_3–PSMI40_42)

Via routing it is possible to define different pads to be possible inputs for a certain peripheral function.

Address: Base + 0x0500–0x0528 (11 registers) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PADSEL0				0	0	0	0	PADSEL1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PADSEL2				0	0	0	0	PADSEL3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-12. Pad Selection for Multiplexed Inputs Register (PSMI0_3)

Table 37-11. PSMI0_3 field descriptions

Field	Description
PADSEL0–3, PADSEL4–7, ... PADSEL28–31	Pad Selection Bits Each PADSEL field selects the pad currently used for a certain input function. See Table 37-12.

In order to multiplex different pads to the same peripheral input, the SIUL provides a register that controls the selection between the different sources.

Table 37-12. Peripheral input pin selection

PSMI register	SIUL address offset	Peripheral input	Mapping of input pin to peripheral input [1]
PSMI[0]	0x500	CAN0_RXD	0: PCR[17] 1: PCR[109]
PSMI[1]	0x501	CAN1_RXD	0: PCR[26] 1: PCR[83] 2: PCR[111]
PSMI[2]	0x502	DCU_PDI[0]	0: PCR[17] 1: PCR[109]
PSMI[3]	0x503	DCU_PDI[1]	0: PCR[16] 1: PCR[110]
PSMI[4]	0x504	DCU_PDI[2]	0: PCR[26] 1: PCR[111]
PSMI[5]	0x505	DCU_PDI[3]	0: PCR[27] 1: PCR[112]

Table 37-12. Peripheral input pin selection (continued)

PSMI register	SIUL address offset	Peripheral input	Mapping of input pin to peripheral input [1]
PSMI[6]	0x506	DCU_PDI[4]	0: PCR[70] 1: PCR[113]
PSMI[7]	0x507	DCU_PDI[5]	0: PCR[71] 1: PCR[114]
PSMI[8]	0x508	DCU_PDI[6]	0: PCR[73] 1: PCR[115]
PSMI[9]	0x509	DCU_PDI[7]	0: PCR[74] 1: PCR[116]
PSMI[10]	0x50A	DCU_PDI[10]	0: PCR[119] 1: PCR[123]
PSMI[11]	0x50B	DCU_PDI[11]	0: PCR[120] 1: PCR[124]
PSMI[12]	0x50C	DCU_PDI[12]	0: PCR[121] 1: PCR[125]
PSMI[13]	0x50D	DCU_PDI[13]	0: PCR[122] 1: PCR[126]
PSMI[14]	0x50E	DSPI1_SS	0: PCR[43] 1: PCR[79]
PSMI[15]	0x50F	EMIOS0_CH8	0: PCR[69] 1: PCR[91] 2: PCR[98]
PSMI[16]	0x510	EMIOS0_CH9	0: PCR[15] 1: PCR[68] 2: PCR[75]
PSMI[17]	0x511	EMIOS0_CH10	0: PCR[14] 1: PCR[67] 2: PCR[74]
PSMI[18]	0x512	EMIOS0_CH11	0: PCR[13] 1: PCR[66] 2: PCR[73]
PSMI[19]	0x513	EMIOS0_CH12	0: PCR[12] 1: PCR[65] 2: PCR[71]
PSMI[20]	0x514	EMIOS0_CH13	0: PCR[11] 1: PCR[64] 2: PCR[70]
PSMI[21]	0x515	EMIOS0_CH14	0: PCR[63] 1: PCR[132]
PSMI[22]	0x516	EMIOS0_CH15	0: PCR[6] 1: PCR[62] 2: PCR[131]

Table 37-12. Peripheral input pin selection (continued)

PSMI register	SIUL address offset	Peripheral input	Mapping of input pin to peripheral input [1]
PSMI[23]	0x517	EMIOS0_CH16	0: PCR[7] 1: PCR[27] 2: PCR[80]
PSMI[24]	0x518	EMIOS0_CH17	0: PCR[5] 1: PCR[122]
PSMI[25]	0x519	EMIOS0_CH18	0: PCR[4] 1: PCR[121]
PSMI[26]	0x51A	EMIOS0_CH19	0: PCR[3] 1: PCR[120]
PSMI[27]	0x51B	EMIOS0_CH20	0: PCR[2] 1: PCR[119]
PSMI[28]	0x51C	EMIOS0_CH21	0: PCR[1] 1: PCR[71] 2: PCR[112]
PSMI[29]	0x51D	EMIOS0_CH22	0: PCR[0] 1: PCR[70] 2: PCR[111]
PSMI[30]	0x51E	EMIOS0_CH23	0: PCR[26] 1: PCR[98]
PSMI[31]	0x51F	EMIOS1_CH16	0: PCR[53] 1: PCR[82] 2: PCR[103]
PSMI[32]	0x520	EMIOS1_CH17	0: PCR[52] 1: PCR[90] 2: PCR[117]
PSMI[33]	0x521	EMIOS1_CH18	0: PCR[9] 1: PCR[29] 2: PCR[51]
PSMI[34]	0x522	EMIOS1_CH19	0: PCR[28] 1: PCR[50] 2: PCR[88]
PSMI[35]	0x523	EMIOS1_CH20	0: PCR[10] 1: PCR[25] 2: PCR[49] 3: PCR[118]
PSMI[36]	0x524	EMIOS1_CH21	0: PCR[24] 1: PCR[48] 2: PCR[89]
PSMI[37]	0x525	EMIOS1_CH22	0: PCR[23] 1: PCR[47]
PSMI[38]	0x526	EMIOS1_CH23	0: PCR[8] 1: PCR[46] 2: PCR[81]

Table 37-12. Peripheral input pin selection (continued)

PSMI register	SIUL address offset	Peripheral input	Mapping of input pin to peripheral input [1]
PSMI[39]	0x527	I2C_SCL_1	0: PCR[79] 1: PCR[132]
PSMI[40]	0x528	I2C_SDA_1	0: PCR[78] 1: PCR[131]
PSMI[41]	0x529	LINFLEX_RXD_1	0: PCR[28] 1: PCR[78]
PSMI[42]	0x52A	EVTI	0: PCR[126] 1: EVTI

1 Connecting a peripheral input to a pad requires assigning both the PSMI value for the peripheral input and the pad assignment in the SIU_PCR register for that signal.

37.5.3.10 GPIO Pad Data Output Registers (GPDO0_3–GPDO132_135)

These registers can be used to set or clear a single GPIO pad with a byte access.

The GPIO pad data output registers are a group of 133 one-byte registers used to set or clear the logic value on their associated pads. Each word contains four registers. The word beginning at Base + 0x0600 contains GPDO0–GPDO3, the word beginning at Base + 0x0604 contains GPDO3–GPDO07, and so on.

Address: Base + 0x0600–0x0684 (34 registers) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PDO [2]	0	0	0	0	0	0	0	PDO [3]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-13. Port GPIO Pad Data Output register 0–3 (GPDO0_3)

Table 37-13. GPDO field descriptions

Field	Description
PDO[x]	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

37.5.3.11 GPIO Pad Data Input Registers (GPDI0_3–GPDI132_135)

These registers can be used to read the GPIO pad data with a byte access.

The GPIO pad data input registers are a group of 133 one-byte registers used to set or clear the logic value on their associated pads. Each word contains four registers. The word beginning at Base + 0x0600 contains GPDI0–GPDI3, the word beginning at Base + 0x0604 contains GPDI3–GPDI07, and so on.

Address: Base + 0x0800–0x0884 (34 registers) Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PDI [2]	0	0	0	0	0	0	0	PDI [3]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-14. Port GPIO Pad Data Input register 0–3 (GPDI0_3)

Table 37-14. GPDI field descriptions

Field	Description
PDI[x]	Pad Data In This bit stores the value of the external GPIO pad associated with this register. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

37.5.3.12 Parallel GPIO Pad Data Out Registers (PGPDO0–PGPDO4)

These registers are used to set or clear the respective pads of the device.

Address: Base + 0x0C00–0x0C10 (5 registers) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PPDO[x+1][15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-15. Parallel GPIO Pad Data Out Register (PGPDO0)

Table 37-15. PGPDO0_4 field descriptions

Field	Description
PPDO[x]	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Output Registers (GPDO0_3–GPDO132_135). The x and bit index define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

NOTE

The PGPDO registers access the same physical resource as the PDO and MPGPDO address locations. Some examples of the mapping:

PPDO[0][0] = PDO[0]

PPDO[2][0] = PDO[32]

37.5.3.13 Parallel GPIO Pad Data In Register (PGPDI0–PGPDI4)

These registers hold the synchronized input value from the pads.

Address: Base + 0x0C40–0x0C50 (5 registers) Access: User read

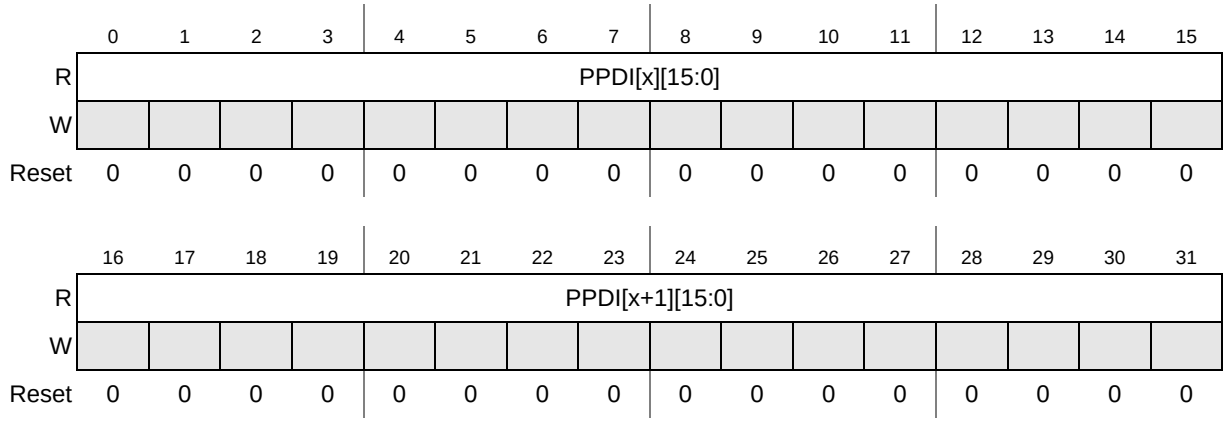


Figure 37-16. Parallel GPIO Pad Data In Register (PGPDI0)

Table 37-16. PGPDI field descriptions

Field	Description
PPDI[x]	Parallel Pad Data In Read the current pad value. Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Input Registers (GPDI0_3–GPDI132_135). The x and bit index define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

37.5.3.14 Masked Parallel GPIO Pad Data Out Register (MPGPDO0–MPGPDO8)

This register can be used to selectively modify the pad values associated to PPDO[x][15:0]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read accesses will return 0.

Address: Base + 0x0C80–0x0CA4 (9 registers) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	MPPDO[x][15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-17. Masked Parallel GPIO Pad Data Out Register (MPGPDO0)

Table 37-17. MPGPDO0_3 field descriptions

Field	Description
MASK[x] [15:0]	Mask Field Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0 The associated bit value in the MPPDO[x] field is ignored 1 The associated bit value in the MPPDO[x] field is written
MPPDO[x] [15:0]	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Output Registers (GPDO0_3–GPDO132_135). The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x*16)+y]

37.5.3.15 Interrupt Filter Maximum Counter Registers (IFMC0–IFMC15)

These registers are used to configure the filter counter associated with each digital glitch filter.

Address: Base + 0x1000–0x103C (16 registers) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	MAXCNTx[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-18. Interrupt Filter Maximum Counter Registers (IFMC0–IFMC15)

Table 37-18. IFMC field descriptions

Field	Description
MAXCNTx [3:0]	Maximum Interrupt Filter Counter setting. Filter Period = T(CK) × MAXCNT*x* + n × T(CK) Where (*n* can be –1 to 3) MAXCNT*x* can be 0 to 15 T(CK): Prescaled Filter Clock Period, which is IRC clock prescaled to IFCP value T(IRC): Basic Filter Clock Period: 62.5 ns (F = 16 MHz)

37.5.3.16 Interrupt Filter Clock Prescaler Register (IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filter counters in the SIUL.

Address: Base + 0x1080 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	IFCP[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-19. Interrupt Filter Clock Prescaler Register (IFCPR)

Table 37-19. IFCPR field descriptions

Field	Description
IFPC [3:0]	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15

37.6 Functional description

37.6.1 General

This section provides a functional description of the System Integration Unit Lite.

37.6.2 Pad control

The SIUL controls the configuration and electrical characteristic of the device pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The pad configuration registers (PCR*n*, see Section 37.5.3.8, Pad Configuration Registers (PCR0–PCR132)) allow software control of the static electrical characteristics of external pins with a single write. These are used to configure the following pad features:

- Open drain output enable
- Slew rate control
- Pull control
- Pad assignment
- Control of analog path switches
- Safe mode behavior configuration

37.6.3 General purpose input and output pads (GPIO)

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

The SIUL manages 133 GPIO pads organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in Figure 37-20, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

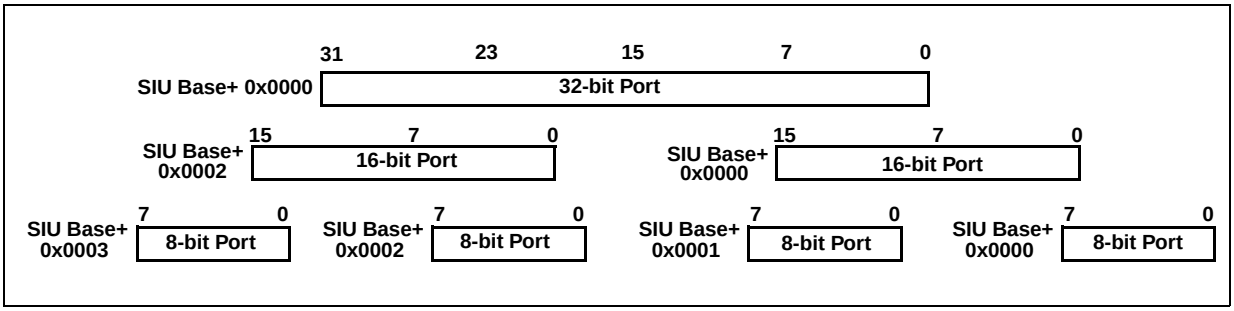


Figure 37-20. Data Port example arrangement showing configuration for different port width accesses

This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL has separate data input (GPDI*n_n*, see Section 37.5.3.11, GPIO Pad Data Input Registers (GPDI0_3–GPDI132_135)) and data output (GPDO*n_n*, see Section 37.5.3.10, GPIO Pad Data Output Registers (GPDO0_3–GPDO132_135)) registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

The allocation of what input function is connected to the pin is defined by the PSMI registers (PCR*n*, see Section 37.5.3.8, Pad Configuration Registers (PCR0–PCR132)).

37.6.4 External interrupts

The SIUL supports 14 external interrupts, EIRQ0–EIRQ13. See Table 37-1 for the map of the external interrupts on the external pins.

The SIUL supports two interrupt vectors to the interrupt controller. Each vector interrupt has up to eight external interrupts combined together with the presence of flag generating an interrupt for that vector if enabled. All of the external interrupt pads within a single group have equal priority.

See Figure 37-21 for an overview of the External Interrupt implementation.

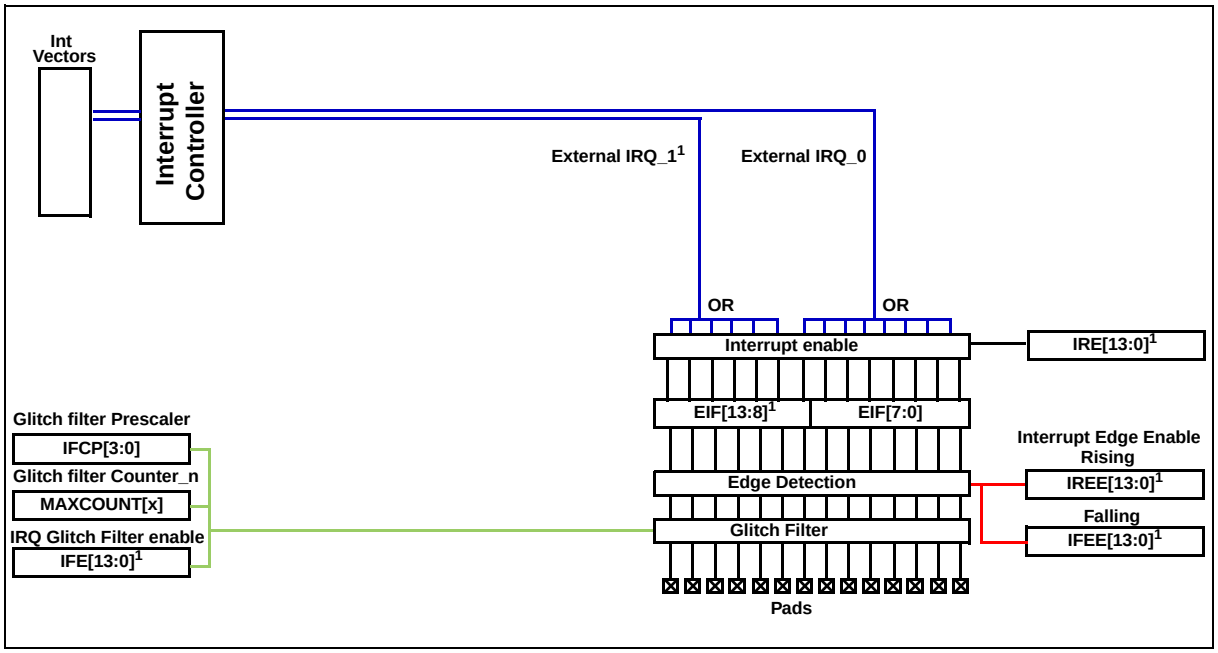


Figure 37-21. External interrupt pad diagram

[1] This value is valid in the 176-pin LQFP and the 208-pin packages

37.6.4.1 External interrupt management

Each interrupt can be enabled or disabled independently. This can be performed using the Interrupt Request Enable Register (see Section 37.5.3.4, Interrupt Request Enable Register (IRER)). A pad defined as an external interrupt can be configured to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured. External interrupts require that the associated input buffer for the pad is enabled (PCR[IBE]=1).

The active EIRQ edge is controlled through the configuration of the registers IREER and IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (see Section 37.5.3.3, Interrupt Status Flag Register (ISR)). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The external interrupt flags map to the INTC vector table as follows:

- EIF[7:0] map to the External IRQ_0 vector
- EIF[13:8] map to the External IRQ_1 vector

37.7 Pin muxing

For pin muxing, please see Chapter 3, Signal Description, of this document.

Chapter 38
System Status and Configuration Module (SSCM)

38.1 Introduction

38.1.1 Overview

The System Status and Configuration Module (SSCM), pictured in Figure 38-1, provides central SOC functionality. On devices with a separate Standby power domain, the System Status block is part of that domain.

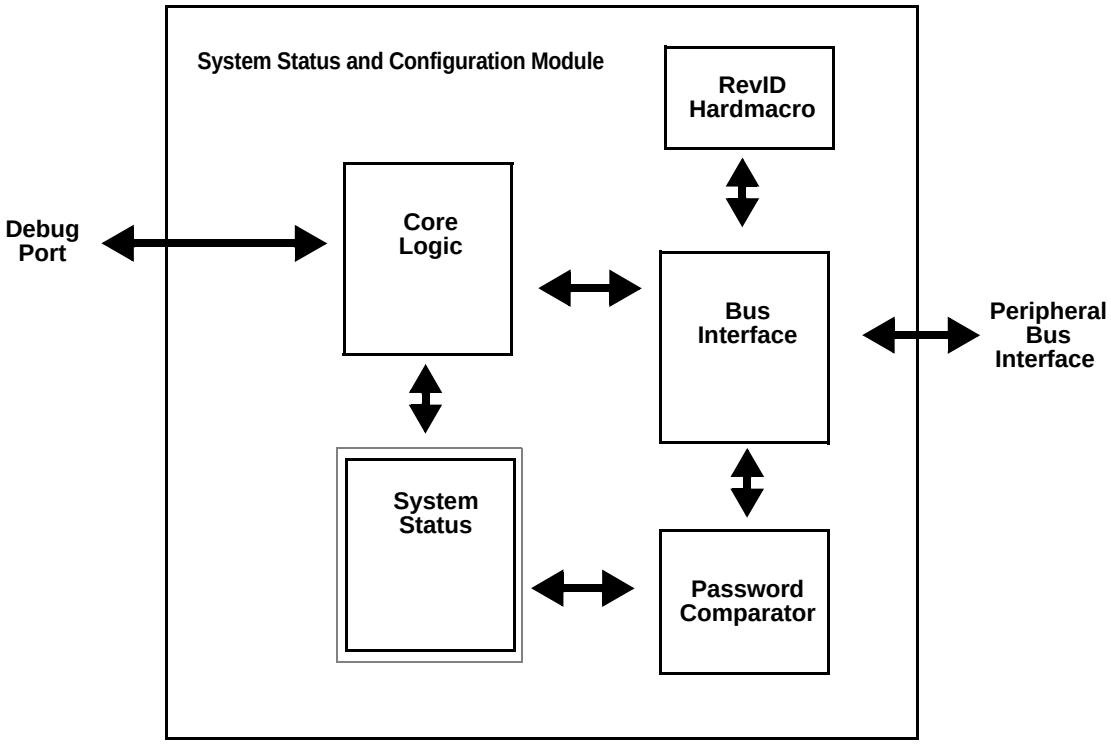


Figure 38-1. System Status and Configuration Module block diagram

38.1.2 Features

The SSCM includes these distinctive features:

- System Configuration and Status
  - Memory sizes/status
  - Device mode and security status
  - Determine boot vector
  - Search Code Flash for bootable sector
- Device identification information (MCU ID Registers)

- Debug Status Port enable and selection
- Bus and peripheral abort enable/disable

38.1.3 Modes of operation

The SSCM operates identically in all system modes.

38.2 Memory map and register description

This section provides a detailed description of all memory-mapped registers in the SSCM.

38.2.1 Memory map

Table 38-1 shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

Table 38-1. module memory map

Address	Register	Size	Access	mode[1]
Base + 0x0000	System Status (STATUS)	16-bit	R	A
Base + 0x0002	System Memory Configuration (MEMCONFIG)	16-bit	R	A
Base + 0x0004	*Reserved*	*16-bit*	*Reads/writes have no effect*	A
Base + 0x0006	Error Configuration (ERROR)	16-bit	R/W	A
Base + 0x0008	Debug Status Port (DEBUGPORT)	16-bit	R/W	A
Base + 0x000A	*Reserved*	*16-bit*	*Reads/writes have no effect*	A
Base + 0x000C	Password Comparison Register High Word	32-bit	R/W	A
Base + 0x0010	Password Comparison Register Low Word	32-bit	R/W	A
Base + 0x0014 to Base + 0x3FFF	*Reserved*	*See Note* [2]		

1 **U** = User mode, **S** = Supervisor mode, **T** = Test mode, **V** = DFV mode, **A** = All (No restrictions)

2 If enabled at the SoC level, accessing these register addresses will cause bus aborts.

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit READ/WRITE to address ‘Base + 0x0002’, but performing a 16-bit access to ‘Base + 0x0003’ is illegal.

38.2.2 Register description

The following memory-mapped registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should be masked out during any read/write operations to avoid conflict with future revisions.

38.2.2.1 System Status Register (STATUS)

The System Status register is a read-only register that reflects the current state of the system.

Address: Base + 0x0000 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	NXEN	PUB	SEC	0	BMODE[2:0]			0	ABD	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0/1	0/1	0/1	0	0	0	0	0

Figure 38-2. Status (STATUS) Register

Table 38-2. STATUS Allowed Register Accesses

	8-bit	16-bit	32-bit[1]
READ	Allowed	Allowed	Allowed
WRITE	Not Allowed	Not Allowed	Not Allowed

1 All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

Table 38-3. STATUS field descriptions

Field	Description
4 NXEN	Nexus enabled.
PUB	Public Serial Access Status. This bit indicates whether serial boot mode with public password is allowed. 0 Serial boot mode with private Flash password is allowed, provided the key hasn't been swallowed 1 Serial boot mode with public password is allowed
SEC	Security Status. This bit reflects the current security state of the Flash. 0 The Flash is not secured 1 The Flash is secured
8-10 BMODE [0:2]	Device Boot Mode. 000 Reserved for FlexRay Boot Serial Boot Loader 001 legacy bootstrap via CAN 010 legacy bootstrap via UART 011 Single Chip 100–111 Reserved This field is only updated during reset.
12 ABD	Autobaud. Indicates that autobaud detection is active when in SCI or CAN serial boot loader mode. No meaning in other modes.

38.2.2.2 System Memory Configuration Register

The System Memory Configuration register is a read-only register that reflects the memory configuration of the system.

Address: Base + 0x0002 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R											IVLD					DVLD
W																
RESET:	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	1

Figure 38-3. System Memory Configuration (MEMCONFIG) Register

Table 38-4. MEMCONFIG field descriptions

Field	Description
IVLD	Code Flash Valid. This bit identifies whether or not the on-chip Code Flash is accessible in the system memory map. The Flash may not be accessible due to security limitations, or because there is no Flash in the system. 0 Code Flash is not accessible 1 Code Flash is accessible
DVLD	Data Flash Valid. This bit identifies whether or not the on-chip Data Flash is visible in the system memory map. The Flash may not be accessible due to security limitations, or because there is no Flash in the system. 0 Data Flash is not visible 1 Data Flash is visible

Table 38-5. MEMCONFIG Allowed Register Accesses

	8-bit	16-bit	32-bit
READ	Allowed	Allowed	Allowed (also reads STATUS register)
WRITE	Not Allowed	Not Allowed	Not Allowed

38.2.2.3 Error Configuration

The Error Configuration register is a read-write register that controls the error handling of the system.

Address: Base + 0x0006 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PAE	RAE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-4. Error Configuration (ERROR) Register

Table 38-6. ERROR field descriptions

Field	Description
14 PAE	Peripheral Bus Abort Enable. This bit enables bus aborts on any access to a peripheral slot that is not used on the device. This feature is intended to aid in debugging when developing application code. 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception
15 RAE	Register Bus Abort Enable. This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code. 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception Transfers to Peripheral Bus resources may be aborted even before they reach the Peripheral Bus (i.e. at the PBRIDGE level). In this case, the **PER_ABORT** and **REG_ABORT** register bits will have no effect on the abort. 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception

Table 38-7. ERROR Allowed Register Accesses

	8-bit	16-bit	32-bit
READ	Allowed	Allowed	Allowed
WRITE	Allowed	Allowed	Not Allowed

38.2.2.4 Debug Status Port Register

The Debug Status Port register is used to (optionally) provide debug data on a set of pins.

Address: Base + 0x0008 Access: Read/Write

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

Figure 38-5. Debug Status Port (DEBUGPORT) Register

Table 38-8. DEBUGPORT field descriptions

Field	Description
13-15 DEBUG _MODE [0:2]	Debug Status Port Mode. This field selects the alternate debug functionality for the Debug Status Port 000 No alternate functionality selected 001 Mode 1 Selected 010 Mode 2 Selected 011 Mode 3 Selected 100 Mode 4 Selected 101 Mode 5 Selected 110 Mode 6 Selected 111 Mode 7 Selected Table 38-9 describes the functionality of the Debug Status Port in each mode.

Table 38-9. Debug status port modes

Pin[1]	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
0	STATUS[0]	STATUS[8]	MEMCONFIG[0]	MEMCONFIG[8]	Reserved	Reserved	Reserved
1	STATUS[1]	STATUS[9]	MEMCONFIG[1]	MEMCONFIG[9]	Reserved	Reserved	Reserved
2	STATUS[2]	STATUS[10]	MEMCONFIG[2]	MEMCONFIG[10]	Reserved	Reserved	Reserved
3	STATUS[3]	STATUS[11]	MEMCONFIG[3]	MEMCONFIG[11]	Reserved	Reserved	Reserved
4	STATUS[4]	STATUS[12]	MEMCONFIG[4]	MEMCONFIG[12]	Reserved	Reserved	Reserved
5	STATUS[5]	STATUS[13]	MEMCONFIG[5]	MEMCONFIG[13]	Reserved	Reserved	Reserved
6	STATUS[6]	STATUS[14]	MEMCONFIG[6]	MEMCONFIG[14]	Reserved	Reserved	Reserved
7	STATUS[7]	STATUS[15]	MEMCONFIG[7]	MEMCONFIG[15]	Reserved	Reserved	Reserved

1 All signals are active high, unless otherwise noted

Table 38-10. DEBUGPORT allowed register accesses

	8-bit	16-bit	32-bit[1]
READ	Allowed	Allowed	Not Allowed
WRITE	Allowed	Allowed	Not Allowed

1 All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

38.2.2.5 Password Comparison Registers

These registers allow to unsecure the device, if the correct password is known.

Address: 0x000C Access: Read/Write

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

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

Figure 38-6. Password Comparison Register High Word (PWCMPH)

Address: 0x0010 Access: Read/Write

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

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

Figure 38-7. Password Comparison Register Low Word (PWCMPL)

Table 38-11. Password Comparison Register field descriptions

Field	Description
0-31 PWD_HI [0:31]	Upper 32 bits of the password
0-31 PWD_LO [0:31]	Lower 32 bits of the password

Table 38-12. PWCMPH/L Allowed Register Accesses

	8-bit	16-bit	32-bit[1]
READ	Allowed	Allowed	Allowed
WRITE	Not Allowed	Not Allowed	Allowed

1 All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

In order to unsecure the device, the password needs to be written as follows: first the upper word to the PWCMPH register, then the lower word to the PWCMPL register. The SSCM compares the password and if the password is correct, unlocks the device.

38.3 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

38.4 Initialization/application information

38.4.1 Reset

The reset state of each individual bit is shown within the Section 38.2.2, Register description.

Chapter 39
System Timer Module (STM)

39.1 Introduction

39.1.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

39.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

39.1.3 Modes of operation

The STM supports two device modes of operation: normal and debug.

When the STM is enabled in normal mode, its counter runs continuously. While debugging, operation of the counter is controlled by the STM_CR[FRZ] bit. If the FRZ bit is set, the counter is stopped when the MCU is stopped by a debugger, otherwise it continues to run.

39.2 External signal description

The STM does not have any external interface signals.

39.3 Memory map and register definition

The STM programming model has fourteen 32-bit registers. The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

39.3.1 Memory map

The STM memory map is shown in Table 39-1.

Table 39-1. STM memory map

Address Offset	Register Name	Register description	Size (bits)	Access	Location
0x0000	STM_CR	STM Control Register	32	R/W	on page 1222
0x0004	STM_CNT	STM Counter Value	32	R/W	on page 1223
0x0008	Reserved				
0x000C	Reserved				
0x0010	STM_CCR0	STM Channel 0 Control Register	32	R/W	on page 1224
0x0014	STM_CIR0	STM Channel 0 Interrupt Register	32	R/W	on page 1224
0x0018	STM_CMP0	STM Channel 0 Compare Register	32	R/W	on page 1225
0x001C	Reserved				
0x0020	STM_CCR1	STM Channel 1 Control Register	32	R/W	on page 1224
0x0024	STM_CIR1	STM Channel 1 Interrupt Register	32	R/W	on page 1224
0x0028	STM_CMP1	STM Channel 1 Compare Register	32	R/W	on page 1225
0x002C	Reserved				
0x0030	STM_CCR2	STM Channel 2 Control Register	32	R/W	on page 1224
0x0034	STM_CIR2	STM Channel 2 Interrupt Register	32	R/W	on page 1224
0x0038	STM_CMP2	STM Channel 2 Compare Register	32	R/W	on page 1225
0x003C	Reserved				
0x0040	STM_CCR3	STM Channel 3 Control Register	32	R/W	on page 1224
0x0044	STM_CIR3	STM Channel 3 Interrupt Register	32	R/W	on page 1224
0x0048	STM_CMP3	STM Channel 3 Compare Register	32	R/W	on page 1225
0x004C - 0x3FFF	Reserved				

39.3.2 Register descriptions

The following sections detail the individual registers within the STM programming model.

39.3.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Offset 0x000 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS								0	0	0	0	0	0	FRZ	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-1. STM Control Register (STM_CR)

Table 39-2. STM_CR field descriptions

Field	Description
CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1–256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ... 0xFF Divide system clock by 256
FRZ	Freeze. Allows the timer counter to be stopped when the MCU is stopped by a debugger. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
TEN	Timer Counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

39.3.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Offset 0x004 Access: Read/Write

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

Figure 39-2. STM Count Register (STM_CNT)

Table 39-3. STM_CNT field descriptions

Field	Description
CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

39.3.2.3 STM Channel Control Register (STM_CCRn)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Offset 0x10+0x10*n — Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-3. STM Channel Control Register (STM_CCRn)

Table 39-4. STM_CCRn field descriptions

Field	Description
CEN	Channel Enable. 0 The channel is disabled. 1 The channel is enabled.

39.3.2.4 STM Channel Interrupt Register (STM_CIRn)

The STM Channel Interrupt Register (STM_CIRn) has the interrupt flag for channel n of the timer.

Offset 0x14+0x10*n — Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-4. STM Channel Interrupt Register (STM_CIRn)

Table 39-5. STM_CIRn field descriptions

Field	Description
CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

39.3.2.5 STM Channel Compare Register (STM_CMPn)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Offset 0x18+0x10*n Access: Read/Write

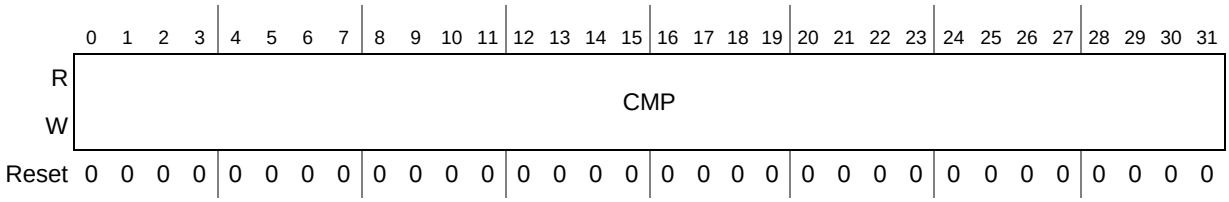


Figure 39-5. STM Channel Compare Register (STM_CMPn)

Table 39-6. STM_CMPn Register field descriptions

Field	Description
CMP	Compare value for channel n. If the STM_CCRn[CEN] bit is set and the STM_CMPn register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIRn[CIF] bit is set.

39.4 Functional description

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode the counter continuously increments. When the MCU is stopped by a debugger, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode, otherwise it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCRn), a channel interrupt register (STM_CIRn) and a channel compare register (STM_CMPn). The channel is enabled by setting the STM_CCRn[CEN] bit. When enabled, the channel will set the STM_CIR[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the STM_CIRn[CIF] bit. A write of 0 to the STM_CIRn[CIF] bit has no effect.

NOTE

The STM counter does not advance when the system clock is stopped.

Chapter 40
Voltage Regulators and Power Supplies

40.1 Introduction

This device includes three on-chip voltage regulators for power management and distribution, allowing low-power operation and optimization of power consumption.

On this device, the general purpose inputs and outputs (GPIO) are arranged in several banks with separate external GPIO power supply pins, allowing groups of GPIO to operate at different supply voltages.

40.2 Voltage regulators

The internal voltage regulators are used to provide a 1.2 V digital supply to the internal logic of the device. The main/input supply is 3.3 V to 5.0 V ± 10% and the digital/regulated output supply is 1.2 V ± 10%. The voltage regulator used in this device comprises three regulators.

- High power or main regulator (HPREG) requiring an external NPN ballast transistor.
- Low-power regulator (LPREG)
- Ultra low-power regulator (ULPREG)

The HPREG and LPREG regulators are switched off in Standby mode to save power consumption by the regulator itself. During Stop and Halt modes only, the HPREG regulator may be switched off. The ULPREG regulator is always kept ON.

The internal voltage regulators require an external capacitance to be connected to the 1.2 V supply pins in order to provide a stable low voltage digital supply to the device. Capacitances should be placed on the board as near as possible to the associated pins.

The regulator has two digital domains, one for the main regulator (HPREG) and the low-power regulator (LPREG) called the “High Power domain”, and one for the ultra low-power regulator (ULPREG) called the “Standby domain”. For each domain there is a low voltage detector (LVD) for the 1.2 V output voltage (ULVDD and MLVDD). Additionally, there are two low voltage detectors for the main/input supply with different thresholds, one at 3.3 V (ULVDM), the other at 5 V (ULVDM5).

40.2.1 Block diagram

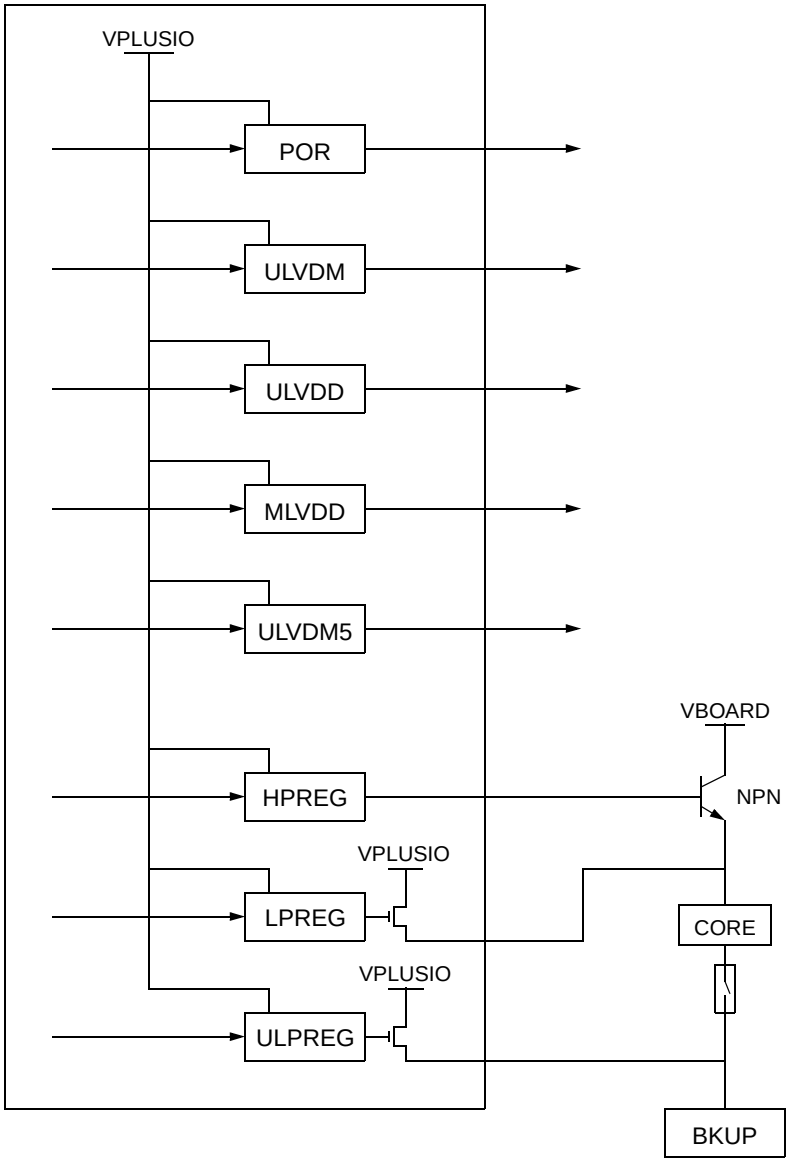


Figure 40-1. Voltage regulator diagram

40.2.2 External signals

Table 40-1 provides an overview of the voltage regulator external signals.

Table 40-1. Voltage regulator external signals

Name	Type	Voltage	Description
VDDR	Supply	3.0 V–5.5 V	Power Supply for the voltage regulators
VSSR	Ground	—	Ground supply for digital core and voltage regulators
VRC	Output	—	Regulator drive for external NPN transistor base

40.2.3 Detailed signal descriptions

40.2.3.1 VDDR

VDDR is the 3.3 V to 5 V supply for the voltage regulators and LVD blocks. See the *MPC5606S Microcontroller Data Sheet* for details of recommended decoupling capacitance on this pin.

40.2.3.2 VRC

VRC is the 1.2 V regulator output that drives the base of the external NPN ballast transistor.

40.3 Memory map and register definition

Table 40-2. Voltage regulator memory map

Address	Register	Access	Reset Value
VREG_BASE + 0x0000	VREG_CTL—Voltage Regulator Control Register	R/W	0x00000001

40.3.1 Voltage Regulator Control Register (VREG_CTL)

Address Offset: 00h — Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5V_LVD_MASK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 40-2. Voltage Regulator Control Register (VREG_CTL)

Table 40-3. VREG_CTL field descriptions

Field	Description
0-30	Reserved.
31	5 V LVD MASK: Mask bit for 5 V LVD from regulator This is a read/write bit and must be unmasked by writing a 1 by software to generate LVD functional reset request to MC_RGM for 5 V trip. 0 5 V LVD is not masked. 1 5 V LVD is masked

40.4 Functional description

40.4.1 High Power or Main Regulator (HPREG)

The HPREG converts the 3.3 V to 5 V input supply voltage to the 1.2 V digital supply. The nominal target output is 1.2 V. The actual output will be in range of 1.08–1.32 V in the full current load range 0–400 mA after trimming.

The HPREG regulator requires an external NPN ballast transistor. Recommended transistors are BCP68 from ON Semiconductor or BCP56 from STMicroelectronics.

Stabilization of the HPREG is achieved using an external capacitance.

The regulator can be switched off by software.

40.4.2 Low-power Regulator (LPREG)

The LPREG generates power for the chip in Stop mode, providing the output supply of 1.2 V. The control part of the regulator can be used to disable the low-power regulator. This action is managed by MC_ME.

40.4.3 Ultra Low-power Regulator (ULPREG)

The ULPREG generates power for the standby domain as well as a part of the main domain. The control circuit of ULPREG can be used to disable the ultra low-power regulator by SW. This action is managed by MC_ME.

40.4.4 Low Voltage Detectors (LVD) and Power On Reset (POR)

Four LVDs are available (see Figure 40-1).

- ULVDM for the 3.3 V to 5 V input supply with threshold at the 3.3 V level
- ULVDM5 for the 3.3 V to 5 V input supply with threshold at the 5 V level
- Two LVD_DIGs, ULVDD and MLVDD, for the 1.2 V output voltage

ULVDM and ULVDM5 sense the 3.3 V to 5 V power supply for CORE, shared with the GPIO ring supply, and indicate when the 3.3 V to 5 V supply is stable. The threshold levels of ULVDM5 are trimmable with the help of LVDM5[0:3] trim bits.

Two LVD_DIGs are provided in the design. One LVD_DIG is placed in the high power domain and senses the HPREG/LPREG output, indicating that the 1.2 V output is stable. The other LVD_DIG is placed in the standby domain and senses the standby 1.2 V supply level, indicating that the 1.2 V output is stable. The reference voltage used for all LVDs is generated by the low-power reference generator and is trimmed for LVD_DIG, using the bits LP[4:7]. Therefore, during the pre-trimming period, LVD_DIG exhibits higher thresholds whereas, after trimming, the thresholds come within the desired range. Power-down pins are provided for LVDs. When LVDs are powered down, their outputs are pulled high.

POR is required to initialize the chip during the voltage supply rise time. POR works only on the rising edge of the main supply voltage. To ensure that it functions during the following rising edge of the supply voltage, it is reset by the output of the ULVDM block when the main supply goes below the lower voltage threshold of ULVDM.

POR is asserted on startup when the Vdd supply is above the minimum value of V_{PORUP} (refer to the *MPC5606S Microcontroller Data Sheet* for this value). It will be released only after the Vdd supply goes above V_{PORH} (refer to the *MPC5606S Microcontroller Data Sheet* for this value). Vdd above V_{PORH} ensures that the power management module, including the internal LVD modules, are fully functional.

40.4.5 VREG digital interface

The voltage regulator digital interface provides the temporization delay at initial startup and at exit from low-power modes. A signal indicating that the ultra low-power domain is powered is used at startup to release reset to the temporization counter. At exit from low-power modes, the power-down for high power regulator request signal is monitored by the digital interface and used to release reset to the temporization counter. In both cases, on completion of the delay counter, an end-of-count signal is released; this is gated with an other signal indicating that the main domain voltage is fine, in order to release the VREGOK signal. This is used by MC_RGM to release the reset to the device. It manages other specific requirements, including the transition between high power or low-power mode to ultra low-power mode, avoiding a voltage drop below the permissible threshold limit of 1.08V.

The VREG digital interface also contains a control register to mask the 5 V LVD status from the voltage regulator at startup.

40.5 GPIO power supply configuration

The GPIO pins on this device are organized in five separate banks. Each bank has associated VDD/VSS power supply pairs. The five banks of GPIO can be powered independently, allowing these banks to be run at different voltages in the 3.0 V to 5.5 V range.

Table 40-4 describes the GPIO banks, their main functions, supply pins and associated port pins. For full details of GPIO functionality, refer to Chapter 3, Signal Description.

Table 40-4. GPIO power bank supplies and functionality

GPIO bank	Available functions[1]	Supply pins	Port pins
Stepper Motor Bank	Stepper Motors [1:0], eMIOS Stepper Motors [3:2], eMIOS Stepper Motors [5:4], eMIOS	VDDMA[2], VSSMA VDDMB[2], VSSMB VDDMC[2], VSSMC	PortD[7:0] PortD[15:8] PortE[7:0]
Analog Bank 0	ADC, 32 kHz SXOSC	VDDE_C, VSSE_C	PortC[15:0]
Digital Bank 0	DCU, QuadSPI, LCD, eMIOS, I2C, CAN, LIN	VDDE_A, VSSE_A	PortA[15:0] PortF[15:3, 1:0] PortG[12:0] PortH[5]
Digital Bank 1	CAN, LIN, I2C, SPI, eMIOS, PDI, ADC-MUX, Nexus[3]	VDDE_B, VSSE_B	PortB[13:12, 9:4] PortF[2] PortH[4:0] PortJ[7:4] PortK[11:2] RESET
Digital Bank 2	CAN, LIN, eMIOS, PDI	VDDE_E, VSSE_E	PortB[11:10, 3:0] PortJ[15:8, 3:0] PortK[1:0]

1 Not all functions are available simultaneously. Refer to Chapter 3, Signal Description.

2 If VDDMA, VDDMB, and VDDMC are all powered, the must all be connected to the same supply level.

3 Nexus available on dedicated pins in Digital Bank 1 for 208MAPBGA package option and as a multiplexed option on 176LQFP package option

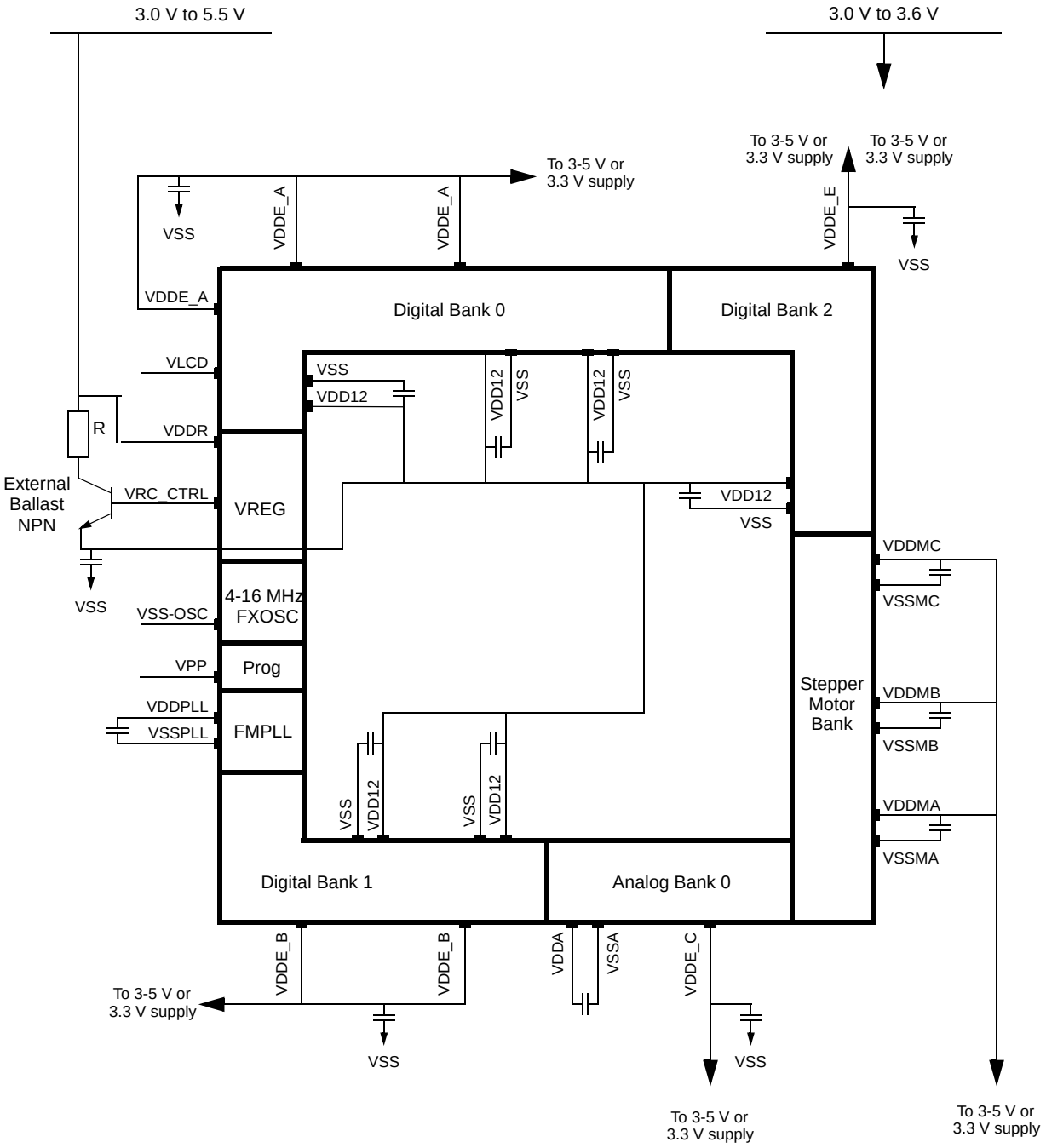


Figure 40-3. Power supply configuration

40.6 Power domain organization

To satisfy stringent requirements for current consumption in the different operational modes, this device is partitioned into different power domains. Organization into these power domains primarily means having separate power supplies that are separated from each other by the use of power switches. These

different separated power supplies are hence able to switch off power to certain regions of the device, to avoid even leakage current consumption in logic supplied by the corresponding power supply.

This device employs three primary power domains: power domains PD0 and PD1, and a RAM power domain RD1.

Power domain organization and connections to the internal regulator are depicted in Figure 40-4.

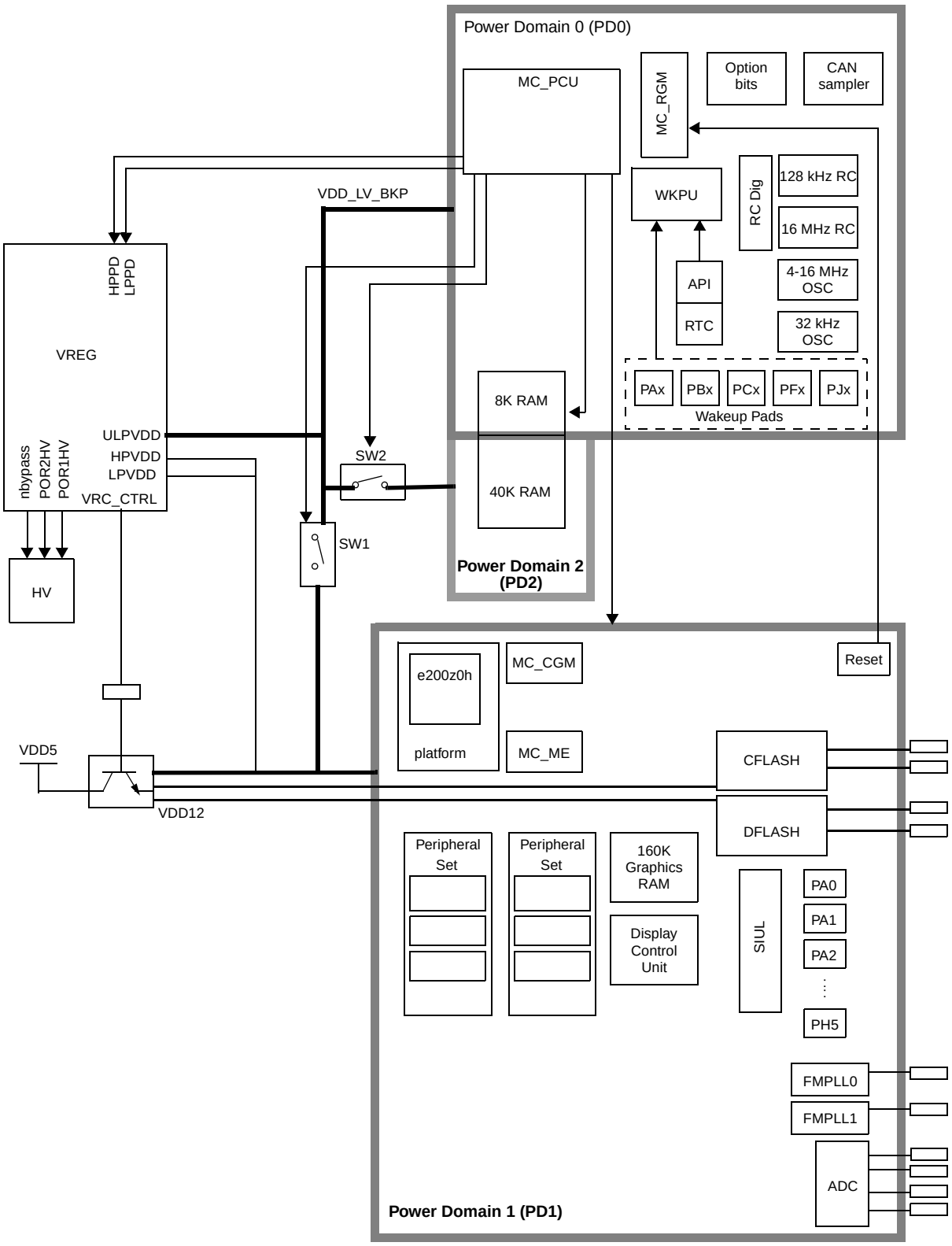


Figure 40-4. Power domain organization

Chapter 41
Wakeup Unit (WKPU)

41.1 Overview

The Wakeup Unit supports up to 19[1] external sources that can generate interrupts or wakeup events, of which one can cause non-maskable interrupt requests. Figure 41-1 is a block diagram of the Wakeup Unit and its interfaces to other system components.

The wakeup lines are mapped to the interrupt vectors as shown in Table 41-1.

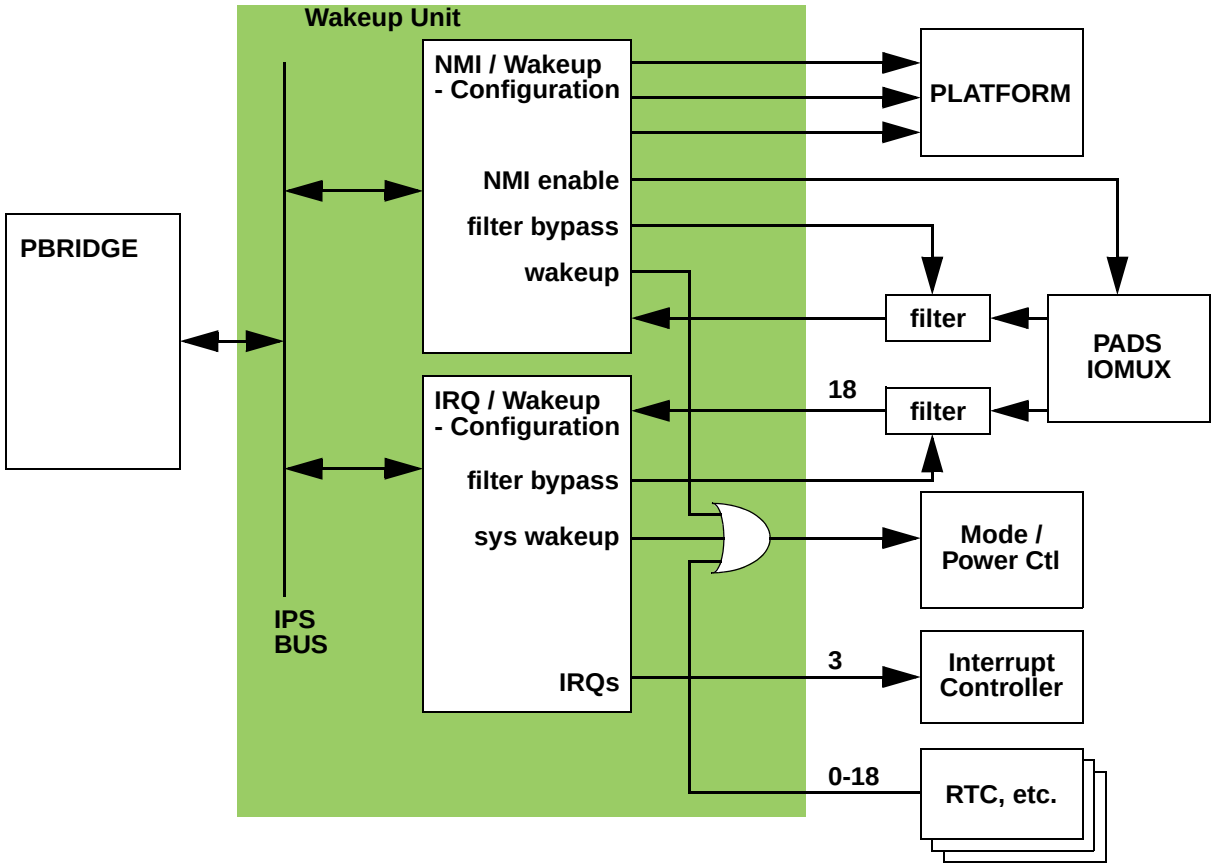


Figure 41-1. Wakeup Unit block diagram

1. Up to 19 external sources in the 176-pin and the 208BGA packages; up to 17 external sources in the 144-pin packages

Table 41-1. Wakeup vector mapping

Wakeup Vector	Wakeup Number	Function					Package		
		Port	#1	#2	#3	Special	144	176	208
0	WKPU0	PA0	DCU_R0	eMIOSA22	SOUND	FP23	x	x	x
	WKPU1	PB1	CANRX_0	PDI0			x	x	x
	WKPU2	PB3	RXD_0				x	x	x
	WKPU3	PB4	SCK_1	MA0			x	x	x
	WKPU4	PB9	SCK_0	eMIOSB20			x	x	x
	WKPU5	PB10	CANRX_1	PDI2	eMIOSA23		x	x	x
	WKPU6	PB12	RXD_1	eMIOSB19	PCS2_0		x	x	x
1	WKPU7	PC0	AN0				x	x	x
	WKPU8	PC10	AN10(mux)	SOUND			x	x	x
	WKPU9	PF0	eMIOSA13	PDI4	eMIOSA22	FP39	x	x	x
	WKPU10	PF2	NMI				x	x	x
	WKPU11	PF3	eMIOSA11	PDI6		FP37	x	x	x
	WKPU12	PF5	eMIOSA9	DCU_TAG		FP35	x	x	x
	WKPU13	PF6	SDA_0			FP34	x	x	x
2	WKPU14	PF8	SDA_1	PCS1_1	RXD_1	FP32	x	x	x
	WKPU15	PF11	eMIOSB23	PCS_C1		FP28	x	x	x
	WKPU16	PF13	SIN_2	CANRX_1		FP26	x	x	x
	WKPU17	PJ4	PDI_0	CANRX_0				x	x
	WKPU18	PJ6	PDI_2	CANRX_1	eMIOSA22			x	x
	WKPU19	API					x	x	x
	WKPU20	RTC					x	x	x

41.2 Features

The Wakeup Unit supports these distinctive features:

- Non-maskable interrupt support with
  - 1 NMI source with bypassable glitch filter
  - independent interrupt destination: non-maskable interrupt, critical interrupt, or machine check request
  - edge detection
- External wakeup/interrupt support with
  - up to 3 system interrupt vectors for 18 interrupt sources
  - analog glitch filter per each wakeup line

  - independent interrupt mask
  - edge detection
  - configurable system wakeup triggering from all interrupt sources
  - configurable pullup
- On-chip wakeup support
  - up to 18 wakeup sources
  - wakeup status mapped to same register as external wakeup/interrupt status

41.3 External signal description

The Wakeup Unit has 19 signal inputs that can be used as external interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

These 19 external signal inputs include one signal input that can be used as a non-maskable interrupt source in normal run mode or a system wakeup sources in certain power down modes.

NOTE

Be aware that the Wake-up pins are enabled in ALL modes. Therefore, the Wake-up pins should be correctly terminated to ensure minimal current consumption. Any unused Wake-up signal input should be terminated by using an external pullup or pulldown, or by internal pullup enabled at WIPUER. Also, care has to be taken on packages where the Wake-up signal inputs are not bonded. For these packages you must ensure the internal pullups are enabled for those signals not bonded.

41.4 Memory map and register description

This section provides a detailed description of all registers accessible in the WKPU module.

41.4.1 Memory map

Table 41-2 gives an overview on the WKPU registers implemented.

Table 41-2. WKPU memory map

Address offset	Use	Abbreviation	Size	Supported access sizes
0x0000	NMI Status Flag Register	NSR	32	32/16/8
0x0004–0x0007	Reserved			
0x0008	NMI Configuration Register	NCR	32	32/16/8
0x000C–0x0013	Reserved			
0x0014	Wakeup/Interrupt Status Flag Register	WISR	32	32
0x0018	Interrupt Request Enable Register	IRER	32	32
0x001C	Wakeup Request Enable Register	WRER	32	32

Table 41-2. WKPU memory map (continued)

Address offset	Use	Abbreviation	Size	Supported access sizes
0x0020–0x0027	Reserved			
0x0028	Wakeup/Interrupt Rising-Edge Event Enable Register	WIREER	32	32
0x002C	Wakeup/Interrupt Falling-Edge Event Enable Register	WIFEER	32	32
0x0030	Wakeup/Interrupt Filter Enable Register	WIFER	32	32
0x0034	Wakeup/Interrupt Pullup Enable Register	WIPUER	32	32
0x0038–0x03FFF	Reserved			

NOTE

Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

41.4.2 Register description

This section describes in address order all the Wakeup Unit registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. The numbering convention of register is MSB = 0, however the numbering of internal field is LSB = 0. For example, EIF[5] = WISR[26].

41.4.2.1 NMI Status Flag Register (NSR)

This register holds the non-maskable interrupt status flags.

Address: Base + 0x0000 Access: User read/write (write 1 to clear)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NIF	NOVF	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c	w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 41-2. NMI Status Flag Register (NSR)

Table 41-3. NSR field descriptions

Field	Description
0 NIF	NMI Status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE or NFEE set), NIF causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by NREE and NFEE has occurred
1 NOVF	NMI Overrun Status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF value whenever an NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (NREE or NFEE set), NOVF causes an interrupt request. 0 No overrun has occurred on NMI input 1 An overrun has occurred on NMI input

41.4.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Address: Base + 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NLOCK	NDSS		NWRE	0	NREE	NFEE	NFE	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 41-3. NMI Configuration Register (NCR)

Table 41-4. NCR field descriptions

Field	Description
0 NLOCK	NMI Configuration Lock Register Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
1-2 NDSS	NMI Destination Source Select 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 Reserved—no NMI, critical interrupt, or machine check request generated
3 NWRE[x]	NMI Wakeup Request Enable 0 System wakeup requests from the corresponding NIF bit are disabled 1 A set NIF bit or set NOVF bit causes a system wakeup request
5 NREE	NMI Rising-edge Events Enable 0 Rising-edge event is disabled 1 Rising-edge event is enabled

Table 41-4. NCR field descriptions (continued)

Field	Description
6 NFEE	NMI Falling-edge Events Enable 0 Falling-edge event is disabled 0 Falling-edge event is disabled 1 Falling-edge event is enabled
7 NFE	NMI Filter Enable Enable analog glitch filter on the NMI pad input. 0 Filter is disabled 1 Filter is enabled

NOTE

Writing a 0 to both NREE and NFEE disables the NMI functionality completely (i.e. no system wakeup or interrupt will be generated on any pad activity)!

41.4.2.3 Wakeup/Interrupt Status Flag Register (WISR)

This register holds the wakeup/interrupt flags.

Address : Base + 0x0014 Access: User read/write (write 1 to clear)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EIF[20:0][1]																				
W												w1c																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-4. Wakeup/Interrupt Status Flag Register (WISR)

[1] Not all bits are available in the 144-pin package.

Table 41-5. WISR field descriptions

Field	Description
EIF[x]	External Wakeup/Interrupt Status Flag x. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER[x]), EIF[x] causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by WIREER and WIFEER has occurred

NOTE

Status bits associated with on-chip wakeup sources are located to the left of the external wakeup/interrupt status bits and are read only. The wakeup for these sources must be configured and cleared at the on-chip wakeup source. Also, the configuration registers for the external interrupts/wakeups do not have corresponding bits.

41.4.2.4 Interrupt Request Enable Register (IRER)

This register is used to enable the interrupt messaging from the wakeup/interrupt pads to the interrupt controller.

Address: Base + 0x0018 Access: User read/write

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

Figure 41-5. Interrupt Request Enable Register (IRER)

[1] Not all bits are available in the 144-pin package.

Table 41-6. IRER field descriptions

Field	Description
EIRE[x]	External Interrupt Request Enable x. 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 A set EIF[x] bit causes an interrupt request

41.4.2.5 Wakeup Request Enable Register (WRER)

This register is used to enable the system wakeup messaging from the wakeup/interrupt pads to the mode entry and power control modules.

Address: Base + 0x001C Access: User read/write

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

Figure 41-6. Wakeup Request Enable Register (WRER)

[1] Not all bits are available in the 144-pin package.

Table 41-7. WRER field descriptions

Field	Description
WRE[x]	External Wakeup Request Enable x. 0 System wakeup requests from the corresponding EIF[x] bit are disabled 1 A set EIF[x] bit causes a system wakeup request

41.4.2.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

This register is used to enable rising-edge triggered events on the corresponding wakeup/interrupt pads.

Note: The RC_API can only be configured on the rising edge.

Address: Base + 0x0028 Access: User read/write

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

Figure 41-7. Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

[1] Not all bits are available in the 144-pin package.

Table 41-8. WIREER field descriptions

Field	Description
IREE[x]	External Interrupt Rising-edge Events Enable x. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

41.4.2.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

This register is used to enable falling-edge triggered events on the corresponding wakeup/interrupt pads.

Address: Base + 0x002C Access: User read/write

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

Figure 41-8. Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

[1] Not all bits are available in the 144-pin package.

Table 41-9. WIFEER field descriptions

Field	Description
IFEEx	External Interrupt Falling-edge Events Enable x. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

41.4.2.8 Wakeup/Interrupt Filter Enable Register (WIFER)

This register is used to enable an analog filter on the corresponding interrupt pads to filter out glitches on the inputs.

NOTE

There is no analog filter for the RC_API.

Address: Base + 0x0030 Access: User read/write

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

Figure 41-9. Wakeup/Interrupt Filter Enable Register (WIFER)

[1] Not all bits are available in the 144-pin package.

Table 41-10. WIFER field descriptions

Field	Description
IFE[x]	External Interrupt Filter Enable x. Enable analog glitch filter on the external interrupt pad input. 0 Filter is disabled 1 Filter is enabled

41.4.2.9 Wakeup/Interrupt Pullup Enable Register (WIPUER)

This register is used to enable a pullup on the corresponding interrupt pads to pull an unconnected wakeup/interrupt input to a value of 1.

Address: Base + 0x0034 Access: User read/write

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

Figure 41-10. Wakeup/Interrupt Pullup Enable Register (WIPUER)

[1] Not all bits are available in the 144-pin package.

Table 41-11. WIPUER field descriptions

Field	Description
IPUE[x]	External Interrupt Pullup Enable x. 0 Pullup is disabled 1 Pullup is enabled

41.5 Functional description

41.5.1 General

This section provides a complete functional description of the Wakeup Unit.

41.5.2 Non-Maskable Interrupts

The Wakeup Unit supports one non-maskable interrupt which is allocated to pin 37 of the 144-pin packages, to pin 45 of the 176-pin packages and to pin L3 of the BGA packages.

The Wakeup Unit supports the generation of three types of interrupts from the NMI. The Wakeup Unit supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

Each NMI passes through a bypassable analog glitch filter.

NOTE

Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

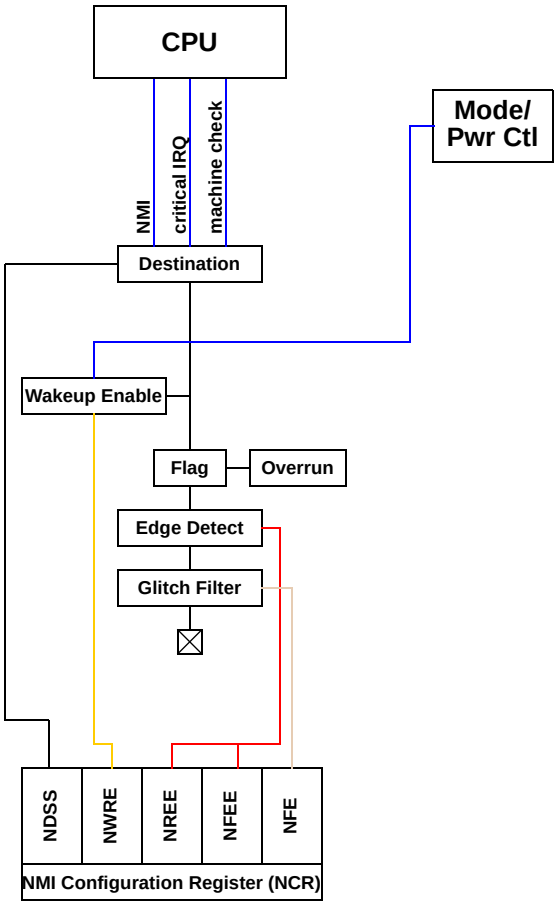


Figure 41-11. NMI Pad Diagram

41.5.2.1 NMI Management

The NMI can be enabled or disabled using the single NCR register laid out to contain all configuration bits for an NMI in a single byte (see Figure 41-3). The pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

NOTE

After reset, NREE and NFEE are set to 0, therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once the pad's NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See Table 41-4 for details.

An NMI supports a status flag and an overrun flag which are located in the NSR register (see Table 41-2). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (i.e. has not yet been cleared).

NOTE

The overrun flag is cleared by writing a 1 to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

41.5.3 External Wakeups/Interrupts

The Wakeup Unit supports up to 19 external wakeup/interrupts which can be allocated to any pad necessary at the SoC level. This allocation is fixed per SoC.

The Wakeup Unit supports three interrupt vectors to the interrupt controller of the SoC. Each interrupt vector can support up to the number of external interrupt sources from the device pads with the total across all vectors being equal to the number of external interrupt sources. Each external interrupt source is assigned to exactly one interrupt vector. The interrupt vector assignment is sequential so that one interrupt vector is for external interrupt sources 0 through N-1, the next is for N through N+M-1, and so forth.

Refer to Figure 41-12 for an overview of the external interrupt implementation for the example of three interrupt vectors with up to eight external interrupt sources each.

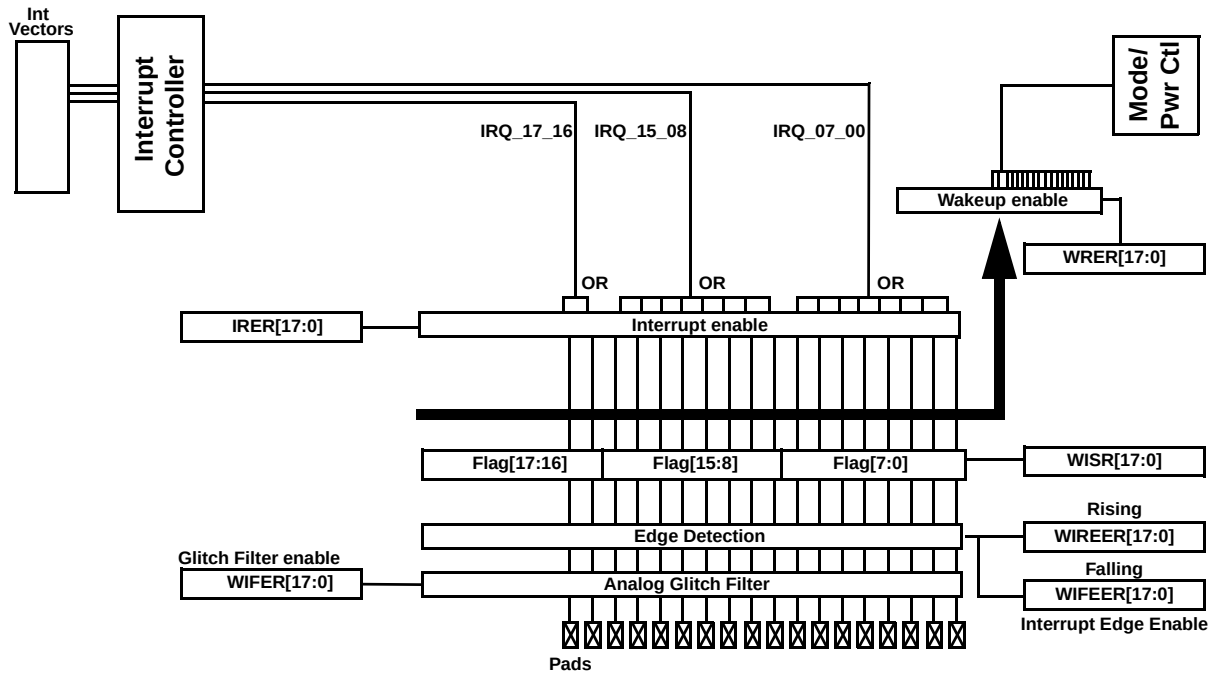


Figure 41-12. External Interrupt Pad Diagram

All of the external interrupt pads within a single group have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

NOTE

Glitch filter control and pad configuration should be done while the external interrupt line is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

41.5.3.1 External Interrupt Management

Each external interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (Table 41-5). A pad defined as an external interrupt can be configured by the user to recognize external interrupts with an active rising edge, an active falling edge or both edges being active.

NOTE

Writing a 0 to both IREE[x] and IFEE[x] disables the external interrupt functionality for that pad completely (i.e. no system wakeup or interrupt will be generated on any activity on that pad)!

The active IRQ edge is controlled by the users through the configuration of the registers WIREER and WIFEER.

Each external interrupt supports an individual flag which is held in the flag register (WISR). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

41.5.4 On-Chip Wakeups

The Wakeup Unit supports one on-chip wakeup source. It combines the on-chip wakeups with the external ones to generate a single wakeup to the system.

41.5.4.1 On-Chip Wakeup Management

In order to allow software to determine the wakeup source at one location, on-chip wakeups are reported along with external wakeups in the WISR register (see Table 41-4 for details). Enabling and clearing of these wakeups are done via the on-chip wakeup source's own registers.

Appendix A
Registers Under Protection

For this device the Register Protection module is operable on the registers of Table A-1.

NOTE

Registers protected in the DCU are described in Chapter 12, Display Control Unit (DCU), in Section 12.6.2, List of protected registers.

Table A-1. Register Under Protection

Module	Register	Register size	Module Base	Register Offset	Protected bitfields
Code Flash, 4 registers to protect					
Code Flash	MCR	32	C3F88000	000	bits[0:31]
Code Flash	BIU0	32	C3F88000	01C	bits[0:31]
Code Flash	BIU1	32	C3F88000	020	bits[0:31]
Code Flash	BIU2	32	C3F88000	024	bits[0:31]
Data Flash,1 registers to protect					
Data Flash	MCR	32	C3F8C000	000	bits[0:31]
SIU lite, 64 registers to protect					
SIUL	IRER	32	C3F90000	018	bits[0:31]
SIUL	IREER	32	C3F90000	028	bits[0:31]
SIUL	IFEER	32	C3F90000	02C	bits[0:31]
SIUL	IFER	32	C3F90000	030	bits[0:31]
SIUL	PCR0	16	C3F90000	040	bits[0:15]
SIUL	PCR1	16	C3F90000	042	bits[0:15]
SIUL	PCR2	16	C3F90000	044	bits[0:15]
SIUL	PCR3	16	C3F90000	046	bits[0:15]
SIUL	PCR4	16	C3F90000	048	bits[0:15]
SIUL	PCR5	16	C3F90000	04A	bits[0:15]
SIUL	PCR6	16	C3F90000	04C	bits[0:15]
SIUL	PCR7	16	C3F90000	04E	bits[0:15]
SIUL	PCR8	16	C3F90000	050	bits[0:15]
SIUL	PCR9	16	C3F90000	052	bits[0:15]
SIUL	PCR10	16	C3F90000	054	bits[0:15]
SIUL	PCR11	16	C3F90000	056	bits[0:15]
SIUL	PCR12	16	C3F90000	058	bits[0:15]

Table A-1. Register Under Protection (continued)

Module	Register	Register size	Module Base	Register Offset	Protected bitfields
SIUL	PCR13	16	C3F90000	05A	bits[0:15]
SIUL	PCR14	16	C3F90000	05C	bits[0:15]
SIUL	PCR15	16	C3F90000	05E	bits[0:15]
SIUL	PCR16	16	C3F90000	060	bits[0:15]
SIUL	PCR17	16	C3F90000	062	bits[0:15]
SIUL	PCR18	16	C3F90000	064	bits[0:15]
SIUL	PCR19	16	C3F90000	066	bits[0:15]
SIUL	PCR34	16	C3F90000	084	bits[0:15]
SIUL	PCR35	16	C3F90000	086	bits[0:15]
SIUL	PCR36	16	C3F90000	088	bits[0:15]
SIUL	PCR37	16	C3F90000	08A	bits[0:15]
SIUL	PCR38	16	C3F90000	08C	bits[0:15]
SIUL	PCR39	16	C3F90000	08E	bits[0:15]
SIUL	PCR40	16	C3F90000	090	bits[0:15]
SIUL	PCR41	16	C3F90000	092	bits[0:15]
SIUL	PCR42	16	C3F90000	094	bits[0:15]
SIUL	PCR43	16	C3F90000	096	bits[0:15]
SIUL	PCR44	16	C3F90000	098	bits[0:15]
SIUL	PCR45	16	C3F90000	09A	bits[0:15]
SIUL	PCR46	16	C3F90000	09C	bits[0:15]
SIUL	PCR47	16	C3F90000	09E	bits[0:15]
SIUL	PSMI0	8	C3F90000	500	bits[0:31]
SIUL	PSMI4	8	C3F90000	504	bits[0:31]
SIUL	PSMI8	8	C3F90000	508	bits[0:31]
SIUL	PSMI12	8	C3F90000	50C	bits[0:31]
SIUL	PSMI16	8	C3F90000	510	bits[0:31]
SIUL	IFMC0	32	C3F90000	1000	bits[0:31]
SIUL	IFMC1	32	C3F90000	1004	bits[0:31]
SIUL	IFMC2	32	C3F90000	1008	bits[0:31]
SIUL	IFMC3	32	C3F90000	100C	bits[0:31]
SIUL	IFMC4	32	C3F90000	1010	bits[0:31]
SIUL	IFMC5	32	C3F90000	1014	bits[0:31]

Table A-1. Register Under Protection (continued)

Module	Register	Register size	Module Base	Register Offset	Protected bitfields
SIUL	IFMC6	32	C3F90000	1018	bits[0:31]
SIUL	IFMC7	32	C3F90000	101C	bits[0:31]
SIUL	IFMC8	32	C3F90000	1020	bits[0:31]
SIUL	IFMC9	32	C3F90000	1024	bits[0:31]
SIUL	IFMC10	32	C3F90000	1028	bits[0:31]
SIUL	IFMC11	32	C3F90000	102C	bits[0:31]
SIUL	IFMC12	32	C3F90000	1030	bits[0:31]
SIUL	IFMC13	32	C3F90000	1034	bits[0:31]
SIUL	IFCPR	32	C3F90000	1080	bits[0:31]
MC Mode Entry, 41 registers to protect					
MC_ME	ME_ME	32	C3FDC000	008	bits[0:31]
MC_ME	ME_IM	32	C3FDC000	010	bits[0:31]
MC_ME	ME_TEST_MC	32	C3FDC000	024	bits[0:31]
MC_ME	ME_SAFE_MC	32	C3FDC000	028	bits[0:31]
MC_ME	ME_DRUN_MC	32	C3FDC000	02C	bits[0:31]
MC_ME	ME_RUN0_MC	32	C3FDC000	030	bits[0:31]
MC_ME	ME_RUN1_MC	32	C3FDC000	034	bits[0:31]
MC_ME	ME_RUN2_MC	32	C3FDC000	038	bits[0:31]
MC_ME	ME_RUN3_MC	32	C3FDC000	03C	bits[0:31]
MC_ME	ME_HALT0_MC	32	C3FDC000	040	bits[0:31]
MC_ME	ME_STOP0_MC	32	C3FDC000	048	bits[0:31]
MC_ME	ME_STANDBY0_MC	32	C3FDC000	054	bits[0:31]
MC_ME	ME_RUN_PC0	32	C3FDC000	080	bits[0:31]
MC_ME	ME_RUN_PC1	32	C3FDC000	084	bits[0:31]
MC_ME	ME_RUN_PC2	32	C3FDC000	088	bits[0:31]
MC_ME	ME_RUN_PC3	32	C3FDC000	08C	bits[0:31]
MC_ME	ME_RUN_PC4	32	C3FDC000	090	bits[0:31]
MC_ME	ME_RUN_PC5	32	C3FDC000	094	bits[0:31]
MC_ME	ME_RUN_PC6	32	C3FDC000	098	bits[0:31]
MC_ME	ME_RUN_PC7	32	C3FDC000	09C	bits[0:31]
MC_ME	ME_LP_PC0	32	C3FDC000	0A0	bits[0:31]
MC_ME	ME_LP_PC1	32	C3FDC000	0A4	bits[0:31]

Table A-1. Register Under Protection (continued)

Module	Register	Register size	Module Base	Register Offset	Protected bitfields
MC_ME	ME_LP_PC2	32	C3FDC000	0A8	bits[0:31]
MC_ME	ME_LP_PC3	32	C3FDC000	0AC	bits[0:31]
MC_ME	ME_LP_PC4	32	C3FDC000	0B0	bits[0:31]
MC_ME	ME_LP_PC5	32	C3FDC000	0B4	bits[0:31]
MC_ME	ME_LP_PC6	32	C3FDC000	0B8	bits[0:31]
MC_ME	ME_LP_PC7	32	C3FDC000	0BC	bits[0:31]
MC_ME	ME_PCTL[4..7]	32	C3FDC000	0C4	bits[0:31]
MC_ME	ME_PCTL[16..19]	32	C3FDC000	0D0	bits[0:31]
MC_ME	ME_PCTL[20..23]	32	C3FDC000	0D4	bits[0:31]
MC_ME	ME_PCTL[32..35]	32	C3FDC000	0E0	bits[0:31]
MC_ME	ME_PCTL[44..47]	32	C3FDC000	0EC	bits[0:31]
MC_ME	ME_PCTL[48..51]	32	C3FDC000	0F0	bits[0:31]
MC_ME	ME_PCTL[56..59]	32	C3FDC000	0F8	bits[0:31]
MC_ME	ME_PCTL[60..63]	32	C3FDC000	0FC	bits[0:31]
MC_ME	ME_PCTL[68..71]	32	C3FDC000	104	bits[0:31]
MC_ME	ME_PCTL[72..75]	32	C3FDC000	108	bits[0:31]
MC_ME	ME_PCTL[88.91]	32	C3FDC000	118	bits[0:31]
MC_ME	ME_PCTL[92..95]	32	C3FDC000	11C	bits[0:31]
MC_ME	ME_PCTL[104..107]	32	C3FDC000	128	bits[0:31]
MC Clock Generation Module, 3 registers to protect					
MC_CGM	CGM_OC_EN	8	C3FE0000	373	bits[0:7]
MC_CGM	CGM_OCDS_SC	8	C3FE0000	374	bits[0:7]
MC_CGM	CGM_SC_DC[0..3]	32	C3FE0000	37C	bits[0:31]
CMU 0, 1 register to protect					
CMU 0	CMU_CSR	32	C3FE0100	000	bits[24:31]
MC Reset Generation Module, 9 registers to protect					
MC_RGM	RGM_FES	16	C3FE4000	000	bits[0:15]
MC_RGM	RGM_DES	16	C3FE4000	002	bits[0:15]
MC_RGM	RGM_FERD	16	C3FE4000	004	bits[0:15]
MC_RGM	RGM_DERD	16	C3FE4000	006	bits[0:15]
MC_RGM	RGM_FEAR	16	C3FE4000	010	bits[0:15]

Table A-1. Register Under Protection (continued)

Module	Register	Register size	Module Base	Register Offset	Protected bitfields
MC_RGM	RGM_DEAR	16	C3FE4000	012	bits[0:15]
MC_RGM	RGM_FESS	16	C3FE4000	018	bits[0:15]
MC_RGM	RGM_STDBY	16	C3FE4000	01A	bits[0:15]
MC_RGM	RGM_FBRE	16	C3FE4000	01C	bits[0:15]
MC Power Control Unit, 1 registers to protect					
MC_PCU	PCONF2	32	C3FE8000	008	bits[0:31]

Appendix B
Register Map

Table B-1. Module base addresses

Module name	Base addresses	Location
ADC 0	0xFFE0_0000	on page 1288
CAN sampler	0xFFE7_0000	on page 1295
Clock Generation Module (MC_CGM)	0xC3FE_0370	on page 1286
CMU0	0xC3FE_0100	on page 1285
Code Flash Configuration	0xC3F8_8000	on page 1258
Data Flash 0 Configuration	0xC3F8_C000	on page 1259
DCU	0xFFE7_C000	See chapter
DMA_MUX	0xFFFD_C000	See chapter
DSPI 0	0xFFF9_0000	on page 1305
DSPI 1	0xFFF9_4000	on page 1305
ECSM	0xFFF4_0000	on page 1299
eMIOS 0	0xC3FA_0000	on page 1268
eMIOS 1	0xC3FA_4000	on page 1275
FlexCan 0 (CAN0)	0xFFFC_0000	on page 1306
FlexCan 1 (CAN1)	0xFFFC_4000	on page 1310
I2C 0	0xFFE3_0000	on page 1290
I2C 1	0xFFE3_4000	on page 1290
I2C 2	0xFFE3_8000	on page 1290
I2C 3	0xFFE3_C000	on page 1290
INTC	0xFFF4_8000	on page 1300
LINFlex 0	0xFFE4_0000	on page 1291
LINFlex 1	0xFFE4_4000	on page 1294
Mode Entry Module (MC_ME)	0xC3FD_C000	on page 1282
MPU	0xFFF1_0000	on page 1296
OSC	0xC3FE_0000	on page 1285
Periodic Interrupt Timer (PIT/RTI)	0xC3FF_0000	on page 1287
PLL0	0xC3FE_00A0	on page 1285
Power Control Unit (MC_PCU)	0xC3FE_8000	on page 1286
QuadSPI	0xFFFA_8000	See chapter
RCOSC16M	0xC3FE_0060	on page 1285

Table B-1. Module base addresses (continued)

Module name	Base addresses	Location
RCOSC128K	0xC3FE_0080	on page 1285
Real Time Counter (RTC/API)	0xC3FE_C000	on page 1287
Reset Generation Module (MC_RGM)	0xC3FE_4000	on page 1286
SGL	0xFFE7_8000	See chapter
SMC	0xFFE6_0000	See chapter
SSD 0	0xFFE6_1000	See chapter
SSD 1	0xFFE6_1800	See chapter
SSD 2	0xFFE6_2000	See chapter
SSD 3	0xFFE6_2800	See chapter
SSD 4	0xFFE6_3000	See chapter
SSD 5	0xFFE6_3800	See chapter
STM	0xFFF3_C000	on page 1298
SWT 0	0xFFF3_8000	on page 1298
SXOSC	0xC3FE_0040	on page 1285
System Integration Unit Lite (SIUL)	0xC3F9_0000	on page 1259
System Status and Configuration Module (SSCM)	0xC3FD_8000	on page 1281
WakeUp Unit	0xC3F9_4000	on page 1259

Table B-2. Detailed register map

Register description	Register Name	Used Size	Address
Program FLASH 0 Configuration Section 17.4.3, Memory map and register definition			**0xC3F8_8000**
Module Configuration Register	CFLASH_MCR	32-bit	Base + 0x0000
Low/Mid Address Space Block Locking Register	CFLASH_LML	32-bit	Base + 0x0004
High Address Space Block Locking Register	CFLASH_HBL	32-bit	Base + 0x0008
Secondary Low/Mid Address Space Block Locking Register	CFLASH_SLL	32-bit	Base + 0x000C
Low/Mid Address Space Block Select Register	CFLASH_LMS	32-bit	Base + 0x0010
High Address Space Block Select Register	CFLASH_HBS	32-bit	Base + 0x0014
Address Register	CFLASH_ADR	32-bit	Base + 0x0018
Bus Interface Unit Register 0	CFLASH_BIU0	32-bit	Base + 0x001C
Bus Interface Unit Register 1	CFLASH_BIU1	32-bit	Base + 0x0020
Bus Interface Unit Register 2	CFLASH_BIU2	32-bit	Base + 0x0024
Reserved	—	—	Base + (0x0028–0x003B)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
User Test Register 0	CFLASH_UT0	32-bit	Base + 0x003C
User Test Register 1	CFLASH_UT1	32-bit	Base + 0x0040
User Test Register 2	CFLASH_UT2	32-bit	Base + 0x0044
User Multiple Input Signature Register 0	CFLASH_UMISR0	32-bit	Base + 0x0048
User Multiple Input Signature Register 1	CFLASH_UMISR1	32-bit	Base + 0x004C
User Multiple Input Signature Register 2	CFLASH_UMISR2	32-bit	Base + 0x0050
User Multiple Input Signature Register 3	CFLASH_UMISR3	32-bit	Base + 0x0054
User Multiple Input Signature Register 4	CFLASH_UMISR4	32-bit	Base + 0x0058
Data FLASH 0 Configuration Section 17.4.3.2, Register descriptions			**0xC3F8_C000**
Module Configuration Register	CFLASH_MCR	32-bit	Base + 0x0000
Low/Mid Address Space Block Locking Register	DFLASH_LML	32-bit	Base + 0x0004
High Address Space Block Locking Register	DFLASH_HBL	32-bit	Base + 0x0008
Secondary Low/Mid Address Space Block Locking Register	DFLASH_SLL	32-bit	Base + 0x000C
Low/Mid Address Space Block Select Register	DFLASH_LMS	32-bit	Base + 0x0010
High Address Space Block Select Register	DFLASH_HBS	32-bit	Base + 0x0014
Address Register	DFLASH_ADR	32-bit	Base + 0x0018
Reserved	—	—	Base + (0x001C–0x003B)
User Test Register 0	DFLASH_UT0	32-bit	Base + 0x003C
User Test Register 1	DFLASH_UT1	32-bit	Base + 0x0040
User Test Register 2	DFLASH_UT2	32-bit	Base + 0x0044
User Multiple Input Signature Register 0	DFLASH_UMISR0	32-bit	Base + 0x0048
User Multiple Input Signature Register 1	DFLASH_UMISR1	32-bit	Base + 0x004C
User Multiple Input Signature Register 2	DFLASH_UMISR2	32-bit	Base + 0x0050
User Multiple Input Signature Register 3	DFLASH_UMISR3	32-bit	Base + 0x0054
User Multiple Input Signature Register 4	DFLASH_UMISR4	32-bit	Base + 0x0058
System Integration Unit Lite (SIUL) Section 37.5, Memory map and register description			**0xC3F9_0000**
Reserved	—	—	Base + (0x0000–0x0003)
MCU ID Register 1	MIDR1	32-bit	Base + 0x0004
MCU ID Register 2	MIDR2	32-bit	Base + 0x0008
Reserved	—	—	Base + (0x000C–0x0013)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Interrupt Status Flag Register	ISR	32-bit	Base + 0x0014
Interrupt Request Enable Register	IRER	32-bit	Base + 0x0018
Reserved	—	—	Base + (0x001C–0x0027)
Interrupt Rising Edge Event Enable	IREER	32-bit	Base + 0x0028
Interrupt Falling-Edge Event Enable	IFEER	32-bit	Base + 0x002C
IFER Interrupt Filter Enable Register	IFER	32-bit	Base + 0x0030
Reserved	—	—	Base + (0x0034–0x003F)
Pad Configuration Register 0	PCR0	16-bit	Base + 0x0040
Pad Configuration Register 1	PCR1	16-bit	Base + 0x0042
Pad Configuration Register 2	PCR2	16-bit	Base + 0x0044
Pad Configuration Register 3	PCR3	16-bit	Base + 0x0046
Pad Configuration Register 4	PCR4	16-bit	Base + 0x0048
Pad Configuration Register 5	PCR5	16-bit	Base + 0x004A
Pad Configuration Register 6	PCR6	16-bit	Base + 0x004C
Pad Configuration Register 7	PCR7	16-bit	Base + 0x004E
Pad Configuration Register 8	PCR8	16-bit	Base + 0x0050
Pad Configuration Register 9	PCR9	16-bit	Base + 0x0052
Pad Configuration Register 10	PCR10	16-bit	Base + 0x0054
Pad Configuration Register 11	PCR11	16-bit	Base + 0x0056
Pad Configuration Register 12	PCR12	16-bit	Base + 0x0058
Pad Configuration Register 13	PCR13	16-bit	Base + 0x005A
Pad Configuration Register 14	PCR14	16-bit	Base + 0x005C
Pad Configuration Register 15	PCR15	16-bit	Base + 0x005E
Pad Configuration Register 16	PCR16	16-bit	Base + 0x0060
Pad Configuration Register 17	PCR17	16-bit	Base + 0x0062
Pad Configuration Register 18	PCR18	16-bit	Base + 0x0064
Pad Configuration Register 19	PCR19	16-bit	Base + 0x0066
Pad Configuration Register 20	PCR20	16-bit	Base + 0x0068
Pad Configuration Register 21	PCR21	16-bit	Base + 0x006A
Pad Configuration Register 22	PCR22	16-bit	Base + 0x006C
Pad Configuration Register 23	PCR23	16-bit	Base + 0x006E

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Pad Configuration Register 24	PCR24	16-bit	Base + 0x0070
Pad Configuration Register 25	PCR25	16-bit	Base + 0x0072
Pad Configuration Register 26	PCR26	16-bit	Base + 0x0074
Pad Configuration Register 27	PCR27	16-bit	Base + 0x0076
Pad Configuration Register 28	PCR28	16-bit	Base + 0x0078
Pad Configuration Register 29	PCR29	16-bit	Base + 0x007A
Pad Configuration Register 30	PCR30	16-bit	Base + 0x007C
Pad Configuration Register 31	PCR31	16-bit	Base + 0x007E
Pad Configuration Register 32	PCR32	16-bit	Base + 0x0080
Pad Configuration Register 33	PCR33	16-bit	Base + 0x0082
Pad Configuration Register 34	PCR34	16-bit	Base + 0x0084
Pad Configuration Register 35	PCR35	16-bit	Base + 0x0086
Pad Configuration Register 36	PCR36	16-bit	Base + 0x0088
Pad Configuration Register 37	PCR37	16-bit	Base + 0x008A
Pad Configuration Register 38	PCR38	16-bit	Base + 0x008C
Pad Configuration Register 39	PCR39	16-bit	Base + 0x008E
Pad Configuration Register 40	PCR40	16-bit	Base + 0x0090
Pad Configuration Register 41	PCR41	16-bit	Base + 0x0092
Pad Configuration Register 42	PCR42	16-bit	Base + 0x0094
Pad Configuration Register 43	PCR43	16-bit	Base + 0x0096
Pad Configuration Register 44	PCR44	16-bit	Base + 0x0098
Pad Configuration Register 45	PCR45	16-bit	Base + 0x009A
Pad Configuration Register 46	PCR46	16-bit	Base + 0x009C
Pad Configuration Register 47	PCR47	16-bit	Base + 0x009E
Pad Configuration Register 48	PCR48	16-bit	Base + 0x00A0
Pad Configuration Register 49	PCR49	16-bit	Base + 0x00A2
Pad Configuration Register 50	PCR50	16-bit	Base + 0x00A4
Pad Configuration Register 51	PCR51	16-bit	Base + 0x00A6
Pad Configuration Register 52	PCR52	16-bit	Base + 0x00A8
Pad Configuration Register 53	PCR53	16-bit	Base + 0x00AA
Pad Configuration Register 54	PCR54	16-bit	Base + 0x00AC
Pad Configuration Register 55	PCR55	16-bit	Base + 0x00AE

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Pad Configuration Register 56	PCR56	16-bit	Base + 0x00B0
Pad Configuration Register 57	PCR57	16-bit	Base + 0x00B2
Pad Configuration Register 58	PCR58	16-bit	Base + 0x00B4
Pad Configuration Register 59	PCR59	16-bit	Base + 0x00B6
Pad Configuration Register 60	PCR60	16-bit	Base + 0x00B8
Pad Configuration Register 61	PCR61	16-bit	Base + 0x00BA
Pad Configuration Register 62	PCR62	16-bit	Base + 0x00BC
Pad Configuration Register 63	PCR63	16-bit	Base + 0x00BE
Pad Configuration Register 64	PCR64	16-bit	Base + 0x00C0
Pad Configuration Register 65	PCR65	16-bit	Base + 0x00C2
Pad Configuration Register 66	PCR66	16-bit	Base + 0x00C4
Pad Configuration Register 67	PCR67	16-bit	Base + 0x00C6
Pad Configuration Register 68	PCR68	16-bit	Base + 0x00C8
Pad Configuration Register 69	PCR69	16-bit	Base + 0x00CA
Pad Configuration Register 70	PCR70	16-bit	Base + 0x00CC
Pad Configuration Register 71	PCR71	16-bit	Base + 0x00CE
Pad Configuration Register 72	PCR72	16-bit	Base + 0x00D0
Pad Configuration Register 73	PCR73	16-bit	Base + 0x00D2
Pad Configuration Register 74	PCR74	16-bit	Base + 0x00D4
Pad Configuration Register 75	PCR75	16-bit	Base + 0x00D6
Pad Configuration Register 76	PCR76	16-bit	Base + 0x00D8
Pad Configuration Register 77	PCR77	16-bit	Base + 0x00DA
Pad Configuration Register 78	PCR78	16-bit	Base + 0x00DC
Pad Configuration Register 79	PCR79	16-bit	Base + 0x00DE
Pad Configuration Register 80	PCR80	16-bit	Base + 0x00E0
Pad Configuration Register 81	PCR81	16-bit	Base + 0x00E2
Pad Configuration Register 82	PCR82	16-bit	Base + 0x00E4
Pad Configuration Register 83	PCR83	16-bit	Base + 0x00E6
Pad Configuration Register 84	PCR84	16-bit	Base + 0x00E8
Pad Configuration Register 85	PCR85	16-bit	Base + 0x00EA
Pad Configuration Register 86	PCR86	16-bit	Base + 0x00EC
Pad Configuration Register 87	PCR87	16-bit	Base + 0x00EE

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Pad Configuration Register 88	PCR88	16-bit	Base + 0x00F0
Pad Configuration Register 89	PCR89	16-bit	Base + 0x00F2
Pad Configuration Register 90	PCR90	16-bit	Base + 0x00F4
Pad Configuration Register 91	PCR91	16-bit	Base + 0x00F6
Pad Configuration Register 92	PCR92	16-bit	Base + 0x00F8
Pad Configuration Register 93	PCR93	16-bit	Base + 0x00FA
Pad Configuration Register 94	PCR94	16-bit	Base + 0x00FC
Pad Configuration Register 95	PCR95	16-bit	Base + 0x00FE
Pad Configuration Register 96	PCR96	16-bit	Base + 0x0100
Pad Configuration Register 97	PCR97	16-bit	Base + 0x0102
Pad Configuration Register 98	PCR98	16-bit	Base + 0x0104
Pad Configuration Register 99	PCR99	16-bit	Base + 0x0106
Pad Configuration Register 100	PCR100	16-bit	Base + 0x0108
Pad Configuration Register 101	PCR101	16-bit	Base + 0x010A
Pad Configuration Register 102	PCR102	16-bit	Base + 0x010C
Pad Configuration Register 103	PCR103	16-bit	Base + 0x010E
Pad Configuration Register 104	PCR104	16-bit	Base + 0x0110
Pad Configuration Register 105	PCR105	16-bit	Base + 0x0112
Pad Configuration Register 106	PCR106	16-bit	Base + 0x0114
Pad Configuration Register 107	PCR107	16-bit	Base + 0x0116
Pad Configuration Register 108	PCR108	16-bit	Base + 0x0118
Pad Configuration Register 109	PCR109	16-bit	Base + 0x011A
Pad Configuration Register 110	PCR110	16-bit	Base + 0x011C
Pad Configuration Register 111	PCR111	16-bit	Base + 0x011E
Pad Configuration Register 112	PCR112	16-bit	Base + 0x0120
Pad Configuration Register 113	PCR113	16-bit	Base + 0x0122
Pad Configuration Register 114	PCR114	16-bit	Base + 0x0124
Pad Configuration Register 115	PCR115	16-bit	Base + 0x0126
Pad Configuration Register 116	PCR116	16-bit	Base + 0x0128
Pad Configuration Register 117	PCR117	16-bit	Base + 0x012A
Pad Configuration Register 118	PCR118	16-bit	Base + 0x012C
Pad Configuration Register 119	PCR119	16-bit	Base + 0x012E

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Pad Configuration Register 120	PCR120	16-bit	Base + 0x0130
Pad Configuration Register 121	PCR121	16-bit	Base + 0x0132
Pad Configuration Register 122	PCR122	16-bit	Base + 0x0134
Pad Configuration Register 123	PCR123	16-bit	Base + 0x0136
Pad Configuration Register 124	PCR124	16-bit	Base + 0x0138
Pad Configuration Register 125	PCR125	16-bit	Base + 0x013A
Pad Configuration Register 126	PCR126	16-bit	Base + 0x013C
Pad Configuration Register 127	PCR127	16-bit	Base + 0x013E
Pad Configuration Register 128	PCR128	16-bit	Base + 0x0140
Pad Configuration Register 129	PCR129	16-bit	Base + 0x0142
Pad Configuration Register 130	PCR130	16-bit	Base + 0x0144
Pad Configuration Register 131	PCR131	16-bit	Base + 0x0146
Pad Configuration Register 132	PCR132	16-bit	Base + 0x0148
Reserved	—	—	Base + (0x014A–0x04FF)
Pad Selection for Multiplexed Inputs	PSMI0_3	32-bit	Base + 0x0500
Pad Selection for Multiplexed Inputs	PSMI4_7	32-bit	Base + 0x0504
Pad Selection for Multiplexed Inputs	PSMI8_11	32-bit	Base + 0x0508
Pad Selection for Multiplexed Inputs	PSMI12_15	32-bit	Base + 0x050C
Pad Selection for Multiplexed Inputs	PSMI16_19	32-bit	Base + 0x0510
Pad Selection for Multiplexed Inputs	PSMI20_23	32-bit	Base + 0x0514
Pad Selection for Multiplexed Inputs	PSMI24_27	32-bit	Base + 0x0518
Pad Selection for Multiplexed Inputs	PSMI28_31	32-bit	Base + 0x051C
Reserved	—	—	Base + (0x0520–0x05FF)
GPIO Pad Data Output Register	GPDO0_3	32-bit	Base + 0x0600
GPIO Pad Data Output Register	GPDO4_7	32-bit	Base + 0x0604
GPIO Pad Data Output Register	GPDO8_11	32-bit	Base + 0x0608
GPIO Pad Data Output Register	GPDO12_15	32-bit	Base + 0x060C
GPIO Pad Data Output Register	GPDO16_19	32-bit	Base + 0x0610
GPIO Pad Data Output Register	GPDO20_23	32-bit	Base + 0x0614
GPIO Pad Data Output Register	GPDO24_27	32-bit	Base + 0x0618
GPIO Pad Data Output Register	GPDO28_31	32-bit	Base + 0x061C

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
GPIO Pad Data Output Register	GPDO32_35	32-bit	Base + 0x0620
GPIO Pad Data Output Register	GPDO36_39	32-bit	Base + 0x0624
GPIO Pad Data Output Register	GPDO40_43	32-bit	Base + 0x0628
GPIO Pad Data Output Register	GPDO44_47	32-bit	Base + 0x062C
GPIO Pad Data Output Register	GPDO48_51	32-bit	Base + 0x0630
GPIO Pad Data Output Register	GPDO52_55	32-bit	Base + 0x0634
GPIO Pad Data Output Register	GPDO56_59	32-bit	Base + 0x0638
GPIO Pad Data Output Register	GPDO60_63	32-bit	Base + 0x063C
GPIO Pad Data Output Register	GPDO64_67	32-bit	Base + 0x0640
GPIO Pad Data Output Register	GPDO68_71	32-bit	Base + 0x0644
GPIO Pad Data Output Register	GPDO72_75	32-bit	Base + 0x0648
GPIO Pad Data Output Register	GPDO76_79	32-bit	Base + 0x064C
GPIO Pad Data Output Register	GPDO80_83	32-bit	Base + 0x0650
GPIO Pad Data Output Register	GPDO84_87	32-bit	Base + 0x0654
GPIO Pad Data Output Register	GPDO88_91	32-bit	Base + 0x0658
GPIO Pad Data Output Register	GPDO92_95	32-bit	Base + 0x065C
GPIO Pad Data Output Register	GPDO96_99	32-bit	Base + 0x0660
GPIO Pad Data Output Register	GPDO100_103	32-bit	Base + 0x0664
GPIO Pad Data Output Register	GPDO104_107	32-bit	Base + 0x0668
GPIO Pad Data Output Register	GPDO108_111	32-bit	Base + 0x066C
GPIO Pad Data Output Register	GPDO112_115	32-bit	Base + 0x0670
GPIO Pad Data Output Register	GPDO116_119	32-bit	Base + 0x0674
GPIO Pad Data Output Register	GPDO120_123	32-bit	Base + 0x0678
Reserved	—	—	Base + (0x067C–0x07FF)
GPIO Pad Data Input Register	GPDI0_3	32-bit	Base + 0x0800
GPIO Pad Data Input Register	GPDI4_7	32-bit	Base + 0x0804
GPIO Pad Data Input Register	GPDI8_11	32-bit	Base + 0x0808
GPIO Pad Data Input Register	GPDI12_15	32-bit	Base + 0x080C
GPIO Pad Data Input Register	GPDI16_19	32-bit	Base + 0x0810
GPIO Pad Data Input Register	GPDI20_23	32-bit	Base + 0x0814
GPIO Pad Data Input Register	GPDI24_27	32-bit	Base + 0x0818

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
GPIO Pad Data Input Register	GPDI28_31	32-bit	Base + 0x081C
GPIO Pad Data Input Register	GPDI32_35	32-bit	Base + 0x0820
GPIO Pad Data Input Register	GPDI36_39	32-bit	Base + 0x0824
GPIO Pad Data Input Register	GPDI40_43	32-bit	Base + 0x0828
GPIO Pad Data Input Register	GPDI44_47	32-bit	Base + 0x082C
GPIO Pad Data Input Register	GPDI48_51	32-bit	Base + 0x0830
GPIO Pad Data Input Register	GPDI52_55	32-bit	Base + 0x0834
GPIO Pad Data Input Register	GPDI56_59	32-bit	Base + 0x0838
GPIO Pad Data Input Register	GPDI60_63	32-bit	Base + 0x083C
GPIO Pad Data Input Register	GPDI64_67	32-bit	Base + 0x0840
GPIO Pad Data Input Register	GPDI68_71	32-bit	Base + 0x0844
GPIO Pad Data Input Register	GPDI72_75	32-bit	Base + 0x0848
GPIO Pad Data Input Register	GPDI76_79	32-bit	Base + 0x084C
GPIO Pad Data Input Register	GPDI80_83	32-bit	Base + 0x0850
GPIO Pad Data Input Register	GPDI84_87	32-bit	Base + 0x0854
GPIO Pad Data Input Register	GPDI88_91	32-bit	Base + 0x0858
GPIO Pad Data Input Register	GPDI92_95	32-bit	Base + 0x085C
GPIO Pad Data Input Register	GPDI96_99	32-bit	Base + 0x0860
GPIO Pad Data Input Register	GPDI100_103	32-bit	Base + 0x0864
GPIO Pad Data Input Register	GPDI104_107	32-bit	Base + 0x0868
GPIO Pad Data Input Register	GPDI108_111	32-bit	Base + 0x086C
GPIO Pad Data Input Register	GPDI112_115	32-bit	Base + 0x0870
GPIO Pad Data Input Register	GPDI116_119	32-bit	Base + 0x0874
GPIO Pad Data Input Register	GPDI120_123	32-bit	Base + 0x0878
Reserved	—	—	Base + (0x087C–0x0BF)
Parallel GPIO Pad Data Out Register	PGPDO0	32-bit	Base + 0x0C00
Parallel GPIO Pad Data Out Register	PGPDO1	32-bit	Base + 0x0C04
Parallel GPIO Pad Data Out Register	PGPDO2	32-bit	Base + 0x0C08
Parallel GPIO Pad Data Out Register	PGPDO3	32-bit	Base + 0x0C0C
Reserved	—	—	Base + (0x0C10–0x0C3)
Parallel GPIO Pad Data In Register	PGPDI0	32-bit	Base + 0x0C40

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Parallel GPIO Pad Data In Register	PGPDI1	32-bit	Base + 0x0C44
Parallel GPIO Pad Data In Register	PGPDI2	32-bit	Base + 0x0C48
Parallel GPIO Pad Data In Register	PGPDI3	32-bit	Base + 0x0C4C
Reserved	—	—	Base + (0x0C50–0x0C7)
Masked Parallel GPIO Pad Data Out Register	MPGPDO0	32-bit	Base + 0x0C80
Masked Parallel GPIO Pad Data Out Register	MPGPDO1	32-bit	Base + 0x0C84
Masked Parallel GPIO Pad Data Out Register	MPGPDO2	32-bit	Base + 0x0C88
Masked Parallel GPIO Pad Data Out Register	MPGPDO3	32-bit	Base + 0x0C8C
Masked Parallel GPIO Pad Data Out Register	MPGPDO4	32-bit	Base + 0x0C90
Masked Parallel GPIO Pad Data Out Register	MPGPDO5	32-bit	Base + 0x0C94
Masked Parallel GPIO Pad Data Out Register	MPGPDO6	32-bit	Base + 0x0C98
Masked Parallel GPIO Pad Data Out Register	MPGPDO7	32-bit	Base + 0x0C9C
Reserved	—	—	Base + (0x0CA0–0x0FF)
Interrupt Filter Maximum Counter Register	IFMC0	32-bit	Base + 0x1000
Interrupt Filter Maximum Counter Register	IFMC1	32-bit	Base + 0x1004
Interrupt Filter Maximum Counter Register	IFMC2	32-bit	Base + 0x1008
Interrupt Filter Maximum Counter Register	IFMC3	32-bit	Base + 0x100C
Interrupt Filter Maximum Counter Register	IFMC4	32-bit	Base + 0x1010
Interrupt Filter Maximum Counter Register	FMC5	32-bit	Base + 0x1014
Interrupt Filter Maximum Counter Register	IFMC6	32-bit	Base + 0x1018
Interrupt Filter Maximum Counter Register	IFMC7	32-bit	Base + 0x101C
Interrupt Filter Maximum Counter Register	IFMC8	32-bit	Base + 0x1020
Interrupt Filter Maximum Counter Register	IFMC9	32-bit	Base + 0x1024
Interrupt Filter Maximum Counter Register	IFMC10	32-bit	Base + 0x1028
Interrupt Filter Maximum Counter Register	IFMC11	32-bit	Base + 0x102C
Interrupt Filter Maximum Counter Register	IFMC12	32-bit	Base + 0x1030
Interrupt Filter Maximum Counter Register	IFMC13	32-bit	Base + 0x1034
Interrupt Filter Maximum Counter Register	IFMC14	32-bit	Base + 0x1038
Interrupt Filter Maximum Counter Register	IFMC15	32-bit	Base + 0x103C
Reserved	—	—	Base + (0x1044–0x107C)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Interrupt Filter Clock Prescaler Register	IFCP	32-bit	Base + 0x1080
Reserved	—	—	Base + (0x1084–0x3FFF)
WakeUp Unit Section 41.4, Memory map and register description			**0xC3F94000**
NMI Status Flag Register	WKPU_NSR	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x0007)
NMI Configuration Register	WKPU_NCR	32-bit	Base + 0x0008
Reserved	—	—	Base + (0x000C–0x0013)
Wakeup/Interrupt Status Flag Register	WKPU_WISR	32-bit	Base + 0x0014
Interrupt Request Enable Register	WKPU_IRER	32-bit	Base + 0x0018
Wakeup Request Enable Register	WKPU_WRER	32-bit	Base + 0x001C
Reserved	—	—	Base + (0x0020–0x0027)
Wakeup/Interrupt Rising-Edge Event Enable Register	WKPU_WIREER	32-bit	Base + 0x0028
Wakeup/Interrupt Falling-Edge Event Enable Register	WKPU_WIFEER	32-bit	Base + 0x002C
Wakeup/Interrupt Filter Enable Register	WKPU_WIFER	32-bit	Base + 0x0030
Wakeup/Interrupt Pullup Enable Register	WKPU_WIPUER	32-bit	Base + 0x0034
Reserved	—	—	Base + (0x0038–0xFFFF)
eMIOS 0 Section 9.4, Memory map and register description			**0xC3FA_0000**
EMIOS Module Configuration Register	EMIOS0_MCR	32-bit	Base + 0x0000
EMIOS Global FLAG Register	EMIOS0_GFLAG	32-bit	Base + 0x0004
EMIOS Output Update Disable Register	EMIOS0_OUDIS	32-bit	Base + 0x0008
EMIOS Disable Channel Register	EMIOS0_UCDIS	32-bit	Base + 0x000C
Reserved	—	—	Base + (0x0010–0x001F)
eMIOS0 UC0 A Register	EMIOS0_UC0_A	32-bit	Base + 0x0020
eMIOS0 UC0 B Register	EMIOS0_UC0_B	32-bit	Base + 0x0024
eMIOS0 UC0 CNT	EMIOS0_UC0_CNT	32-bit	Base + 0x0028
eMIOS0 UC0 Control Register	EMIOS0_UC0_SC	32-bit	Base + 0x002C
eMIOS0 UC0 Status Register	EMIOS0_UC0_SS	32-bit	Base + 0x0030
Reserved	—	—	Base + (0x0034–0x003F)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
eMIOS0 UC1 A Register	EMIOS0_UC1_A	32-bit	Base + 0x0040
eMIOS0 UC1 B Register	EMIOS0_UC1_B	32-bit	Base + 0x0044
Reserved	—	—	Base + (0x0048–0x004B)
eMIOS0 UC1 Control Register	EMIOS0_UC1_SC	32-bit	Base + 0x004C
eMIOS0 UC1 Status Register	EMIOS0_UC1_SS	32-bit	Base + 0x0050
Reserved	—	—	Base + (0x0054–0x005F)
eMIOS0 UC2 A Register	EMIOS0_UC2_A	32-bit	Base + 0x0060
eMIOS0 UC2 B Register	EMIOS0_UC2_B	32-bit	Base + 0x0064
Reserved	—	—	Base + (0x0068–0x006B
eMIOS0 UC2 Control Register	EMIOS0_UC2_SC	32-bit	Base + 0x006C
eMIOS0 UC2 Status Register	EMIOS0_UC2_SS	32-bit	Base + 0x0070
Reserved	—	—	Base + (0x0074–0x007F)
eMIOS0 UC3 A Register	EMIOS0_UC3_A	32-bit	Base + 0x0080
eMIOS0 UC3 B Register	EMIOS0_UC3_B	32-bit	Base + 0x0084
Reserved	—	—	Base + (0x0088–0x008B)
eMIOS0 UC3 Control Register	EMIOS0_UC3_SC	32-bit	Base + 0x008C
eMIOS0 UC3 Status Register	EMIOS0_UC3_SS	32-bit	Base + 0x0090
Reserved	—	—	Base + (0x0094–0x009F)
eMIOS0 UC4 A Register	EMIOS0_UC4_A	32-bit	Base + 0x00A0
eMIOS0 UC4 B Register	EMIOS0_UC4_B	32-bit	Base + 0x00A4
Reserved	—	—	Base + (0x00A8–0x00AB)
eMIOS0 UC4 Control Register	EMIOS0_UC4_SC	32-bit	Base + 0x00AC
eMIOS0 UC4 Status Register	EMIOS0_UC4_SS	32-bit	Base + 0x00B0
Reserved	—	—	Base + (0x00B4–0x00BF)
eMIOS0 UC5 A Register	EMIOS0_UC5_A	32-bit	Base + 0x00C0
eMIOS0 UC5 B Register	EMIOS0_UC5_B	32-bit	Base + 0x00C4

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x00C8–0x00CB)
eMIOS0 UC5 Control Register	EMIOS0_UC5_SC	32-bit	Base + 0x00CC
eMIOS0 UC5 Status Register	EMIOS0_UC5_SS	32-bit	Base + 0x00D0
Reserved	—	—	Base + (0x00D4–0x00DF)
eMIOS0 UC6 A Register	EMIOS0_UC6_A	32-bit	Base + 0x00E0
eMIOS0 UC6 B Register	EMIOS0_UC6_B	32-bit	Base + 0x00E4
Reserved	—	—	Base + (0x00E8–0x00EB)
eMIOS0 UC6 Control Register	EMIOS0_UC6_SC	32-bit	Base + 0x00EC
eMIOS0 UC6 Status Register	EMIOS0_UC6_SS	32-bit	Base + 0x00F0
Reserved	—	—	Base + (0x00F4–0x00FF)
eMIOS0 UC7 A Register	EMIOS0_UC7_A	32-bit	Base + 0x0100
eMIOS0 UC7 B Register	EMIOS0_UC7_B	32-bit	Base + 0x0104
Reserved	—	—	Base + (0x0108–0x010B)
eMIOS0 UC7 Control Register	EMIOS0_UC7_SC	32-bit	Base + 0x010C
eMIOS0 UC7 Status Register	EMIOS0_UC7_SS	32-bit	Base + 0x0110
Reserved	—	—	Base + (0x0114–0x011F)
eMIOS0 UC8 A Register	EMIOS0_UC8_A	32-bit	Base + 0x0120
eMIOS0 UC8 B Register	EMIOS0_UC8_B	32-bit	Base + 0x0124
eMIOS0 UC8 CNT	EMIOS0_UC8_CNT	32-bit	Base + 0x0128
eMIOS0 UC8 Control Register	EMIOS0_UC8_SC	32-bit	Base + 0x012C
eMIOS0 UC8 Status Register	EMIOS0_UC8_SS	32-bit	Base + 0x0130
Reserved	—	—	Base + (0x0134–0x013F)
eMIOS0 UC9 A Register	EMIOS0_UC9_A	32-bit	Base + 0x0140
eMIOS0 UC9 B Register	EMIOS0_UC9_B	32-bit	Base + 0x0144
Reserved	—	—	Base + (0x0148–0x014B)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
eMIOS0 UC9 Control Register	EMIOS0_UC9_SC	32-bit	Base + 0x014C
eMIOS0 UC9 Status Register	EMIOS0_UC9_SS	32-bit	Base + 0x0150
Reserved	—	—	Base + (0x0154–0x015F)
eMIOS0 UC10 A Register	EMIOS0_UC10_A	32-bit	Base + 0x0160
eMIOS0 UC10 B Register	EMIOS0_UC10_B	32-bit	Base + 0x0164
Reserved	—	—	Base + (0x0168–0x016B)
eMIOS0 UC10 Control Register	EMIOS0_UC10_SC	32-bit	Base + 0x016C
eMIOS0 UC10 Status Register	EMIOS0_UC10_SS	32-bit	Base + 0x0170
Reserved	—	—	Base + (0x0174–0x017F)
eMIOS0 UC11 A Register	EMIOS0_UC11_A	32-bit	Base + 0x0180
eMIOS0 UC11 B Register	EMIOS0_UC11_B	32-bit	Base + 0x0184
Reserved	—	—	Base + (0x0188–0x018B)
eMIOS0 UC11 Control Register	EMIOS0_UC11_SC	32-bit	Base + 0x018C
eMIOS0 UC11 Status Register	EMIOS0_UC11_SS	32-bit	Base + 0x0190
Reserved	—	—	Base + (0x0194–0x019F)
eMIOS0 UC12 A Register	EMIOS0_UC12_A	32-bit	Base + 0x01A0
eMIOS0 UC12 B Register	EMIOS0_UC12_B	32-bit	Base + 0x01A4
Reserved	—	—	Base + (0x01A8–0x01AB)
eMIOS0 UC12 Control Register	EMIOS0_UC12_SC	32-bit	Base + 0x01AC
eMIOS0 UC12 Status Register	EMIOS0_UC12_SS	32-bit	Base + 0x01B0
Reserved	—	—	Base + (0x01B4–0x01BF
eMIOS0 UC13 A Register	EMIOS0_UC13_A	32-bit	Base + 0x01C0
eMIOS0 UC13 B Register	EMIOS0_UC13_B	32-bit	Base + 0x01C4
Reserved	—	—	Base + (0x01C8–0x01CB)
eMIOS0 UC13 Control Register	EMIOS0_UC13_SC	32-bit	Base + 0x01CC
eMIOS0 UC13 Status Register	EMIOS0_UC13_SS	32-bit	Base + 0x01D0

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x01D4–0x01DF)
eMIOS0 UC14 A Register	EMIOS0_UC14_A	32-bit	Base + 0x01E0
eMIOS0 UC14 B Register	EMIOS0_UC14_B	32-bit	Base + 0x01E4
Reserved	—	—	Base + (0x01E8–0x01EB)
eMIOS0 UC14 Control Register	EMIOS0_UC14_SC	32-bit	Base + 0x01EC
eMIOS0 UC14 Status Register	EMIOS0_UC14_SS	32-bit	Base + 0x01F0
Reserved	—	—	Base + (0x01F4–0x01FF)
eMIOS0 UC15 A Register	EMIOS0_UC15_A	32-bit	Base + 0x0200
eMIOS0 UC15 B Register	EMIOS0_UC15_B	32-bit	Base + 0x0204
Reserved	—	—	Base + (0x0208–0x020B)
eMIOS0 UC15 Control Register	EMIOS0_UC15_SC	32-bit	Base + 0x020C
eMIOS0 UC15 Status Register	EMIOS0_UC15_SS	32-bit	Base + 0x0210
Reserved	—	—	Base + (0x0214–0x021F)
eMIOS0 UC16 A Register	EMIOS0_UC16_A	32-bit	Base + 0x0220
eMIOS0 UC16 B Register	EMIOS0_UC16_B	32-bit	Base + 0x0224
eMIOS0 UC16 CNT	EMIOS0_UC16_CNT	32-bit	Base + 0x0228
eMIOS0 UC16 Control Register	EMIOS0_UC16_SC	32-bit	Base + 0x022C
eMIOS0 UC16 Status Register	EMIOS0_UC16_SS	32-bit	Base + 0x0230
Reserved	—	—	Base + (0x0234–0x023F)
eMIOS0 UC17 A Register	EMIOS0_UC17_A	32-bit	Base + 0x0240
eMIOS0 UC17 B Register	EMIOS0_UC17_B	32-bit	Base + 0x0244
Reserved	—	—	Base + (0x0248–0x024B)
eMIOS0 UC17 Control Register	EMIOS0_UC17_SC	32-bit	Base + 0x024C
eMIOS0 UC17 Status Register	EMIOS0_UC17_SS	32-bit	Base + 0x0250
Reserved	—	—	Base + (0x0254–0x025F)
eMIOS0 UC18 A Register	EMIOS0_UC18_A	32-bit	Base + 0x0260

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
eMIOS0 UC18 B Register	EMIOS0_UC18_B	32-bit	Base + 0x0264
Reserved	—	—	Base + (0x0268–0x026B)
eMIOS0 UC18 Control Register	EMIOS0_UC18_SC	32-bit	Base + 0x026C
eMIOS0 UC18 Status Register	EMIOS0_UC18_SS	32-bit	Base + 0x0270
Reserved	—	—	Base(_0x0274–0 x027F0)
eMIOS0 UC19 A Register	EMIOS0_UC19_A	32-bit	Base + 0x0280
eMIOS0 UC19 B Register	EMIOS0_UC19_B	32-bit	Base + 0x0284
Reserved	—	—	Base + (0x0288–0x028B)
eMIOS0 UC19 Control Register	EMIOS0_UC19_SC	32-bit	Base + 0x028C
eMIOS0 UC19 Status Register	EMIOS0_UC19_SS	32-bit	Base + 0x0290
Reserved	—	—	Base + (0x0294–0x029F)
eMIOS0 UC20 A Register	EMIOS0_UC20_A	32-bit	Base + 0x02A0
eMIOS0 UC20 B Register	EMIOS0_UC20_B	32-bit	Base + 0x02A4
Reserved	—	—	Base + (0x02A8–0x02AB)
eMIOS0 UC20 Control Register	EMIOS0_UC20_SC	32-bit	Base + 0x02AC
eMIOS0 UC20 Status Register	EMIOS0_UC20_SS	32-bit	Base + 0x02B0
Reserved	—	—	Base + (0x02B4–0x02BF)
eMIOS0 UC21 A Register	EMIOS0_UC21_A	32-bit	Base + 0x02C0
eMIOS0 UC21 B Register	EMIOS0_UC21_B	32-bit	Base + 0x02C4
Reserved	—	—	Base + (0x02C8–0x02CB)
eMIOS0 UC21 Control Register	EMIOS0_UC21_SC	32-bit	Base + 0x02CC
eMIOS0 UC21 Status Register	EMIOS0_UC21_SS	32-bit	Base + 0x02D0
Reserved	—	—	Base + (0x02D4–0x02DF)
eMIOS0 UC22 A Register	EMIOS0_UC22_A	32-bit	Base + 0x02E0
eMIOS0 UC22 B Register	EMIOS0_UC22_B	32-bit	Base + 0x02E4

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x02E8–0x02EB)
eMIOS0 UC22 Control Register	EMIOS0_UC22_SC	32-bit	Base + 0x02EC
eMIOS0 UC22 Status Register	EMIOS0_UC22_SS	32-bit	Base + 0x02F0
Reserved	—	—	Base + (0x02F4–0x02FF)
eMIOS0 UC23 A Register	EMIOS0_UC23_A	32-bit	Base + 0x0300
eMIOS0 UC23 B Register	EMIOS0_UC23_B	32-bit	Base + 0x0304
eMIOS0 UC23 CNT	EMIOS0_UC23_CNT	32-bit	Base + 0x0308
eMIOS0 UC23 Control Register	EMIOS0_UC23_SC	32-bit	Base + 0x030C
eMIOS0 UC23 Status Register	EMIOS0_UC23_SS	32-bit	Base + 0x0310
Reserved	—	—	Base + (0x0314–0x031F)
eMIOS0 UC24 A Register	EMIOS0_UC24_A	32-bit	Base + 0x0320
eMIOS0 UC24 B Register	EMIOS0_UC24_B	32-bit	Base + 0x0324
eMIOS0 UC24 CNT	EMIOS0_UC24_CNT	32-bit	Base + 0x0328
eMIOS0 UC24 Control Register	EMIOS0_UC24_SC	32-bit	Base + 0x032C
eMIOS0 UC24 Status Register	EMIOS0_UC24_SS	32-bit	Base + 0x0330
Reserved	—	—	Base + (0x0334–0x033F)
eMIOS0 UC25 A Register	EMIOS0_UC25_A	32-bit	Base + 0x0340
Reserved	—	—	Base + (0x0344–0x034B
eMIOS0 UC25 Control Register	EMIOS0_UC25_SC	32-bit	Base + 0x034C
eMIOS0 UC25 Status Register	EMIOS0_UC25_SS	32-bit	Base + 0x0350
Reserved	—	—	Base + (0x0354–0x035F)
eMIOS0 UC26 A Register	EMIOS0_UC26_A	32-bit	Base + 0x0360
Reserved	—	—	Base + (0x0364–0x036B)
eMIOS0 UC26 Control Register	EMIOS0_UC26_SC	32-bit	Base + 0x036C
eMIOS0 UC26 Status Register	EMIOS0_UC26_SS	32-bit	Base + 0x0370
Reserved	—	—	Base + (0x0374–0x037F)
eMIOS0 UC27 A Register	EMIOS0_UC27_A	32-bit	Base + 0x0380

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0384–0x038B)
eMIOS0 UC27 Control Register	EMIOS0_UC27_SC	32-bit	Base + 0x038C
eMIOS0 UC27 Status Register	EMIOS0_UC27_SS	32-bit	Base + 0x0390
Reserved	—	—	Base + (0x0394–0x03FF)
eMIOS 1 Section 9.4, Memory map and register description			**0xC3FA_4000**
EMIOS Module Configuration Register	eMIOS1_MCR	32-bit	Base + 0x0000
EMIOS Global FLAG Register	eMIOS1_GFLAG	32-bit	Base + 0x0004
EMIOS Output Update Disable Register	eMIOS1_OUDIS	32-bit	Base + 0x0008
EMIOS Disable Channel Register	eMIOS1_UCDIS	32-bit	Base + 0x000C
Reserved	—	—	Base + (0x001C–0x001F)
eMIOS1 UC0 A Register	eMIOS1_UC0_A	32-bit	Base + 0x0020
eMIOS1 UC0 B Register	eMIOS1_UC0_B	32-bit	Base + 0x0024
eMIOS1 UC0 CNT	eMIOS1_UC0_CNT	32-bit	Base + 0x0028
eMIOS1 UC0 Control Register	eMIOS1_UC0_SC	32-bit	Base + 0x002C
eMIOS1 UC0 Status Register	eMIOS1_UC0_SS	32-bit	Base + 0x0030
Reserved	—	—	Base + (0x003–0x003F)
eMIOS1 UC1 A Register	eMIOS1_UC1_A	32-bit	Base + 0x0040
eMIOS1 UC1 B Register	eMIOS1_UC1_B	32-bit	Base + 0x0044
Reserved	—	—	Base + (0x0048–0x004B)
eMIOS1 UC1 Control Register	eMIOS1_UC1_SC	32-bit	Base + 0x004C
eMIOS1 UC1 Status Register	eMIOS1_UC1_SS	32-bit	Base + 0x0050
Reserved	—	—	Base + (0x0054–0x005F)
eMIOS1 UC2 A Register	eMIOS1_UC2_A	32-bit	Base + 0x0060
eMIOS1 UC2 B Register	eMIOS1_UC2_B	32-bit	Base + 0x0064
Reserved	—	—	Base + (0x0068–0x006B)
eMIOS1 UC2 Control Register	eMIOS1_UC2_SC	32-bit	Base + 0x006C
eMIOS1 UC2 Status Register	eMIOS1_UC2_SS	32-bit	Base + 0x0070

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0074–0x007F)
eMIOS1 UC3 A Register	eMIOS1_UC3_A	32-bit	Base + 0x0080
eMIOS1 UC3 B Register	eMIOS1_UC3_B	32-bit	Base + 0x0084
Reserved	—	—	Base + (0x0088–0x008B)
eMIOS1 UC3 Control Register	eMIOS1_UC3_SC	32-bit	Base + 0x008C
eMIOS1 UC3 Status Register	eMIOS1_UC3_SS	32-bit	Base + 0x0090
Reserved	—	—	Base + (0x0094–0x009F)
eMIOS1 UC4 A Register	eMIOS1_UC4_A	32-bit	Base + 0x00A0
eMIOS1 UC4 B Register	eMIOS1_UC4_B	32-bit	Base + 0x00A4
Reserved	—	—	Base + (0x00A8–0x00AB)
eMIOS1 UC4 Control Register	eMIOS1_UC4_SC	32-bit	Base + 0x00AC
eMIOS1 UC4 Status Register	eMIOS1_UC4_SS	32-bit	Base + 0x00B0
Reserved	—	—	Base + (0x00B4–0x00BF)
eMIOS1 UC5 A Register	eMIOS1_UC5_A	32-bit	Base + 0x00C0
eMIOS1 UC5 B Register	eMIOS1_UC5_B	32-bit	Base + 0x00C4
Reserved	—	—	Base + (0x00C8–0x00CB)
eMIOS1 UC5 Control Register	eMIOS1_UC5_SC	32-bit	Base + 0x00CC
eMIOS1 UC5 Status Register	eMIOS1_UC5_SS	32-bit	Base + 0x00D0
Reserved	—	—	Base + (0x00D4–0x00DF)
eMIOS1 UC6 A Register	eMIOS1_UC6_A	32-bit	Base + 0x00E0
eMIOS1 UC6 B Register	eMIOS1_UC6_B	32-bit	Base + 0x00E4
Reserved	—	—	Base + (0x00E8–0x00EB)
eMIOS1 UC6 Control Register	eMIOS1_UC6_SC	32-bit	Base + 0x00EC
eMIOS1 UC6 Status Register	eMIOS1_UC6_SS	32-bit	Base + 0x00F0

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x00F4–0x00FF)
eMIOS1 UC7 A Register	eMIOS1_UC7_A	32-bit	Base + 0x0100
eMIOS1 UC7 B Register	eMIOS1_UC7_B	32-bit	Base + 0x0104
Reserved	—	—	Base + (0x0108–0x010B)
eMIOS1 UC7 Control Register	eMIOS1_UC7_SC	32-bit	Base + 0x010C
eMIOS1 UC7 Status Register	eMIOS1_UC7_SS	32-bit	Base + 0x0110
Reserved	—	—	Base + (0x0114–0x011F)
eMIOS1 UC8 A Register	eMIOS1_UC8_A	32-bit	Base + 0x0120
eMIOS1 UC8 B Register	eMIOS1_UC8_B	32-bit	Base + 0x0124
eMIOS1 UC8 CNT	eMIOS1_UC8_CNT	32-bit	Base + 0x0128
eMIOS1 UC8 Control Register	eMIOS1_UC8_SC	32-bit	Base + 0x012C
eMIOS1 UC8 Status Register	eMIOS1_UC8_SS	32-bit	Base + 0x0130
Reserved	—	—	Base + (0x0134–0x013F)
eMIOS1 UC9 A Register	eMIOS1_UC9_A	32-bit	Base + 0x0140
eMIOS1 UC9 B Register	eMIOS1_UC9_B	32-bit	Base + 0x0144
Reserved	—	—	Base + (0x0148–0x014B)
eMIOS1 UC9 Control Register	eMIOS1_UC9_SC	32-bit	Base + 0x014C
eMIOS1 UC9 Status Register	eMIOS1_UC9_SS	32-bit	Base + 0x0150
Reserved	—	—	Base + (0x0154–0x015F)
eMIOS1 UC10 A Register	eMIOS1_UC10_A	32-bit	Base + 0x0160
eMIOS1 UC10 B Register	eMIOS1_UC10_B	32-bit	Base + 0x0164
Reserved	—	—	Base + (0x0168–0x016B)
eMIOS1 UC10 Control Register	eMIOS1_UC10_SC	32-bit	Base + 0x016C
eMIOS1 UC10 Status Register	eMIOS1_UC10_SS	32-bit	Base + 0x0170
Reserved	—	—	Base + (0x0174–0x017F)
eMIOS1 UC11 A Register	eMIOS1_UC11_A	32-bit	Base + 0x0180
eMIOS1 UC11 B Register	eMIOS1_UC11_B	32-bit	Base + 0x0184

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0188–0x018B)
eMIOS1 UC11 Control Register	eMIOS1_UC11_SC	32-bit	Base + 0x018C
eMIOS1 UC11 Status Register	eMIOS1_UC11_SS	32-bit	Base + 0x0190
Reserved	—	—	Base + (0x0194–0x019F)
eMIOS1 UC12 A Register	eMIOS1_UC12_A	32-bit	Base + 0x01A0
eMIOS1 UC12 B Register	eMIOS1_UC12_B	32-bit	Base + 0x01A4
Reserved	—	—	Base + (0x01A8–0x01AB)
eMIOS1 UC12 Control Register	eMIOS1_UC12_SC	32-bit	Base + 0x01AC
eMIOS1 UC12 Status Register	eMIOS1_UC12_SS	32-bit	Base + 0x01B0
Reserved	—	—	Base + (0x01B4–0x01BF)
eMIOS1 UC13 A Register	eMIOS1_UC13_A	32-bit	Base + 0x01C0
eMIOS1 UC13 B Register	eMIOS1_UC13_B	32-bit	Base + 0x01C4
Reserved	—	—	Base + (0x01C8–0x01CB)
eMIOS1 UC13 Control Register	eMIOS1_UC13_SC	32-bit	Base + 0x01CC
eMIOS1 UC13 Status Register	eMIOS1_UC13_SS	32-bit	Base + 0x01D0
Reserved	—	—	Base + (0x01D4–0x01DF)
eMIOS1 UC14 A Register	eMIOS1_UC14_A	32-bit	Base + 0x01E0
eMIOS1 UC14 B Register	eMIOS1_UC14_B	32-bit	Base + 0x01E4
Reserved	—	—	Base + (0x01E8–0x01EB)
eMIOS1 UC14 Control Register	eMIOS1_UC14_SC	32-bit	Base + 0x01EC
eMIOS1 UC14 Status Register	eMIOS1_UC14_SS	32-bit	Base + 0x01F0
Reserved	—	—	Base + (0x01F4–0x01FF)
eMIOS1 UC15 A Register	eMIOS1_UC15_A	32-bit	Base + 0x0200
eMIOS1 UC15 B Register	eMIOS1_UC15_B	32-bit	Base + 0x0204

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0208–0x020B)
eMIOS1 UC15 Control Register	eMIOS1_UC15_SC	32-bit	Base + 0x020C
eMIOS1 UC15 Status Register	eMIOS1_UC15_SS	32-bit	Base + 0x0210
Reserved	—	—	Base + (0x0214–0x021F)
eMIOS1 UC16 A Register	eMIOS1_UC16_A	32-bit	Base + 0x0220
eMIOS1 UC16 B Register	eMIOS1_UC16_B	32-bit	Base + 0x0224
eMIOS1 UC16 CNT	eMIOS1_UC16_CNT	32-bit	Base + 0x0228
eMIOS1 UC16 Control Register	eMIOS1_UC16_SC	32-bit	Base + 0x022C
eMIOS1 UC16 Status Register	eMIOS1_UC16_SS	32-bit	Base + 0x0230
Reserved	—	—	Base + (0x0234–0x023F)
eMIOS1 UC17 A Register	eMIOS1_UC17_A	32-bit	Base + 0x0240
eMIOS1 UC17 B Register	eMIOS1_UC17_B	32-bit	Base + 0x0244
Reserved	—	—	Base + (0x0248–0x024B)
eMIOS1 UC17 Control Register	eMIOS1_UC17_SC	32-bit	Base + 0x024C
eMIOS1 UC17 Status Register	eMIOS1_UC17_SS	32-bit	Base + 0x0250
Reserved	—	—	Base + (0x0254–0x025F)
eMIOS1 UC18 A Register	eMIOS1_UC18_A	32-bit	Base + 0x0260
eMIOS1 UC18 B Register	eMIOS1_UC18_B	32-bit	Base + 0x0264
Reserved	—	—	Base + (0x0268–0x026B)
eMIOS1 UC18 Control Register	eMIOS1_UC18_SC	32-bit	Base + 0x026C
eMIOS1 UC18 Status Register	eMIOS1_UC18_SS	32-bit	Base + 0x0270
Reserved	—	—	Base + (0x0274–0x027F)
eMIOS1 UC19 A Register	eMIOS1_UC19_A	32-bit	Base + 0x0280
eMIOS1 UC19 B Register	eMIOS1_UC19_B	32-bit	Base + 0x0284
Reserved	—	—	Base + (0x0288–0x028B)
eMIOS1 UC19 Control Register	eMIOS1_UC19_SC	32-bit	Base + 0x028C
eMIOS1 UC19 Status Register	eMIOS1_UC19_SS	32-bit	Base + 0x0290

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + 0x0294–0x029F)
eMIOS1 UC20 A Register	eMIOS1_UC20_A	32-bit	Base + 0x02A0
eMIOS1 UC20 B Register	eMIOS1_UC20_B	32-bit	Base + 0x02A4
Reserved	—	—	Base + (0x02A8–0x02AB)
eMIOS1 UC20 Control Register	eMIOS1_UC20_SC	32-bit	Base + 0x02AC
eMIOS1 UC20 Status Register	eMIOS1_UC20_SS	32-bit	Base + 0x02B0
Reserved	—	—	Base + (0x02B4–0x02BF)
eMIOS1 UC21 A Register	eMIOS1_UC21_A	32-bit	Base + 0x02C0
eMIOS1 UC21 B Register	eMIOS1_UC21_B	32-bit	Base + 0x02C4
Reserved	—	—	Base + (0x02C8–0x02CB)
eMIOS1 UC21 Control Register	eMIOS1_UC21_SC	32-bit	Base + 0x02CC
eMIOS1 UC21 Status Register	eMIOS1_UC21_SS	32-bit	Base + 0x02D0
Reserved	—	—	Base + (0x02D4–0x02DF)
eMIOS1 UC22 A Register	eMIOS1_UC22_A	32-bit	Base + 0x02E0
eMIOS1 UC22 B Register	eMIOS1_UC22_B	32-bit	Base + 0x02E4
Reserved	—	—	Base + (0x02E8–0x02EB)
eMIOS1 UC22 Control Register	eMIOS1_UC22_SC	32-bit	Base + 0x02EC
eMIOS1 UC22 Status Register	eMIOS1_UC22_SS	32-bit	Base + 0x02F0
Reserved	—	—	Base + (0x02F4–0x02FF)
eMIOS1 UC23 A Register	eMIOS1_UC23_A	32-bit	Base + 0x0300
eMIOS1 UC23 B Register	eMIOS1_UC23_B	32-bit	Base + 0x0304
eMIOS1 UC23 CNT	eMIOS1_UC23_CNT	32-bit	Base + 0x0308
eMIOS1 UC23 Control Register	eMIOS1_UC23_SC	32-bit	Base + 0x030C
eMIOS1 UC23 Status Register	eMIOS1_UC23_SS	32-bit	Base + 0x0310

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0314–0x031F)
eMIOS1 UC24 A Register	eMIOS1_UC24_A	32-bit	Base + 0x0320
eMIOS1 UC24 B Register	eMIOS1_UC24_B	32-bit	Base + 0x0324
eMIOS1 UC24 CNT	eMIOS1_UC24_CNT	32-bit	Base + 0x0328
eMIOS1 UC24 Control Register	eMIOS1_UC24_SC	32-bit	Base + 0x032C
eMIOS1 UC24 Status Register	eMIOS1_UC24_SS	32-bit	Base + 0x0330
Reserved	—	—	Base + (0x0334–0x033F)
eMIOS1 UC25 A Register	eMIOS1_UC25_A	32-bit	Base + 0x0340
Reserved	—	—	Base + (0x0344–0x034B)
eMIOS1 UC25 Control Register	eMIOS1_UC25_SC	32-bit	Base + 0x034C
eMIOS1 UC25 Status Register	eMIOS1_UC25_SS	32-bit	Base + 0x0350
Reserved	—	—	Base + (0x0354–0x035F)
eMIOS1 UC26 A Register	eMIOS1_UC26_A	32-bit	Base + 0x0360
Reserved	—	—	Base + (0x0364–0x036B)
eMIOS1 UC26 Control Register	eMIOS1_UC26_SC	32-bit	Base + 0x036C
eMIOS1 UC26 Status Register	eMIOS1_UC26_SS	32-bit	Base + 0x0370
Reserved	—	—	Base + (0x0374–0x037F)
eMIOS1 UC27 A Register	eMIOS1_UC27_A	32-bit	Base + 0x0380
Reserved	—	—	Base + (0x0384–0x038B)
eMIOS1 UC27 Control Register	eMIOS1_UC27_SC	32-bit	Base + 0x038C
eMIOS1 UC27 Status Register	eMIOS1_UC27_SS	32-bit	Base + 0x0390
Reserved	—	—	Base + (0x0394–0x07FF)
System Status and Configuration Module (SSCM) Section 38.2, Memory map and register description			**0xC3FD_8000**
System Status Register	STATUS	16-bit	Base + 0x0000
System Memory Configuration Register	MEMCONFIG	16-bit	Base + 0x0002
Reserved	—	—	Base + (0x0004–0x0005)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Error Configuration	ERROR	16-bit	Base + 0x0006
Debug Status Port Register	DEBUGPORT	16-bit	Base + 0x0008
Reserved	—	—	Base + (0x000A–0x000B)
Password Comparison Register High Word	PWCMPH	32-bit	Base + 0x000C
Password Comparison Register Low Word	PWCMPL	32-bit	Base + 0x0010
Reserved	—	—	Base + (0x0014–0x3FFF)
Mode Entry Module (MC_ME) Section 25.3, Memory map and register definition			**0xC3FD_C000**
Global Status	ME_GS	32-bit	Base + 0x0000
Mode Control	ME_MCTL	32-bit	Base + 0x0004
Mode Enable	ME_ME	32-bit	Base + 0x0008
Interrupt Status	ME_IS	32-bit	Base + 0x000C
Interrupt Mask	ME_IM	32-bit	Base + 0x0010
Invalid Mode Transition status	ME_IMTS	32-bit	Base + 0x0014
Reserved	—	—	Base + (0x0018–0x001F
Reset Mode Configuration	ME_RESET_MC	32-bit	Base + 0x0020
Test Mode Configuration	ME_TEST_MC	32-bit	Base + 0x0024
Safe Mode Configuration	ME_SAFE_MC	32-bit	Base + 0x0028
DRUN Mode Configuration	ME_DRUN_MC	32-bit	Base + 0x002C
Run0 Mode Configuration	ME_RUN0_MC	32-bit	Base + 0x0030
Run1 Mode Configuration	ME_RUN1_MC	32-bit	Base + 0x0034
Run2 Mode Configuration	ME_RUN2_MC	32-bit	Base + 0x0038
Run3 Mode Configuration	ME_RUN3_MC	32-bit	Base + 0x003C
Halt0 Mode Configuration	ME_HALT0_MC	32-bit	Base + 0x0040
Reserved	—	—	Base + (0x004–0x0047)
Stop0 Mode Configuration	ME_STOP0_MC	32-bit	Base + 0x0048
Reserved	—	—	Base + (0x004C–0x0053)
Standby0 Mode Configuration	ME_STANDBY0_MC	32-bit	Base + 0x0054
Reserved	—	—	Base + (0x0058–0x005F)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Peripheral Status Registers	ME_PS0	32-bit	Base + 0x0060
Peripheral Status Registers	ME_PS1	32-bit	Base + 0x0064
Peripheral Status Registers	ME_PS2	32-bit	Base + 0x0068
Peripheral Status Registers	ME_PS3	32-bit	Base + 0x006C
Reserved	—	—	Base + (0x0070–0x007F)
Run Peripheral Configuration Registers	ME_RUN_PC0	32-bit	Base + 0x0080
Run Peripheral Configuration Registers	ME_RUN_PC1	32-bit	Base + 0x0084
Run Peripheral Configuration Registers	ME_RUN_PC2	32-bit	Base + 0x0088
Run Peripheral Configuration Registers	ME_RUN_PC3	32-bit	Base + 0x008C
Run Peripheral Configuration Registers	ME_RUN_PC4	32-bit	Base + 0x0090
Run Peripheral Configuration Registers	ME_RUN_PC5	32-bit	Base + 0x0094
Run Peripheral Configuration Registers	ME_RUN_PC6	32-bit	Base + 0x0098
Run Peripheral Configuration Registers	ME_RUN_PC7	32-bit	Base + 0x009C
Low-Power Peripheral Configuration Registers	ME_LP_PC0	32-bit	Base + 0x00A0
Low-Power Peripheral Configuration Registers	ME_LP_PC1	32-bit	Base + 0x00A4
Low-Power Peripheral Configuration Registers	ME_LP_PC2	32-bit	Base + 0x00A8
Low-Power Peripheral Configuration Registers	ME_LP_PC3	32-bit	Base + 0x00AC
Low-Power Peripheral Configuration Registers	ME_LP_PC4	32-bit	Base + 0x00B0
Low-Power Peripheral Configuration Registers	ME_LP_PC5	32-bit	Base + 0x00B4
Low-Power Peripheral Configuration Registers	ME_LP_PC6	32-bit	Base + 0x00B8
Low-Power Peripheral Configuration Registers	ME_LP_PC7	32-bit	Base + 0x00BC
Reserved	—	—	Base + (0x00C0–0x00C3)
Peripheral Control Registers	ME_PCTL4	8-bit	Base + 0x00C4
Peripheral Control Registers	ME_PCTL5	8-bit	Base + 0x00C5
Peripheral Control Registers	ME_PCTL6	8-bit	Base + 0x00C6
Reserved	—	—	Base + (0x00C7–0x00CF)
Peripheral Control Registers	ME_PCTL16	8-bit	Base + 0x00D0
Peripheral Control Registers	ME_PCTL17	8-bit	Base + 0x00D1
Peripheral Control Registers	ME_PCTL18	8-bit	Base + 0x00D2
Peripheral Control Registers	ME_PCTL19	8-bit	Base + 0x00D3

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Peripheral Control Registers	ME_PCTL20	8-bit	Base + 0x00D4
Peripheral Control Registers	ME_PCTL21	8-bit	Base + 0x00D5
Reserved	—	—	Base + (0x00D6–0x00DF)
Peripheral Control Registers	ME_PCTL32	8-bit	Base + 0x00E0
Reserved	—	—	Base + (0x00E1–0x00EB)
Peripheral Control Registers	ME_PCTL44	8-bit	Base + 0x00EC
Reserved	—	—	Base + (0x00ED–0x00EF)
Peripheral Control Registers	ME_PCTL48	8-bit	Base + 0x00F0
Peripheral Control Registers	ME_PCTL49	8-bit	Base + 0x00F1
Peripheral Control Registers	ME_PCTL50	8-bit	Base + 0x00F2
Peripheral Control Registers	ME_PCTL51	8-bit	Base + 0x00F3
Reserved	—	—	Base + (0x00F4–0x00F8)
Peripheral Control Registers	ME_PCTL57	8-bit	Base + 0x00F9
Reserved	—	—	Base + (0x00FA–0x00FB)
Peripheral Control Registers	ME_PCTL60	8-bit	Base + 0x00FC
Reserved	—	—	Base + (0x00FD–0x0103)
Peripheral Control Registers	ME_PCTL68	8-bit	Base + 0x0104
Peripheral Control Registers	ME_PCTL69	8-bit	Base + 0x0105
Reserved	—	—	Base + (0x0106–0x0107)
Peripheral Control Registers	ME_PCTL72	8-bit	Base + 0x0108
Peripheral Control Registers	ME_PCTL73	8-bit	Base + 0x0109
Reserved	—	—	Base + 0x010A–0x011A)
Peripheral Control Registers	ME_PCTL91	8-bit	Base + 0x011B
Peripheral Control Registers	ME_PCTL92	8-bit	Base + 0x011C

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x011D–0x0127)
Peripheral Control Registers	ME_PCTL104	8-bit	Base + 0x0128
Reserved	—	—	Base + (0x0129–0x014F)
FXOSC Section 8.5.3, Register description			**0xC3FE_0000**
High Frequency Oscillator Control Register	OSC_CTL	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x003F)
SXOSC Section 8.6.4, Register description			**0xC3FE_0040**
Low Frequency Oscillator Control Register	OSC_CTL	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x005F)
RC Digital Interface Registers Section 8.8.3, Register description)			**0xC3FE_0060**
RC Digital Interface Registers	RC_CTL	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x007F)
LPRC Digital Interface Section 8.7.3, Register descriptionn			**0xC3FE_0080**
Low-Power RC Control Register	LPRC_CTL	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x009F)
PLLD0 Section 8.9.5, Register description			**0xC3FE_00A0**
Control Register	PLLD0_CR	32-bit	Base + 0x0000
PLLD Modulation Register	PLLD0_MR	32-bit	Base + 0x0004
Reserved	—	—	Base + (0x0008–0x00FF)
CMU0 Section 8.10.5, Memory map and register description			**0xC3FE_0100**
Control Status Register	CMU0_CSR	32-bit	Base + 0x0000
Frequency Display Register	CMU0_FDR	32-bit	Base + 0x0004
High Frequency Reference Register	CMU0_HFREFR_A	32-bit	Base + 0x0008
Low Frequency Reference Register	CMU0_LFREFR_A	32-bit	Base + 0x000C
Interrupt Status Register	CMU0_ISR	32-bit	Base + 0x0010
Interrupt Mask Register	CMU0_IMR	32-bit	Base + 0x0014
Measurement Duration Register	CMU0_MDR	32-bit	Base + 0x0018

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x001C–0x036F)
Clock Generation Module (MC_CGM) Section 8.4.3, Memory map and register definition			**0xC3FE_0370**
Output Clock Enable Register	CGM_OC_EN	32-bit	Base + 0x0000
Output Clock Division Select Register	CGM_OCDS_SC	32-bit	Base + 0x0004
System Clock Select Status Register	CGM_SC_SS	32-bit	Base + 0x0008
System Clock Divider Configuration Registers	CGM_SC_DC0	8-bit	Base + 0x000C
System Clock Divider Configuration Registers	CGM_SC_DC1	8-bit	Base + 0x000D
System Clock Divider Configuration Registers	CGM_SC_DC2	8-bit	Base + 0x000E
System Clock Divider Configuration Registers	CGM_SC_DC3	8-bit	Base + 0x000F
Reset Generation Module (MC_RGM) Section 31.3, Memory map and register definition			**0xC3FE_4000**
Functional Event Status	RGM_FES	16-bit	Base + 0x0000
Destructive Event Status	RGM_DES	16-bit	Base + 0x0002
Functional Event Reset Disable	RGM_FERD	16-bit	Base + 0x0004
Destructive Event Reset Disable	RGM_DERD	16-bit	Base + 0x0006
Reserved	—	—	Base + (0x0008–0x000F)
Functional Event Alternate Request	RGM_FEAR	16-bit	Base + 0x0010
Destructive Event Alternate Request	RGM_DEAR	16-bit	Base + 0x0012
Reserved	—	—	Base + (0x0014–0x0017)
Functional Event Short Sequence	RGM_FESS	16-bit	Base + 0x0018
Standby reset sequence	RGM_STDBY	16-bit	Base + 0x001A
Functional Bidirectional Reset Enable	RGM_FBRE	16-bit	Base + 0x001C
Reserved	—	—	Base + (0x001E–0x3FFF)
Power Control Unit (MC_PCU) Section 29.3, Memory map and register definition			**0xC3FE_8000**
Power domain #0 configuration register	PCONF0	32-bit	Base + 0x0000
Power domain #1 configuration register	PCONF1	32-bit	Base + 0x0004
Power domain #1 configuration register	PCONF2	32-bit	Base + 0x0008
Reserved	—	—	Base + (0x000C–0x003F)
Power Domain Status Register	PSTAT	32-bit	Base + 0x0040

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0044–0x007C)
Voltage Regulator Control Register	VCTL	32-bit	Base + 0x0080
Reserved	—	—	Base + (0x0084–0x3FFF)
Real Time Counter (RTC/API) Section 32.6, Register descriptions			**0xC3FE_C000**
RTC Supervisor Control Register	RTCSUPV	32-bit	Base + 0x0000
RTC Control Register	RTCC	32-bit	Base + 0x0004
RTC Status Register	RTCS	32-bit	Base + 0x0008
RTC Counter Register	RTCCNT	32-bit	Base + 0x000C
Reserved	—	—	Base + (0x0010–0x3FFF)
Periodic Interrupt Timer (PIT/RTI) Section 27.3, Memory map and register description			**0xC3FF_0000**
PIT Module Control Register	PITMCR	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x00FC)
Timer Load Value Register	LDVAL0	32-bit	Base + 0x0100
Current Timer Value Register 0	CVAL0	32-bit	Base + 0x0104
Timer Control Register 0	TCTRL0	32-bit	Base + 0x0108
Timer Flag Register 0	TFLG0	32-bit	Base + 0x010C
Timer Load Value Register 1	LDVAL1	32-bit	Base + 0x0110
Current Timer Value Register 1	CVAL1	32-bit	Base + 0x0114
Timer Control Register 1	TCTRL1	32-bit	Base + 0x0118
Timer Flag Register 1	TFLG1	32-bit	Base + 0x011C
Timer Load Value Register 2	LDVAL2	32-bit	Base + 0x0120
Current Timer Value Register 2	CVAL2	32-bit	Base + 0x0124
Timer Control Register 2	TCTRL2	32-bit	Base + 0x0128
Timer Flag Register 2	TFLG2	32-bit	Base + 0x012C
Timer Load Value Register 3	LDVAL3	32-bit	Base + 0x0130
Current Timer Value Register 3	CVAL3	32-bit	Base + 0x0134
Timer Control Register 3	TCTRL3	32-bit	Base + 0x0138
Timer Flag Register 3	TFLG3	32-bit	Base + 0x013C
Timer Load Value Register 4	LDVAL4	32-bit	Base + 0x0140
Current Timer Value Register 4	CVAL4	32-bit	Base + 0x0144

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Timer Control Register 4	TCTRL4	32-bit	Base + 0x0148
Timer Flag Register 4	TFLG4	32-bit	Base + 0x014C
Timer Load Value Register 5	LDVAL5	32-bit	Base + 0x0150
Current Timer Value Register 5	CVAL5	32-bit	Base + 0x0154
Timer Control Register 5	TCTRL5	32-bit	Base + 0x0158
Timer Flag Register 5	TFLG5	32-bit	Base + 0x015C
Reserved	—	—	Base + (0x0160–0x01FF
ADC 0 Section 5.4, Register descriptions			**0xFFE0_0000**
Main Configuration Register	MCR	32-bit	Base + 0x000
Main Status Register	MSR	32-bit	Base + 0x004
Reserved	—	—	Base + (0x008–0x00C)
Interrupt Status Register	ISR	32-bit	Base + 0x010
Reserved	—	—	Base + 0x014
Channel Pending Register	CEOCFR1	32-bit	Base + 0x018
Channel Pending Register	CEOCFR2	32-bit	Base + 0x01C
Interrupt Mask Register	IMR	32-bit	Base + 0x020
Reserved	—	—	Base + 0x024
Channel Interrupt Mask Register	CIMR1	32-bit	Base + 0x028
Channel Interrupt Mask Register	CIMR2	32-bit	Base + 0x02C
Watchdog Threshold Interrupt Status Register	WTISR	32-bit	Base + 0x030
Watchdog Threshold Interrupt Mask Register	WTIMR	32-bit	Base + 0x034
Reserved	—	—	Base + (0x038–0x03C
DMA Enable Register	DMAE	32-bit	Base + 0x040
Reserved	—	—	Base + 0x044
DMA Channel Select Register 1	DMAR1	32-bit	Base + 0x048
DMA Channel Select Register 2	DMAR2	32-bit	Base + 0x04C
Threshold Control Register 0	TRC0	32-bit	Base + 0x050
Threshold Control Register 1	TRC1	32-bit	Base + 0x054
Threshold Control Register 2	TRC2	32-bit	Base + 0x058
Threshold Control Register 3	TRC3	32-bit	Base + 0x05C

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Threshold Register 0	THRHLR0	32-bit	Base + 0x060
Threshold Register 1	THRHLR1	32-bit	Base + 0x064
Threshold Register 2	THRHLR2	32-bit	Base + 0x068
Threshold Register 3	THRHLR3	32-bit	Base + 0x06C
Reserved	—	—	Base + (0x070–0x094)
Conversion Timing Register 1	CTR1	32-bit	Base + 0x098
Conversion Timing Register 2	CTR2	32-bit	Base + 0x09C
Reserved	—	—	Base + (0x0A0–0x0A4)
Normal Conversion Mask Register 1	NCMR1	32-bit	Base + 0x0A8
Normal Conversion Mask Register 2	NCMR2	32-bit	Base + 0x0AC
Reserved	—	—	Base + (0x0B0–0x0B4)
Injected Conversion Mask Register 1	JCMR1	32-bit	Base + 0x0B8
Injected Conversion Mask Register 2	JCMR2	32-bit	Base + 0x0BC
Reserved	—	—	Base + 0x0C0
Decode Signals Delay Register	DSDR	32-bit	Base + 0x0C4
Power-down Exit Delay Register	PDEDR	32-bit	Base + 0x0C8
Reserved	—	—	Base + (0x0CC–0x17C)
Channel 32 Data Register	CDR32	32-bit	Base + 0x180
Channel 33 Data Register	CDR33	32-bit	Base + 0x184
Channel 34 Data Register	CDR34	32-bit	Base + 0x188
Channel 35 Data Register	CDR35	32-bit	Base + 0x18C
Channel 36 Data Register	CDR36	32-bit	Base + 0x190
Channel 37 Data Register	CDR37	32-bit	Base + 0x194
Channel 38 Data Register	CDR38	32-bit	Base + 0x198
Channel 39 Data Register	CDR39	32-bit	Base + 0x19C
Channel 40 Data Register	CDR40	32-bit	Base + 0x1A0
Channel 41 Data Register	CDR41	32-bit	Base + 0x1A4
Channel 43 Data Register	CDR43	32-bit	Base + 0x1AC
Channel 44 Data Register	CDR44	32-bit	Base + 0x1B0
Channel 45 Data RegisteR	CDR45	32-bit	Base + 0x1B4

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Channel 46 Data Register	CDR46	32-bit	Base + 0x1B8
Channel 47 Data Register	CDR47	32-bit	Base + 0x1BC
Channel 48 Data Register	CDR48	32-bit	Base + 0x1C0
Reserved	—	—	Base + (0x1C4–0x1FC)
Channel 64 Data Register	CDR64	32-bit	Base + 0x200
Channel 65 Data Register	CDR65	32-bit	Base + 0x204
Channel 66 Data Register	CDR66	32-bit	Base + 0x208
Channel 67 Data Register	CDR67	32-bit	Base + 0x20C
Channel 68 Data Register	CDR68	32-bit	Base + 0x210
Channel 69 Data Register	CDR69	32-bit	Base + 0x214
Channel 70 Data Register	CDR70	32-bit	Base + 0x218
Channel 71 Data Register	CDR71	32-bit	Base + 0x21C
Reserved	—	—	Base + (0x220–0x2FC)
I2C 0 Section 20.4, Memory map and register description			**0xFFE3_0000**
I2C Bus Address Register	IBAD	8-bit	Base + 0x0000
I2C Bus Frequency Divider Register	IBFD	8-bit	Base + 0x0001
I2C Bus Control Register	IBCR	8-bit	Base + 0x0002
I2C Bus Status Register	IBSR	8-bit	Base + 0x0003
I2C Bus Data I/O Register	IBDR	8-bit	Base + 0x0004
I2C Bus Interrupt Config Register	IBIC	8-bit	Base + 0x0005
Reserved	—	—	Base + (0x0006–0xFFFF)
I2C 1 Section 20.4, Memory map and register description			**0xFFE3_4000**
I2C Bus Address Register	IBAD	8-bit	Base + 0x0000
I2C Bus Frequency Divider Register	IBFD	8-bit	Base + 0x0001
I2C Bus Control Register	IBCR	8-bit	Base + 0x0002
I2C Bus Status Register	IBSR	8-bit	Base + 0x0003
I2C Bus Data I/O Register	IBDR	8-bit	Base + 0x0004
I2C Bus Interrupt Config Register	IBIC	8-bit	Base + 0x0005
Reserved	—	—	Base + (0x0006–0xFFFF)
I2C 2 Section 20.4, Memory map and register description			**0xFFE3_8000**

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
I2C Bus Address Register	IBAD	8-bit	Base + 0x0000
I2C Bus Frequency Divider Register	IBFD	8-bit	Base + 0x0001
I2C Bus Control Register	IBCR	8-bit	Base + 0x0002
I2C Bus Status Register	IBSR	8-bit	Base + 0x0003
I2C Bus Data I/O Register	IBDR	8-bit	Base + 0x0004
I2C Bus Interrupt Config Register	IBIC	8-bit	Base + 0x0005
Reserved	—	—	Base + (0x0006–0xFFFF)
I2C 3 Section 20.4, Memory map and register description			**0xFFE3_C000**
I2C Bus Address Register	IBAD	8-bit	Base + 0x0000
I2C Bus Frequency Divider Register	IBFD	8-bit	Base + 0x0001
I2C Bus Control Register	IBCR	8-bit	Base + 0x0002
I2C Bus Status Register	IBSR	8-bit	Base + 0x0003
I2C Bus Data I/O Register	IBDR	8-bit	Base + 0x0004
I2C Bus Interrupt Config Register	IBIC	8-bit	Base + 0x0005
Reserved	—	—	Base + (0x0006–0xFFFF)
LINFlex 0 Section 23.7, Memory map and registers description			**0xFFE4_0000**
LIN Control Register	LINCR1	16-bit	Base + 0x0000
Reserved	—	—	Base + (0x0002–0x0003)
LIN Interrupt Enable Register	LINIER	16-bit	Base + 0x0004
Reserved	—	—	Base + (0x0006–0x0007)
LIN Status Register	LINSR	16-bit	Base + 0x0008
Reserved	—	—	Base + (0x000A–0x000B)
LIN Error Status Register	LINESR	16-bit	Base + 0x000C
Reserved	—	—	Base + (0x000E–0x000F)
UART Mode Control Register	UARTCR	16-bit	Base + 0x0010
Reserved	—	—	Base + (0x0012–0x0013)
UART Mode Status Register	UARTSR	16-bit	Base + 0x0014

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0016–0x0017)
LIN Timeout Control Status Register	LINTCSR	16-bit	Base + 0x0018
Reserved	—	—	Base + (0x001A–0x001B)
LIN Output Compare Register	LINOCR	16-bit	Base + 0x001C
Reserved	—	—	Base + (0x001E–0x001F)
LIN Timeout Control Register	LINTOCR	16-bit	Base + 0x0020
Reserved	—	—	Base + (0x0022–0x0023)
LIN Fractional Baud Rate Register	LINFBRR	16-bit	Base + 0x0024
Reserved	—	—	Base + (0x0026–0x0027)
LIN Integer Baud Rate Register	LINIBRR	16-bit	Base + 0x0028
Reserved	—	—	Base + (0x002A–0x002B)
LIN Checksum Field Register	LINCFR	16-bit	Base + 0x002C
Reserved	—	—	Base + (0x002E–0x002F)
LIN Control Register 2	LINCR2	16-bit	Base + 0x0030
Reserved	—	—	Base + (0x0032–0x0033)
Buffer Identifier Register	BIDR	16-bit	Base + 0x0034
Reserved	—	—	Base + (0x0036–0x0037)
Buffer Data Register Least Significant	BDRL	16-bit	Base + 0x0038
Reserved	—	—	Base + (0x003A–0x003B)
Buffer Data Register Most Significant	BDRM	16-bit	Base + 0x003C
Reserved	—	—	Base + (0x003E–0x003F)
Identifier Filter Enable Register	IFER	16-bit	Base + 0x0040
Reserved	—	—	Base + (0x0042–0x0043)
Identifier Filter Match Index	IFMI	16-bit	Base + 0x0044

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0046–0x0047)
Identifier Filter Mode Register	IFMR	16-bit	Base + 0x0048
Reserved	—	—	Base + (0x004A–0x004B)
Identifier Filter Control Register	IFCR0	16-bit	Base + 0x004C
Reserved	—	—	Base + (0x004E–0x004F)
Identifier Filter Control Register	IFCR1	16-bit	Base + 0x0050
Reserved	—	—	Base + (0x0052–0x0053)
Identifier Filter Control Register	IFCR2	16-bit	Base + 0x0054
Reserved	—	—	Base + (0x0056–0x0057)
Identifier Filter Control Register	IFCR3	16-bit	Base + 0x0058
Reserved	—	—	Base + (0x005A–0x005B)
Identifier Filter Control Register	IFCR4	16-bit	Base + 0x005C
Reserved	—	—	Base + (0x005E–0x005F)
Identifier Filter Control Register	IFCR5	16-bit	Base + 0x0060
Reserved	—	—	Base + (0x0062–0x0063)
Identifier Filter Control Register	IFCR6	16-bit	Base + 0x0064
Reserved	—	—	Base + (0x0066–0x0067)
Identifier Filter Control Register	IFCR7	16-bit	Base + 0x0068
Reserved	—	—	Base + (0x006A–0x006B)
Identifier Filter Control Register	IFCR8	16-bit	Base + 0x006C
Reserved	—	—	Base + (0x006E–0x006F)
Identifier Filter Control Register	IFCR9	16-bit	Base + 0x0070
Reserved	—	—	Base + (0x0072–0x0073)
Identifier Filter Control Register	IFCR10	16-bit	Base + 0x0074

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0076–0x0077)
Identifier Filter Control Register	IFCR11	16-bit	Base + 0x0078
Reserved	—	—	Base + (0x007A–0x007B)
Identifier Filter Control Register	IFCR12	16-bit	Base + 0x007C
Reserved	—	—	Base + (0x007E–0x007F)
Identifier Filter Control Register	IFCR13	16-bit	Base + 0x0080
Reserved	—	—	Base + (0x0082–0x0083)
Identifier Filter Control Register	IFCR14	16-bit	Base + 0x0084
Reserved	—	—	Base + (0x0086–0x0087)
Identifier Filter Control Register	IFCR15	16-bit	Base + 0x0088
Reserved	—	—	Base + (0x009A–0x3FFF)
LINFlex 1 Section 23.7, Memory map and registers description			**0xFFE4_4000**
LIN Control Register	LINCR1	16-bit	Base + 0x0000
Reserved	—	—	Base + (0x0002–0x0003)
LIN Interrupt Enable Register	LINIER	16-bit	Base + 0x0004
Reserved	—	—	Base + (0x0006–0x0007)
LIN Status Register	LINSR	16-bit	Base + 0x0008
Reserved	—	—	Base + (0x000A–0x000B)
LIN Error Status Register	LINESR	16-bit	Base + 0x000C
Reserved	—	—	Base + (0x000E–0x000F)
UART Mode Control Register	UARTCR	16-bit	Base + 0x0010
Reserved	—	—	Base + (0x0012–0x0013)
UART Mode Status Register	UARTSR	16-bit	Base + 0x0014
Reserved	—	—	Base + (0x0016–0x0017)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
LIN Timeout Control Status Register	LINTCSR	16-bit	Base + 0x0018
Reserved	—	—	Base + (0x001A–0x001B)
LIN Output Compare Register	LINOCR	16-bit	Base + 0x001C
Reserved	—	—	Base + (0x001E–0x001F)
LIN Timeout Control Register	LINTOCR	16-bit	Base + 0x0020
Reserved	—	—	Base + (0x0022–0x0023)
LIN Fractional Baud Rate Register	LINFBRR	16-bit	Base + 0x0024
Reserved	—	—	Base + (0x0026–0x0027)
LIN Integer Baud Rate Register	LINIBRR	16-bit	Base + 0x0028
Reserved	—	—	Base + (0x002A–0x002B)
LIN Checksum Field Register	LINCFR	16-bit	Base + 0x002C
Reserved	—	—	Base + (0x002E–0x002F)
LIN Control Register 2	LINCR2	16-bit	Base + 0x0030
Reserved	—	—	Base + (0x0032–0x0033)
Buffer Identifier Register	BIDR	16-bit	Base + 0x0034
Reserved	—	—	Base + (0x0036–0x0037)
Buffer Data Register Least Significant	BDRL	16-bit	Base + 0x0038
Reserved	—	—	Base + (0x003A–0x003B)
Buffer Data Register Most Significant	BDRM	16-bit	Base + 0x003C
Reserved	—	—	Base + (0x003E–0x3FFF)
Reserved	—	—	FFE6_4000–FFE6_FFFF
CAN Sampler Section 7.3, Register description			**0xFFE7_0000**
Control Status Register	CANS_CR	32-bit	Base + 0x0000
Sample Register 0	CAN_SR0	32-bit	Base + 0x0004
Sample Register 1	CAN_SR1	32-bit	Base + 0x0008

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Sample Register 2	CAN_SR2	32-bit	Base + 0x000C
Sample Register 3	CAN_SR3	32-bit	Base + 0x0010
Sample Register 4	CAN_SR4	32-bit	Base + 0x0014
Sample Register 5	CAN_SR5	32-bit	Base + 0x0018
Sample Register 6	CAN_SR6	32-bit	Base + 0x001C
Sample Register 7	CAN_SR7	32-bit	Base + 0x0020
Sample Register 8	CAN_SR8	32-bit	Base + 0x0024
Sample Register 9	CAN_SR9	32-bit	Base + 0x0028
Sample Register 10	CAN_SR10	32-bit	Base + 0x002C
Sample Register 11	CAN_SR11	32-bit	Base + 0x0030
Reserved	—	—	Base + 0x0034 –FFF0_FFFF
MPU Section 24.2, Memory map and register description			**0xFFF1_0000**
MPU Control/Error Status Register	MPU_CESR	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x000F)
MPU Error Address Register, Slave Port 0	MPU_EAR0	32-bit	Base + 0x0010
MPU Error Detail Register, Slave Port 0	MPU_EDR0	32-bit	Base + 0x0014
MPU Error Address Register, Slave Port 1	MPU_EAR1	32-bit	Base + 0x0018
MPU Error Detail Register, Slave Port 1	MPU_EDR1	32-bit	Base + 0x001C
MPU Error Address Register, Slave Port 2	MPU_EAR2	32-bit	Base + 0x0020
MPU Error Detail Register, Slave Port 2	MPU_EDR2	32-bit	Base + 0x0024
MPU Error Address Register, Slave Port 3	MPU_EAR3	32-bit	Base + 0x0028
MPU Error Detail Register, Slave Port 3	MPU_EDR3	32-bit	Base + 0x002C
Reserved for MPU Extended Error Detail Register, Slave Port 0	MPU_XEDR0	64-bit	Base + 0x0030
Reserved for MPU Extended Error Detail Register, Slave Port 1	MPU_XEDR1	64-bit	Base + 0x0038
Reserved for MPU Extended Error Detail Register, Slave Port 2	MPU_XEDR2	64-bit	Base + 0x0040
Reserved for MPU Extended Error Detail Register, Slave Port 3	MPU_XEDR3	64-bit	Base + 0x0048
Reserved		—	Base + (0x0050–0x03FF)
MPU Region Descriptor 0	MPU_RGD0	128-bit	Base + 0x0400

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
MPU Region Descriptor 1	MPU_RGD1	128-bit	Base + 0x0410
MPU Region Descriptor 2	MPU_RGD2	128-bit	Base + 0x0420
MPU Region Descriptor 3	MPU_RGD3	128-bit	Base + 0x0430
MPU Region Descriptor 4	MPU_RGD4	128-bit	Base + 0x0440
MPU Region Descriptor 5	MPU_RGD5	128-bit	Base + 0x0450
MPU Region Descriptor 6	MPU_RGD6	128-bit	Base + 0x0460
MPU Region Descriptor 7	MPU_RGD7	128-bit	Base + 0x0470
MPU Region Descriptor 8	MPU_RGD8	128-bit	Base + 0x0480
MPU Region Descriptor 9	MPU_RGD9	128-bit	Base + 0x0490
MPU Region Descriptor 10	MPU_RGD10	128-bit	Base + 0x04A0
MPU Region Descriptor 11	MPU_RGD11	128-bit	Base + 0x04B0
MPU Region Descriptor 12	MPU_RGD12	128-bit	Base + 0x04C0
MPU Region Descriptor 13	MPU_RGD13	128-bit	Base + 0x04D0
MPU Region Descriptor 14	MPU_RGD14	128-bit	Base + 0x04E0
MPU Region Descriptor 15	MPU_RGD15	128-bit	Base + 0x04F0
Reserved	—	—	Base + (0x0500–0x07FF)
MPU RGD Alternate Access Control 0	MPU_RGDAAC0	32-bit	Base + 0x0800
MPU RGD Alternate Access Control 1	MPU_RGDAAC1	32-bit	Base + 0x0804
MPU RGD Alternate Access Control 2	MPU_RGDAAC2	32-bit	Base + 0x0808
MPU RGD Alternate Access Control 3	MPU_RGDAAC3	32-bit	Base + 0x080c
MPU RGD Alternate Access Control 4	MPU_RGDAAC4	32-bit	Base + 0x0810
MPU RGD Alternate Access Control 5	MPU_RGDAAC5	32-bit	Base + 0x0814
MPU RGD Alternate Access Control 6	MPU_RGDAAC6	32-bit	Base + 0x0818
MPU RGD Alternate Access Control 7	MPU_RGDAAC7	32-bit	Base + 0x081C
MPU RGD Alternate Access Control 8	MPU_RGDAAC8	32-bit	Base + 0x0820
MPU RGD Alternate Access Control 9	MPU_RGDAAC9	32-bit	Base + 0x0824
MPU RGD Alternate Access Control 10	MPU_RGDAAC10	32-bit	Base + 0x0828
MPU RGD Alternate Access Control 11	MPU_RGDAAC11	32-bit	Base + 0x082C
MPU RGD Alternate Access Control 12	MPU_RGDAAC12	32-bit	Base + 0x0830
MPU RGD Alternate Access Control 13	MPU_RGDAAC13	32-bit	Base + 0x0834
MPU RGD Alternate Access Control 14	MPU_RGDAAC14	32-bit	Base + 0x0838

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
MPU RGD Alternate Access Control 15	MPU_RGDAAC15	32-bit	Base + 0x083C
Reserved	—	—	Base + (0x0840–0x3FFF)
SWT 0 Section 4.2.5, Memory map and register description			**0xFFF3_8000**
Control Register	SWT_CR	32-bit	Base + 0x0000
SWT Interrupt Register	SWT_IR	32-bit	Base + 0x0004
SWT Timeout Register	SWT_TO	32-bit	Base + 0x0008
SWT Window Register	SWT_WN	32-bit	Base + 0x000C
SWT Service Register	SWT_SR	32-bit	Base + 0x0010
SWT Counter Output Register	SWT_CO	32-bit	Base + 0x0014
Reserved	—	—	Base + 0x0018 –0xFFF3_BFFF
STM Section 39.3, Memory map and register definition			**0xFFF3_C000**
Control Register	STM_CR	32-bit	Base + 0x0000
STM Count Register	STM_CNT	32-bit	Base + 0x0004
Reserved	—	—	Base + (0x0008–0x000F)
STM Channel 0 Control Register	STM_CCR0	32-bit	Base + 0x0010
STM Channel 0 Interrupt Register	STM_CIR0	32-bit	Base + 0x0014
STM Channel 0 Compare Register	STM_CMP0	32-bit	Base + 0x0018
Reserved	—	—	Base + (0x001C–0x001F)
STM Channel 1 Control Register	STM_CCR1	32-bit	Base + 0x0020
STM Channel 1 Interrupt Register	STM_CIR1	32-bit	Base + 0x0024
STM Channel 1 Compare Register	STM_CMP1	32-bit	Base + 0x0028
Reserved	—	—	Base + (0x002C–0x002F)
STM Channel 2 Control Register	STM_CCR2	32-bit	Base + 0x0030
STM Channel 2 Interrupt Register	STM_CIR2	32-bit	Base + 0x0034
STM Channel 2 Compare Register	STM_CMP2	32-bit	Base + 0x0038
Reserved	—	—	Base + (0x003C–0x003F)
STM Channel 3 Control Register	STM_CCR3	32-bit	Base + 0x0040
STM Channel 3 Interrupt Register	STM_CIR3	32-bit	Base + 0x0044

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
STM Channel 3 Compare Register	STM_CMP3	32-bit	Base + 0x0048
Reserved	—	—	Base + (0x003C–0x3FF)
ECSM Section 16.4, Memory map and register description			**0xFFF4_0000**
Processor Core Type	ECSM_PCT	16-bit	Base + 0x0000
SOC-Defined Platform Revision	ECSM_PLREV	16-bit	Base + 0x0002
Reserved	—	—	Base + (0x0004–0x0007)
IPS On-Platform Module Configuration	ECSM_IOPMC	32-bit	Base + 0x0008
Reserved	—	—	Base + (0x000C–0x000E)
Miscellaneous Reset Status Register	ECSM_MRSR	8-bit	Base + 0x000F
Reserved	—	—	Base + (0x00010–0x001)
Miscellaneous Wakeup Control Register	ECSM_MWCR	8-bit	Base + 0x0013
Reserved	—	—	Base + (0x0014–0x001E)
Miscellaneous Interrupt Register	ECSM_MIR	8-bit	Base + 0x001F
Reserved	—	—	Base + (0x0020–0x0023)
Miscellaneous User Defined Control Register	ECSM_MUDCR	32-bit	Base + 0x0024
Reserved	—	—	Base + (0x0028–0x0042)
ECC Configuration Register	ECSM_ECR	8-bit	Base + 0x0043
Reserved	—	—	Base + (0x0044–0x0046)
ECC Status Register	ECSM_ESR	8-Base bit	Base + 0x0047
Reserved	—	—	Base + (0x0048–0x0049)
ECC Error Generation Register	ECSM_EEGR	16-bit	Base + 0x004A
Reserved	—	—	Base + (0x004C–0x004F)
Platform Flash ECC Error Address Register	ECSM_PFEAR	32-bit	Base + 0x0050
Reserved	—	—	Base + (0x054–0x0055)

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Platform Flash ECC Master Number Register	ECSM_PFEMR	8-bit	Base + 0x0056
Platform Flash ECC Attributes Register	ECSM_PFEAT	8-bit	Base + 0x0057
Reserved	—	—	Base + (0x058–0x005B)
Platform Flash ECC Data Register	ECSM_PFEDR	32-bit	Base + 0x005C
Platform RAM ECC Address Register	ECSM_PREAR	32-bit	Base + 0x0060
Reserved	—	—	Base + 0x064
Platform RAM ECC Syndrome Register	ECSM_PRESR	8-bit	Base + 0x0065
Platform RAM ECC Master Number Register	ECSM_PREMR	8-bit	Base + 0x0066
Platform RAM ECC Attributes Register	ECSM_PREAT	8-bit	Base + 0x0067
Reserved	—	—	Base + (0x068–0x006B)
Platform RAM ECC Data Register	ECSM_PREDR	32-bit	Base + 0x006C
Reserved	—	—	Base + (0x0070–0x3FFF)
INTC Section 21.5, Memory map and register description			**0xFFF4_8000**
Block Configuration Register	INTC_PBCR	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x0007)
Current Priority Register	INTC_CPR	32-bit	Base + 0x0008
Reserved	—	—	Base + (0x000C–0x000F)
Interrupt Acknowledge Register	INTC_IACKR	32-bit	Base + 0x0010
Reserved	—	—	Base + (0x0014–0x0017)
End of Interrupt Register	INTC_EOIR	32-bit	Base + 0x0018
Reserved	—	—	Base + (0x001C–0x001F)
Software Set/Clear Interrupt Register	INTC_SSCIR0_3	32-bit	Base + 0x0020
Software Set/Clear Interrupt Register	INTC_SSCIR4_7	32-bit	Base + 0x0024
Reserved	—	—	Base + (0x0028–0x003F)
Priority Select Register	INTC_PSR0_3	32-bit	Base + 0x0040
Priority Select Register	INTC_PSR4_7	32-bit	Base + 0x0044
Priority Select Register	INTC_PSR8_11	32-bit	Base + 0x0048

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Priority Select Register	INTC_PSR12_15	32-bit	Base + 0x004C
Priority Select Register	INTC_PSR16_19	32-bit	Base + 0x0050
Priority Select Register	INTC_PSR20_23	32-bit	Base + 0x0054
Priority Select Register	INTC_PSR24_27	32-bit	Base + 0x0058
Priority Select Register	INTC_PSR28_31	32-bit	Base + 0x005C
Priority Select Register	INTC_PSR32_35	32-bit	Base + 0x0060
Priority Select Register	INTC_PSR36_39	32-bit	Base + 0x0064
Priority Select Register	INTC_PSR40_43	32-bit	Base + 0x0068
Priority Select Register	INTC_PSR44_47	32-bit	Base + 0x006C
Priority Select Register	INTC_PSR48_51	32-bit	Base + 0x0070
Priority Select Register	INTC_PSR52_55	32-bit	Base + 0x0074
Priority Select Register	INTC_PSR56_59	32-bit	Base + 0x0078
Priority Select Register	INTC_PSR60_63	32-bit	Base + 0x007C
Priority Select Register	INTC_PSR64_67	32-bit	Base + 0x0080
Priority Select Register	INTC_PSR68_71	32-bit	Base + 0x0084
Priority Select Register	INTC_PSR72_75	32-bit	Base + 0x0088
Priority Select Register	INTC_PSR76_79	32-bit	Base + 0x008C
Priority Select Register	INTC_PSR80_83	32-bit	Base + 0x0090
Priority Select Register	INTC_PSR84_87	32-bit	Base + 0x0094
Priority Select Register	INTC_PSR88_91	32-bit	Base + 0x0098
Priority Select Register	INTC_PSR92_95	32-bit	Base + 0x009C
Priority Select Register	INTC_PSR96_99	32-bit	Base + 0x00A0
Priority Select Register	INTC_PSR100_103	32-bit	Base + 0x00A4
Priority Select Register	INTC_PSR104_107	32-bit	Base + 0x00A8
Priority Select Register	INTC_PSR108_111	32-bit	Base + 0x00AC
Priority Select Register	INTC_PSR112_115	32-bit	Base + 0x00B0
Priority Select Register	INTC_PSR116_119	32-bit	Base + 0x00B4
Priority Select Register	INTC_PSR120_123	32-bit	Base + 0x00B8
Priority Select Register	INTC_PSR124_127	32-bit	Base + 0x00BC
Priority Select Register	INTC_PSR128_131	32-bit	Base + 0x00C0
Priority Select Register	INTC_PSR132_135	32-bit	Base + 0x00C4
Priority Select Register	INTC_PSR136_139	32-bit	Base + 0x00C8

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Priority Select Register	INTC_PSR140_143	32-bit	Base + 0x00CC
Priority Select Register	INTC_PSR144_147	32-bit	Base + 0x00D0
Priority Select Register	INTC_PSR148_151	32-bit	Base + 0x00D4
Priority Select Register	INTC_PSR152_155	32-bit	Base + 0x00D8
Priority Select Register	INTC_PSR156_159	32-bit	Base + 0x00DC
Priority Select Register	INTC_PSR160_163	32-bit	Base + 0x00E0
Priority Select Register	INTC_PSR164_167	32-bit	Base + 0x00E4
Priority Select Register	INTC_PSR168_171	32-bit	Base + 0x00E8
Priority Select Register	INTC_PSR172_175	32-bit	Base + 0x00EC
Priority Select Register	INTC_PSR176_179	32-bit	Base + 0x00F0
Priority Select Register	INTC_PSR180_183	32-bit	Base + 0x00F4
Priority Select Register	INTC_PSR184_187	32-bit	Base + 0x00F8
Priority Select Register	INTC_PSR188_191	32-bit	Base + 0x00FC
Priority Select Register	INTC_PSR192_195	32-bit	Base + 0x0100
Priority Select Register	INTC_PSR196_199	32-bit	Base + 0x0104
Priority Select Register	INTC_PSR200_203	32-bit	Base + 0x0108
Priority Select Register	INTC_PSR204_207	32-bit	Base + 0x010C
Priority Select Register	INTC_PSR208_211	32-bit	Base + 0x0110
Priority Select Register	INTC_PSR212_215	32-bit	Base + 0x0114
Priority Select Register	INTC_PSR216_219	32-bit	Base + 0x0118
Priority Select Register	INTC_PSR220_223	32-bit	Base + 0x011C
Priority Select Register	INTC_PSR224_227	32-bit	Base + 0x0120
Priority Select Register	INTC_PSR228_231	32-bit	Base + 0x0124
Priority Select Register	INTC_PSR232_235	32-bit	Base + 0x0128
Priority Select Register	INTC_PSR236_239	32-bit	Base + 0x012C
Priority Select Register	INTC_PSR240_243	32-bit	Base + 0x0130
Priority Select Register	INTC_PSR244_247	32-bit	Base + 0x0134
Priority Select Register	INTC_PSR248_251	32-bit	Base + 0x0138
Priority Select Register	INTC_PSR252_255	32-bit	Base + 0x013C
Priority Select Register	INTC_PSR256_259	32-bit	Base + 0x0140
Priority Select Register	INTC_PSR260_263	32-bit	Base + 0x0144
Priority Select Register	INTC_PSR264_267	32-bit	Base + 0x0148

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Priority Select Register	INTC_PSR268_271	32-bit	Base + 0x014C
Priority Select Register	INTC_PSR272_275	32-bit	Base + 0x0150
Priority Select Register	INTC_PSR276_279	32-bit	Base + 0x0154
Priority Select Register	INTC_PSR280_283	32-bit	Base + 0x0158
Priority Select Register	INTC_PSR284_287	32-bit	Base + 0x015C
Priority Select Register	INTC_PSR288_291	32-bit	Base + 0x0160
Priority Select Register	INTC_PSR292_295	32-bit	Base + 0x0164
Priority Select Register	INTC_PSR296_299	32-bit	Base + 0x0168
Priority Select Register	INTC_PSR300_303	32-bit	Base + 0x016C
Priority Select Register	INTC_PSR304_307	32-bit	Base + 0x0170
Priority Select Register	INTC_PSR308_311	32-bit	Base + 0x0174
Priority Select Register	INTC_PSR312_315	32-bit	Base + 0x0178
Priority Select Register	INTC_PSR316_319	32-bit	Base + 0x017C
Priority Select Register	INTC_PSR320_323	32-bit	Base + 0x0180
Priority Select Register	INTC_PSR324_327	32-bit	Base + 0x0184
Priority Select Register	INTC_PSR328_331	32-bit	Base + 0x0188
Priority Select Register	INTC_PSR332_335	32-bit	Base + 0x018C
Priority Select Register	INTC_PSR336_339	32-bit	Base + 0x0190
Priority Select Register	INTC_PSR340_343	32-bit	Base + 0x0194
Priority Select Register	INTC_PSR344_347	32-bit	Base + 0x0198
Priority Select Register	INTC_PSR348_351	32-bit	Base + 0x019C
Priority Select Register	INTC_PSR352_355	32-bit	Base + 0x01A0
Priority Select Register	INTC_PSR356_359	32-bit	Base + 0x01A4
Priority Select Register	INTC_PSR360_363	32-bit	Base + 0x01A8
Priority Select Register	INTC_PSR364_367	32-bit	Base + 0x01AC
Priority Select Register	INTC_PSR368_371	32-bit	Base + 0x01B0
Priority Select Register	INTC_PSR372_375	32-bit	Base + 0x01B4
Priority Select Register	INTC_PSR376_379	32-bit	Base + 0x01B8
Priority Select Register	INTC_PSR380_383	32-bit	Base + 0x01BC
Priority Select Register	INTC_PSR384_387	32-bit	Base + 0x01C0
Priority Select Register	INTC_PSR388_391	32-bit	Base + 0x01C4
Priority Select Register	INTC_PSR392_395	32-bit	Base + 0x01C8

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Priority Select Register	INTC_PSR396_399	32-bit	Base + 0x01CC
Priority Select Register	INTC_PSR400_403	32-bit	Base + 0x01D0
Priority Select Register	INTC_PSR404_407	32-bit	Base + 0x01D4
Priority Select Register	INTC_PSR408_411	32-bit	Base + 0x01D8
Priority Select Register	INTC_PSR412_415	32-bit	Base + 0x01DC
Priority Select Register	INTC_PSR416_419	32-bit	Base + 0x01E0
Priority Select Register	INTC_PSR420_423	32-bit	Base + 0x01E4
Priority Select Register	INTC_PSR424_427	32-bit	Base + 0x01E8
Priority Select Register	INTC_PSR428_431	32-bit	Base + 0x01EC
Priority Select Register	INTC_PSR432_435	32-bit	Base + 0x01F0
Priority Select Register	INTC_PSR436_439	32-bit	Base + 0x01F4
Priority Select Register	INTC_PSR440_443	32-bit	Base + 0x01F8
Priority Select Register	INTC_PSR444_447	32-bit	Base + 0x01FC
Priority Select Register	INTC_PSR448_451	32-bit	Base + 0x0200
Priority Select Register	INTC_PSR452_455	32-bit	Base + 0x0204
Priority Select Register	INTC_PSR456_459	32-bit	Base + 0x0208
Priority Select Register	INTC_PSR460_463	32-bit	Base + 0x020C
Priority Select Register	INTC_PSR464_467	32-bit	Base + 0x0210
Priority Select Register	INTC_PSR468_471	32-bit	Base + 0x0214
Priority Select Register	INTC_PSR472_475	32-bit	Base + 0x0218
Priority Select Register	INTC_PSR476_479	32-bit	Base + 0x021C
Priority Select Register	INTC_PSR480_483	32-bit	Base + 0x0220
Priority Select Register	INTC_PSR484_487	32-bit	Base + 0x0224
Priority Select Register	INTC_PSR488_491	32-bit	Base + 0x0228
Priority Select Register	INTC_PSR492_495	32-bit	Base + 0x022C
Priority Select Register	INTC_PSR496_499	32-bit	Base + 0x0230
Priority Select Register	INTC_PSR500_503	32-bit	Base + 0x0234
Priority Select Register	INTC_PSR504_507	32-bit	Base + 0x0238
Priority Select Register	INTC_PSR508_511	32-bit	Base + 0x023C
DSPI 0 Section 11.7, Memory map and register description		**0xFFF9_0000**	
Module Configuration Register	PMCR	32-bit	Base + 0x0000

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Reserved	—	—	Base + (0x0004–0x0007)
Transfer Count Register	TCR	32-bit	Base + 0x0008
Clock and Transfer Attribute Registers	CTAR0	32-bit	Base + 0x000C
Clock and Transfer Attribute Registers	CTAR1	32-bit	Base + 0x0010
Clock and Transfer Attribute Registers	CTAR2	32-bit	Base + 0x0014
Clock and Transfer Attribute Registers	CTAR3	32-bit	Base + 0x0018
Clock and Transfer Attribute Registers	CTAR4	32-bit	Base + 0x001C
Clock and Transfer Attribute Registers	CTAR5	32-bit	Base + 0x0020
Clock and Transfer Attribute Registers	CTAR6	32-bit	Base + 0x0024
Clock and Transfer Attribute Registers	CTAR7	32-bit	Base + 0x0028
Status Register	SR	32-bit	Base + 0x002C
DMA/Interrupt Request Register	RSER	32-bit	Base + 0x0030
PUSH TX FIFO Register	PUSHR	32-bit	Base + 0x0034
POP RX FIFO Register	POPR	32-bit	Base + 0x0038
DSPI Transmit FIFO Registers	TXFR0	32-bit	Base + 0x003C
DSPI Transmit FIFO Registers	TXFR1	32-bit	Base + 0x0040
DSPI Transmit FIFO Registers	TXFR2	32-bit	Base + 0x0044
DSPI Transmit FIFO Registers	TXFR3	32-bit	Base + 0x0048
Reserved	—	—	Base + (0x004C–0x007B)
Receive FIFO Registers	RXFR0	32-bit	Base + 0x007C
Receive FIFO Registers	RXFR1	32-bit	Base + 0x0080
Receive FIFO Registers	RXFR2	32-bit	Base + 0x0084
Receive FIFO Registers	RXFR3	32-bit	Base + 0x0088
Reserved	—	—	Base + (0x008C–0x3FF)
DSPI 1 Section 11.7, Memory map and register description		**0xFFF9_4000**	
Module Configuration Register	PMCR	32-bit	Base + 0x0000
Reserved	—	—	Base + (0x0004–0x0007)
Transfer Count Register	TCR	32-bit	Base + 0x0008
Clock and Transfer Attribute Registers	CTAR0	32-bit	Base + 0x000C

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Clock and Transfer Attribute Registers	CTAR1	32-bit	Base + 0x0010
Clock and Transfer Attribute Registers	CTAR2	32-bit	Base + 0x0014
Clock and Transfer Attribute Registers	CTAR3	32-bit	Base + 0x0018
Clock and Transfer Attribute Registers	CTAR4	32-bit	Base + 0x001C
Clock and Transfer Attribute Registers	CTAR5	32-bit	Base + 0x0020
Clock and Transfer Attribute Registers	CTAR6	32-bit	Base + 0x0024
Clock and Transfer Attribute Registers	CTAR7	32-bit	Base + 0x0028
Status Register	SR	32-bit	Base + 0x002C
DMA/Interrupt Request Register	RSER	32-bit	Base + 0x0030
PUSH TX FIFO Register	PUSHR	32-bit	Base + 0x0034
POP RX FIFO Register	POPR	32-bit	Base + 0x0038
DSPI Transmit FIFO Registers	TXFR0	32-bit	Base + 0x003C
DSPI Transmit FIFO Registers	TXFR1	32-bit	Base + 0x0040
DSPI Transmit FIFO Registers	TXFR2	32-bit	Base + 0x0044
DSPI Transmit FIFO Registers	TXFR3	32-bit	Base + 0x0048
Reserved	—	—	Base + (0x004C–0x007B)
Receive FIFO Registers	RXFR0	32-bit	Base + 0x007C
Receive FIFO Registers	RXFR1	32-bit	Base + 0x0080
Receive FIFO Registers	RXFR2	32-bit	Base + 0x0084
Receive FIFO Registers	RXFR3	32-bit	Base + 0x0088
Reserved	—	—	Base + (0x0090–0x3FFF)
FlexCAN 0 Section 18.3, Memory map and register description		**0xFFFC0000**	
Module Configuration	MCR	32-bit	Base + 0x0000
Control Register	CTRL	32-bit	Base + 0x0004
Free Running Timer	TIMER	32-bit	Base + 0x0008
Reserved	—	—	Base + (0x000C–0x000F)
Rx Global Mask Register	RXGMASK	32-bit	Base + 0x0010
Rx 14 Mask Register	RX14MASK	32-bit	Base + 0x0014
Rx 15 Mask Register	RX15MASK	32-bit	Base + 0x0018
Error Counter Register	ECR	32-bit	Base + 0x001C

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Error and Status Register	ESR	32-bit	Base + 0x0020
Interrupt Mask High Register	IMRH	32-bit	Base + 0x0024
Interrupt Mask Low Register	IMRL	32-bit	Base + 0x0028
Interrupt Flag High Register	IFRH	32-bit	Base + 0x002C
Interrupt Flag Low Register	IFRL	32-bit	Base + 0x0030
Reserved	—	—	Base + (0x0034–0x007F)
Message Buffer 0	MB0	128 bits per MB	Base + 0x0080
Message Buffer 1	MB1	128 bits per MB	Base + 0x0090
Message Buffer 2	MB2	128 bits per MB	Base + 0x00A0
Message Buffer 3	MB3	128 bits per MB	Base + 0x00B0
Message Buffer 4	MB4	128 bits per MB	Base + 0x00C0
Message Buffer 5	MB5	128 bits per MB	Base + 0x00D0
Message Buffer 6	MB6	128 bits per MB	Base + 0x00E0
Message Buffer 7	MB7	128 bits per MB	Base + 0x00F0
Message Buffer 8	MB8	128 bits per MB	Base + 0x0100
Message Buffer 9	MB9	128 bits per MB	Base + 0x0110
Message Buffer 10	MB10	128 bits per MB	Base + 0x0120

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 11	MB11	128 bits per MB	Base + 0x0130
Message Buffer 12	MB12	128 bits per MB	Base + 0x0140
Message Buffer 13	MB13	128 bits per MB	Base + 0x0150
Message Buffer 14	MB14	128 bits per MB	Base + 0x0160
Message Buffer 15	MB15	128 bits per MB	Base + 0x0170
Message Buffer 16	MB16	128 bits per MB	Base + 0x0180
Message Buffer 17	MB17	128 bits per MB	Base + 0x0190
Message Buffer 18	MB18	128 bits per MB	Base + 0x01A0
Message Buffer 19	MB19	128 bits per MB	Base + 0x01B0
Message Buffer 20	MB20	128 bits per MB	Base + 0x01C0
Message Buffer 21	MB21	128 bits per MB	Base + 0x01D0
Message Buffer 22	MB22	128 bits per MB	Base + 0x01E0
Message Buffer 23	MB23	128 bits per MB	Base + 0x01F0
Message Buffer 24	MB24	128 bits per MB	Base + 0x0200

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 25	MB25	128 bits per MB	Base + 0x0210
Message Buffer 26	MB26	128 bits per MB	Base + 0x0220
Message Buffer 27	MB27	128 bits per MB	Base + 0x0230
Message Buffer 28	MB28	128 bits per MB	Base + 0x0240
Message Buffer 29	MB29	128 bits per MB	Base + 0x0250
Message Buffer 30	MB30	128 bits per MB	Base + 0x0260
Message Buffer 31	MB31	128 bits per MB	Base + 0x0270
Message Buffer 32	MB32	128 bits per MB	Base + 0x0280
Message Buffer 33	MB33	128 bits per MB	Base + 0x0290
Message Buffer 34	MB34	128 bits per MB	Base + 0x02A0
Message Buffer 35	MB35	128 bits per MB	Base + 0x02B0
Message Buffer 36	MB36	128 bits per MB	Base + 0x02C0
Message Buffer 37	MB37	128 bits per MB	Base + 0x02D0
Message Buffer 38	MB38	128 bits per MB	Base + 0x02E0
FlexCAN 1 Section 18.3, Memory map and register description		**0xFFFC4000**	

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Module Configuration	MCR	32-bit	Base + 0x0000
Control Register	CTRL	32-bit	Base + 0x0004
Free Running Timer	TIMER	32-bit	Base + 0x0008
Reserved	—	—	Base + (0x000C–0x000F)
Rx Global Mask Register	RXGMASK	32-bit	Base + 0x0010
Rx 14 Mask Register	RX14MASK	32-bit	Base + 0x0014
Rx 15 Mask Register	RX15MASK	32-bit	Base + 0x0018
Error Counter Register	ECR	32-bit	Base + 0x001C
Error and Status Register	ESR	32-bit	Base + 0x0020
Interrupt Mask High Register	IMRH	32-bit	Base + 0x0024
Interrupt Mask Low Register	IMRL	32-bit	Base + 0x0028
Interrupt Flag High Register	IFRH	32-bit	Base + 0x002C
Interrupt Flag Low Register	IFRL	32-bit	Base + 0x0030
Reserved	—	—	Base + (0x0034–0x007F)
Message Buffer 0	MB0	128 bits per MB	Base + 0x0080
Message Buffer 1	MB1	128 bits per MB	Base + 0x0090
Message Buffer 2	MB2	128 bits per MB	Base + 0x00A0
Message Buffer 3	MB3	128 bits per MB	Base + 0x00B0
Message Buffer 4	MB4	128 bits per MB	Base + 0x00C0
Message Buffer 5	MB5	128 bits per MB	Base + 0x00D0
Message Buffer 6	MB6	128 bits per MB	Base + 0x00E0

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 7	MB7	128 bits per MB	Base + 0x00F0
Message Buffer 8	MB8	128 bits per MB	Base + 0x0100
Message Buffer 9	MB9	128 bits per MB	Base + 0x0110
Message Buffer 10	MB10	128 bits per MB	Base + 0x0120
Message Buffer 11	MB11	128 bits per MB	Base + 0x0130
Message Buffer 12	MB12	128 bits per MB	Base + 0x0140
Message Buffer 13	MB13	128 bits per MB	Base + 0x0150
Message Buffer 14	MB14	128 bits per MB	Base + 0x0160
Message Buffer 15	MB15	128 bits per MB	Base + 0x0170
Message Buffer 16	MB16	128 bits per MB	Base + 0x0180
Message Buffer 17	MB17	128 bits per MB	Base + 0x0190
Message Buffer 18	MB18	128 bits per MB	Base + 0x01A0
Message Buffer 19	MB19	128 bits per MB	Base + 0x01B0
Message Buffer 20	MB20	128 bits per MB	Base + 0x01C0

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 21	MB21	128 bits per MB	Base + 0x01D0
Message Buffer 22	MB22	128 bits per MB	Base + 0x01E0
Message Buffer 23	MB23	128 bits per MB	Base + 0x01F0
Message Buffer 24	MB24	128 bits per MB	Base + 0x0200
Message Buffer 25	MB25	128 bits per MB	Base + 0x0210
Message Buffer 26	MB26	128 bits per MB	Base + 0x0220
Message Buffer 27	MB27	128 bits per MB	Base + 0x0230
Message Buffer 28	MB28	128 bits per MB	Base + 0x0240
Message Buffer 29	MB29	128 bits per MB	Base + 0x0250
Message Buffer 30	MB30	128 bits per MB	Base + 0x0260
Message Buffer 31	MB31	128 bits per MB	Base + 0x0270
Message Buffer 32	MB32	128 bits per MB	Base + 0x0280
Message Buffer 33	MB33	128 bits per MB	Base + 0x0290
Message Buffer 34	MB34	128 bits per MB	Base + 0x02A0

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 35	MB35	128 bits per MB	Base + 0x02B0
Message Buffer 36	MB36	128 bits per MB	Base + 0x02C0
Message Buffer 37	MB37	128 bits per MB	Base + 0x02D0
Message Buffer 38	MB38	128 bits per MB	Base + 0x02E0
Message Buffer 39	MB39	128 bits per MB	Base + 0x02F0
Message Buffer 40	MB40	128 bits per MB	Base + 0x0300
Message Buffer 41	MB41	128 bits per MB	Base + 0x0310
Message Buffer 42	MB42	128 bits per MB	Base + 0x0320
Message Buffer 43	MB43	128 bits per MB	Base + 0x0330
Message Buffer 44	MB44	128 bits per MB	Base + 0x0340
Message Buffer 45	MB45	128 bits per MB	Base + 0x0350
Message Buffer 46	MB46	128 bits per MB	Base + 0x0360
Message Buffer 47	MB47	128 bits per MB	Base + 0x0370
Message Buffer 48	MB48	128 bits per MB	Base + 0x0380

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 49	MB49	128 bits per MB	Base + 0x0390
Message Buffer 50	MB50	128 bits per MB	Base + 0x03A0
Message Buffer 51	MB51	128 bits per MB	Base + 0x03B0
Message Buffer 52	MB52	128 bits per MB	Base + 0x03C0
Message Buffer 53	MB53	128 bits per MB	Base + 0x03D0
Message Buffer 54	MB54	128 bits per MB	Base + 0x03E0
Message Buffer 55	MB55	128 bits per MB	Base + 0x03F0
Message Buffer 56	MB56	128 bits per MB	Base + 0x0400
Message Buffer 57	MB57	128 bits per MB	Base + 0x0410
Message Buffer 58	MB58	128 bits per MB	Base + 0x0420
Message Buffer 59	MB59	128 bits per MB	Base + 0x0430
Message Buffer 60	MB60	128 bits per MB	Base + 0x0440
Message Buffer 61	MB61	128 bits per MB	Base + 0x0450
Message Buffer 62	MB62	128 bits per MB	Base + 0x0460

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 63	MB63	128 bits per MB	Base + 0x0470
Reserved	—	—	Base + (0x0480–0x087F)
RX Individual Mask Register 0	RXIMR0	32-bit	Base + 0x0880
RX Individual Mask Register 1	RXIMR1	32-bit	Base + 0x0884
RX Individual Mask Register 2	RXIMR2	32-bit	Base + 0x0888
RX Individual Mask Register 3	RXIMR3	32-bit	Base + 0x088C
RX Individual Mask Register 4	RXIMR4	32-bit	Base + 0x0890
RX Individual Mask Register 5	RXIMR5	32-bit	Base + 0x0894
RX Individual Mask Register 6	RXIMR6	32-bit	Base + 0x0898
RX Individual Mask Register 7	RXIMR7	32-bit	Base + 0x089C
RX Individual Mask Register 8	RXIMR8	32-bit	Base + 0x08A0
RX Individual Mask Register 9	RXIMR9	32-bit	Base + 0x08A4
RX Individual Mask Register 10	RXIMR10	32-bit	Base + 0x08A8
RX Individual Mask Register 11	RXIMR11	32-bit	Base + 0x08AC
RX Individual Mask Register 12	RXIMR12	32-bit	Base + 0x08B0
RX Individual Mask Register 13	RXIMR13	32-bit	Base + 0x08B4
RX Individual Mask Register 14	RXIMR14	32-bit	Base + 0x08B8
RX Individual Mask Register 15	RXIMR15	32-bit	Base + 0x08BC
RX Individual Mask Register 16	RXIMR16	32-bit	Base + 0x08C0
RX Individual Mask Register 17	RXIMR17	32-bit	Base + 0x08C4
RX Individual Mask Register 18	RXIMR18	32-bit	Base + 0x08C8
RX Individual Mask Register 19	RXIMR19	32-bit	Base + 0x08CC
RX Individual Mask Register 20	RXIMR20	32-bit	Base + 0x08D0
RX Individual Mask Register 21	RXIMR21	32-bit	Base + 0x08D4
RX Individual Mask Register 22	RXIMR22	32-bit	Base + 0x08D8
RX Individual Mask Register 23	RXIMR23	32-bit	Base + 0x08DC
RX Individual Mask Register 24	RXIMR24	32-bit	Base + 0x08E0
RX Individual Mask Register 25	RXIMR25	32-bit	Base + 0x08E4
RX Individual Mask Register 26	RXIMR26	32-bit	Base + 0x08E8
RX Individual Mask Register 27	RXIMR27	32-bit	Base + 0x08EC

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
RX Individual Mask Register 28	RXIMR28	32-bit	Base + 0x08F0
RX Individual Mask Register 29	RXIMR29	32-bit	Base + 0x08F4
RX Individual Mask Register 30	RXIMR30	32-bit	Base + 0x08F8
RX Individual Mask Register 31	RXIMR31	32-bit	Base + 0x08FC
RX Individual Mask Register 32	RXIMR32	32-bit	Base + 0x0900
RX Individual Mask Register 33	RXIMR33	32-bit	Base + 0x0904
RX Individual Mask Register 34	RXIMR34	32-bit	Base + 0x0908
RX Individual Mask Register 35	RXIMR35	32-bit	Base + 0x090C
RX Individual Mask Register 36	RXIMR36	32-bit	Base + 0x0910
RX Individual Mask Register 37	RXIMR37	32-bit	Base + 0x0914
RX Individual Mask Register 38	RXIMR38	32-bit	Base + 0x0918
RX Individual Mask Register 39	RXIMR39	32-bit	Base + 0x091C
RX Individual Mask Register 40	RXIMR40	32-bit	Base + 0x0920
RX Individual Mask Register 41	RXIMR41	32-bit	Base + 0x0924
RX Individual Mask Register 42	RXIMR42	32-bit	Base + 0x0928
RX Individual Mask Register 43	RXIMR43	32-bit	Base + 0x092C
RX Individual Mask Register 44	RXIMR44	32-bit	Base + 0x0930
RX Individual Mask Register 45	RXIMR45	32-bit	Base + 0x0934
RX Individual Mask Register 46	RXIMR46	32-bit	Base + 0x0938
RX Individual Mask Register 47	RXIMR47	32-bit	Base + 0x093C
RX Individual Mask Register 48	RXIMR48	32-bit	Base + 0x0940
RX Individual Mask Register 49	RXIMR49	32-bit	Base + 0x0944
RX Individual Mask Register 50	RXIMR50	32-bit	Base + 0x0948
RX Individual Mask Register 51	RXIMR51	32-bit	Base + 0x094C
RX Individual Mask Register 52	RXIMR52	32-bit	Base + 0x0950
RX Individual Mask Register 53	RXIMR53	32-bit	Base + 0x0954
RX Individual Mask Register 54	RXIMR54	32-bit	Base + 0x0958
RX Individual Mask Register 55	RXIMR55	32-bit	Base + 0x095C
RX Individual Mask Register 56	RXIMR56	32-bit	Base + 0x0960
RX Individual Mask Register 57	RXIMR57	32-bit	Base + 0x0964
RX Individual Mask Register 58	RXIMR58	32-bit	Base + 0x0968
RX Individual Mask Register 59	RXIMR59	32-bit	Base + 0x096C

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
RX Individual Mask Register 60	RXIMR60	32-bit	Base + 0x0970
RX Individual Mask Register 61	RXIMR61	32-bit	Base + 0x0974
RX Individual Mask Register 62	RXIMR62	32-bit	Base + 0x0978
RX Individual Mask Register 63	RXIMR63	32-bit	Base + 0x097C
Reserved	—	—	Base + (0x0980–0x3FFF)
Message Buffer 39	MB39	128 bits per MB	Base + 0x02F0
Message Buffer 40	MB40	128 bits per MB	Base + 0x0300
Message Buffer 41	MB41	128 bits per MB	Base + 0x0310
Message Buffer 42	MB42	128 bits per MB	Base + 0x0320
Message Buffer 43	MB43	128 bits per MB	Base + 0x0330
Message Buffer 44	MB44	128 bits per MB	Base + 0x0340
Message Buffer 45	MB45	128 bits per MB	Base + 0x0350
Message Buffer 46	MB46	128 bits per MB	Base + 0x0360
Message Buffer 47	MB47	128 bits per MB	Base + 0x0370
Message Buffer 48	MB48	128 bits per MB	Base + 0x0380
Message Buffer 49	MB49	128 bits per MB	Base + 0x0390
Message Buffer 50	MB50	128 bits per MB	Base + 0x03A0

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
Message Buffer 51	MB51	128 bits per MB	Base + 0x03B0
Message Buffer 52	MB52	128 bits per MB	Base + 0x03C0
Message Buffer 53	MB53	128 bits per MB	Base + 0x03D0
Message Buffer 54	MB54	128 bits per MB	Base + 0x03E0
Message Buffer 55	MB55	128 bits per MB	Base + 0x03F0
Message Buffer 56	MB56	128 bits per MB	Base + 0x0400
Message Buffer 57	MB57	128 bits per MB	Base + 0x0410
Message Buffer 58	MB58	128 bits per MB	Base + 0x0420
Message Buffer 59	MB59	128 bits per MB	Base + 0x0430
Message Buffer 60	MB60	128 bits per MB	Base + 0x0440
Message Buffer 61	MB61	128 bits per MB	Base + 0x0450
Message Buffer 62	MB62	128 bits per MB	Base + 0x0460
Message Buffer 63	MB63	128 bits per MB	Base + 0x0470
Reserved	—	—	Base + (0x0480–0x087F)
RX Individual Mask Register 0	RXIMR0	32-bit	Base + 0x0880
RX Individual Mask Register 1	RXIMR1	32-bit	Base + 0x0884

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
RX Individual Mask Register 2	RXIMR2	32-bit	Base + 0x0888
RX Individual Mask Register 3	RXIMR3	32-bit	Base + 0x088C
RX Individual Mask Register 4	RXIMR4	32-bit	Base + 0x0890
RX Individual Mask Register 5	RXIMR5	32-bit	Base + 0x0894
RX Individual Mask Register 6	RXIMR6	32-bit	Base + 0x0898
RX Individual Mask Register 7	RXIMR7	32-bit	Base + 0x089C
RX Individual Mask Register 8	RXIMR8	32-bit	Base + 0x08A0
RX Individual Mask Register 9	RXIMR9	32-bit	Base + 0x08A4
RX Individual Mask Register 10	RXIMR10	32-bit	Base + 0x08A8
RX Individual Mask Register 11	RXIMR11	32-bit	Base + 0x08AC
RX Individual Mask Register 12	RXIMR12	32-bit	Base + 0x08B0
RX Individual Mask Register 13	RXIMR13	32-bit	Base + 0x08B4
RX Individual Mask Register 14	RXIMR14	32-bit	Base + 0x08B8
RX Individual Mask Register 15	RXIMR15	32-bit	Base + 0x08BC
RX Individual Mask Register 16	RXIMR16	32-bit	Base + 0x08C0
RX Individual Mask Register 17	RXIMR17	32-bit	Base + 0x08C4
RX Individual Mask Register 18	RXIMR18	32-bit	Base + 0x08C8
RX Individual Mask Register 19	RXIMR19	32-bit	Base + 0x08CC
RX Individual Mask Register 20	RXIMR20	32-bit	Base + 0x08D0
RX Individual Mask Register 21	RXIMR21	32-bit	Base + 0x08D4
RX Individual Mask Register 22	RXIMR22	32-bit	Base + 0x08D8
RX Individual Mask Register 23	RXIMR23	32-bit	Base + 0x08DC
RX Individual Mask Register 24	RXIMR24	32-bit	Base + 0x08E0
RX Individual Mask Register 25	RXIMR25	32-bit	Base + 0x08E4
RX Individual Mask Register 26	RXIMR26	32-bit	Base + 0x08E8
RX Individual Mask Register 27	RXIMR27	32-bit	Base + 0x08EC
RX Individual Mask Register 28	RXIMR28	32-bit	Base + 0x08F0
RX Individual Mask Register 29	RXIMR29	32-bit	Base + 0x08F4
RX Individual Mask Register 30	RXIMR30	32-bit	Base + 0x08F8
RX Individual Mask Register 31	RXIMR31	32-bit	Base + 0x08FC
RX Individual Mask Register 32	RXIMR32	32-bit	Base + 0x0900
RX Individual Mask Register 33	RXIMR33	32-bit	Base + 0x0904

Table B-2. Detailed register map (continued)

Register description	Register Name	Used Size	Address
RX Individual Mask Register 34	RXIMR34	32-bit	Base + 0x0908
RX Individual Mask Register 35	RXIMR35	32-bit	Base + 0x090C
RX Individual Mask Register 36	RXIMR36	32-bit	Base + 0x0910
RX Individual Mask Register 37	RXIMR37	32-bit	Base + 0x0914
RX Individual Mask Register 38	RXIMR38	32-bit	Base + 0x0918
RX Individual Mask Register 39	RXIMR39	32-bit	Base + 0x091C
RX Individual Mask Register 40	RXIMR40	32-bit	Base + 0x0920
RX Individual Mask Register 41	RXIMR41	32-bit	Base + 0x0924
RX Individual Mask Register 42	RXIMR42	32-bit	Base + 0x0928
RX Individual Mask Register 43	RXIMR43	32-bit	Base + 0x092C
RX Individual Mask Register 44	RXIMR44	32-bit	Base + 0x0930
RX Individual Mask Register 45	RXIMR45	32-bit	Base + 0x0934
RX Individual Mask Register 46	RXIMR46	32-bit	Base + 0x0938
RX Individual Mask Register 47	RXIMR47	32-bit	Base + 0x093C
RX Individual Mask Register 48	RXIMR48	32-bit	Base + 0x0940
RX Individual Mask Register 49	RXIMR49	32-bit	Base + 0x0944
RX Individual Mask Register 50	RXIMR50	32-bit	Base + 0x0948
RX Individual Mask Register 51	RXIMR51	32-bit	Base + 0x094C
RX Individual Mask Register 52	RXIMR52	32-bit	Base + 0x0950
RX Individual Mask Register 53	RXIMR53	32-bit	Base + 0x0954
RX Individual Mask Register 54	RXIMR54	32-bit	Base + 0x0958
RX Individual Mask Register 55	RXIMR55	32-bit	Base + 0x095C
RX Individual Mask Register 56	RXIMR56	32-bit	Base + 0x0960
RX Individual Mask Register 57	RXIMR57	32-bit	Base + 0x0964
RX Individual Mask Register 58	RXIMR58	32-bit	Base + 0x0968
RX Individual Mask Register 59	RXIMR59	32-bit	Base + 0x096C
RX Individual Mask Register 60	RXIMR60	32-bit	Base + 0x0970
RX Individual Mask Register 61	RXIMR61	32-bit	Base + 0x0974
RX Individual Mask Register 62	RXIMR62	32-bit	Base + 0x0978
RX Individual Mask Register 63	RXIMR63	32-bit	Base + 0x097C
Reserved	—	—	Base + (0x0980–0x3FFF)

Appendix C
Revision History

This appendix describes corrections to the *MPC5606S Reference Manual*. For convenience, the corrections are grouped by revision.

C.1 Changes between revisions 6 and 7

Table C-1. Changes between revisions 6 and 7

Chapter	Description
Throughout	Updated document title to *MPC5606S Microcontroller Reference Manual*. Editorial changes and improvements.
Analog-to-Digital Converter (ADC)	In Table 5-5 (ADC digital registers), added entry for channel 42 and removed entry for channel 48. Added footnote explaining channel usage. In Section 5.3.7, External decode signals delay, added this sentence to the end of the paragraph: “When this programmed delay is taking place, the ADCSTATUS[0:2] field in the Main Status Register (MSR) will display the value 010 (wait state).” Added Note: at end of Section 5.3.1.5, Abort conversion: “NOTE: Setting either the ABORT or ABORTCHAIN bit when no conversion is taking place can cause undetermined operation of the next programmed conversion chain.”
Configurable Enhanced Modular IO Subsystem (eMIOS200)	Removed REDC block from Figure 9-4 (eMIOS200 block diagram). Removed bullet item “One Real-Time Signal Bus Client (REDC)” from Section 9.2.2, Features. Removed ETB and SRV bits from Figure 9-5 (eMIOS200 Module Configuration Register (EMIOSMCR)). Removed ETB and SRV bits from Table 9-9 (EMIOSMCR field descriptions). Removed “(former STAC Bus)” legend from Figure 9-19 (Unified Channel block diagram). Removed Section 9.5.3, Real-Time Signal Client submodule (REDC).
e200z0h Core	Changed bullet in Section 14.2, Features from “Power saving modes: doze, nap, sleep and wait” to “Dedicated power saving state: wait”.
Flash Memory	In Table 17-5 (Shadow block structure), removed “NVSRC” as a listed table and changed its description to “Start of Shadow block.” Added Warning at end of Section 17.2.5, User mode operation.
FlexCAN	Removed Doze mode references: — In Figure 18-5 (Module Configuration Register (MCR)), changed bit 13 (DOZE)to reserved. — Table 18-8 (MCR field descriptions), changed bit 13 (DOZE) to reserved. — Section 18.4.9.1, Freeze mode. removed reference to Doze mode.
Inter-Integrated Circuit Bus Controller Module (I2C)	Removed Doze mode references: — Removed IBSDOZE bit in Figure 20-6 (I2C Bus Control Register (IBCR)). — Removed IBSDOZE bit in Table 20-8 (IBCR field descriptions).

Table C-1. Changes between revisions 6 and 7 (continued)

Chapter	Description
Mode Entry Module (MC_ME)	Corrected reset value of of Figure 25-17 (Peripheral Status Register 0 (ME_PS0)) to 0x0000_0000. Corrected reset value of of Figure 25-18 (Peripheral Status Register 1 (ME_PS1)) to 0x0000_0000. Corrected reset value of of Figure 25-19 (Peripheral Status Register 2 (ME_PS2)) to 0x0000_0000. Corrected reset value of of Figure 25-20 (Peripheral Status Register 3 (ME_PS3)) to 0x0000_0000. Corrected title of Table 25-12 (Peripheral Status Registers 0…3 (ME_PS0…3) field descriptions) to reflect the correct number of ME_PS*n* registers (0:3).
Nexus Development Interface (NDI)	Removed Doze mode references: — In Table 26-7 (DS field descriptions), changed value xx1 for LPS bit to reserved.
Quad Serial Peripheral Interface (QuadSPI)	Removed Doze mode references: — In Section 30.2.2, Features, removed bullet “Support for global signal Doze mode “ — In Section 30.5.1, Modes of operation, removed this phrase from the final bullet: “ or when a request is asserted by an external controller while QSPI_MCR[DOZE] is set” — In Figure 30-2 (Module Configuration Register (QSPI_MCR)), removed DOZE bit. — In Table 30-9 (QSPI_MCR field descriptions), removed DOZE bit. — Removed Figure 577 (QuadSPI module with Power Management Block). — In Section 30.2.3.4, Module Disable mode, removed bullet “The Module Disable mode can also be initiated by hardware. A power management block can initiate Module Disable mode by asserting the ipg_doze signal while the DOZE bit in the QSPI_MCR is asserted.”
Sound Generation Logic (SGL)	Changed the term “monotonic” to “monophonic.” Updated Table 34-1 (Detailed signal descriptions), to clarify the description for the row “pwm_ch0 to pwm_ch15.” Added Table 34-4 (eMIOS channel mapping).
Stepper Stall Detect (SSD)	Removed Doze mode references: — Removed Section 36.1.3.3, Power Down modes as prefatory matter. — Removed Section 36.1.3.3.1, Doze mode. — Removed Section 36.1.3.3, Power Down modes as prefatory matter. — Promoted Section 36.1.3.3.1, Doze mode by one heading level to Section 36.1.3.3, Stop mode. In Figure 36-2 (SSD Control and Status Register (CONTROL)), changed bit 0 to reserved. — In Table 36-3 (CONTROL Register field description), changed bit 0 to reserved.
System Integration Unit Lite (SIUL)	Added Notes to WPE bit in Table 37-10 (PCRx field descriptions): [[[FSL_Specific]]] **Note:** When a pin is configured as an output, the weak internal pull up/down is disabled regardless of the WPE or WPS settings in the PCR. In Table 37-12 (Peripheral input pin selection), changed Peripheral input description for PSMI[41] from I2C_RXD_1 to LINFLEX_RXD_1.
Appendix B, Register Map	In Table B-2 (Detailed register map), added rows for PCR121–PCR132 and updated memory space of subsequent reserved memory.

C.2 Changes between revisions 5 and 6

Table C-2. Changes between revisions 5 and 6

Chapter	Description
Throughout	Updated document title to *MPC5606S Microcontroller Reference Manual*. Editorial changes and improvements.
Overview	Reformatted the “Operating mode summary” table. In the “Operating mode summary” table, changed VREG startup for Standby mode from 50 µs to 250 µs Removed bullet item “Watchdog supports optional halting during low-power modes” from Software Watchdog Timer (SWT) list.
Memory Map	Corrected abbreviation of WakeUp Unit module to WKPU. Corrected “AIPS” to “PBRIDGE.” Corrected “AXBS” to” XBAR.”
Signal Description	ERR003298: — In 208 MAPBGA pin-out, corrected K1 and M1 to XTAL and EXTAL. — In “System pin descriptions” table, corrected K1 and M1 to XTAL and EXTAL. Replaced “Debug pin descriptions” table. (ERR003377)
Safety	ERR003197, Removed STP_CR[ST]P bit: — Removed this text from first paragraph of “Modes of operation” section: “and STOP when it may be disabled by the SWT_CR[STP] bit. If the STP bit is set, the counter is stopped in STOP mode, otherwise it continues to run. On exit from STOP mode, the SWT will continue from the state it was before entering STOP mode” — Removed SWT_CR[STP] column from the “SWT operation after reset” table. — Added STOP mode to the Normal mode row of the “SWT operation after reset” table. — Added STOP mode to the Debug mode row of the “SWT operation after reset” table. — Removed the STOP mode row from “SWT operation after reset” table. — Removed the STP bit from the “SWT Control Register (SWT_CR)” figure and the “SWT_CR field descriptions” table. Added **NOTE:** to the “Functional description” section describing initial value for SWT. (ERR003076) Added the following to support the Service Ket register (SWT_SK): — In Table 4-6 (SWT memory map), added row for SWT_KEY register. — In Figure 4-12 (SWT Control Register (SWT_CR)), added KEY bit (bit 22). — In Table 4-7 (SWT_CR field descriptions), added KEY bit description. — Added SWT_KEY register (Table 4-13 (SWT_SK field descriptions) and Figure 4-18 (SWT Service Key Register (SWT_SK))).

Table C-2. Changes between revisions 5 and 6 (continued)

Chapter	Description
Analog-to-Digital Converter (ADC)	Added this note to bit 25, ABORT, in the “Main Configuration Register (MCR) field descriptions” table: “**NOTE:** If the abort pulse is valid in the last cycle of the SAMPLE phase, the current channel is correctly aborted but the data register (CDR[0..15]) of the next channel conversion shows an invalid value.” (ERRR003248) In the “DMA Enable Register (DMAE) field descriptions” table, bit 30, DCLR, changed bit value=1 to “DMA request cleared automatically before DMA occurs. In this mode the DMA will not be performed” (ERR003069). In the “Decode Signals Delay Register (DSDR) field descriptions” table, added the following to the DSSD bit description: “For the case when ADC clock = Peripheral Clock/2, the DSD bit field has to be incremented by 2 to see an additional ADC clock cycle delay on the decode signal. For example: 0000 0 ADC clock cycle delay 0010 1 ADC clock cycle delay 0100 2 ADC clock cycle delay 0110 3 ADC clock cycle delay” (ERR003169).
Boot Assist Module (BAM)	In the “Entering boot modes” section, added “**NOTE**: The watchdog (SWT) is disabled at the start of BAM execution. In the case of an unexpected issue during BAM execution, the CPU may be stalled and it will be necessary to generate an external reset to recover.” (ERR003022) In the “Download 64-bit password and password check” section, second bulleted list, SEC=1 item, changed “standby mode” to “static mode”. Added **NOTE:** to step 5 of the “FlexCAN Boot Mode Download Protocol (Autobaud Disabled)” table: “**NOTE:** Once the BAM has downloaded the code, it attempts to disable the FlexCAN module. If there is continuing traffic on the CAN bus from the boot master or other nodes, then the FlexCAN module cannot be disabled and the BAM will stall. This will prevent the downloaded code from executing and a reset will be required to recover the MCU.” (ERR003165) Added the “Flash memory password swapping” section (Project Sync 5929)
CAN Sampler	In the “Control Register (CR) field description” table, added **NOTE:** to CAN_SAMPLER bit field description (Project Sync 9259) Removed the “Register Map” section at the end of the chapter
Clock Description	In the “Features” section, first bullet list, changed “Input clock frequency from an 4 MHz to 40 MHz” to “Input clock frequency from an 4 MHz to 16 MHz” In the “CR field descriptions” table, added note in unlock_once bit field description: “**NOTE:** If the FMPLL is locked and a functional reset occurs, FMPLL_CR[UNLOCK_ONCE] is automatically set even when the FMPLL has not lost lock.” (ERR002883)

Table C-2. Changes between revisions 5 and 6 (continued)

Chapter	Description
Configurable Enhanced Modular IO Subsystem (eMIOS200)	ERR003305: — Removed GTBE bit from the “eMIOS200 Module Configuration Register (EMIOSMCR)” figure. — Removed GTBE graphical content from the “eMIOS200 Full Channel Configuration using Unified Channels only” figure. — At end of the “Input Programmable Filter (IPF)” section, changed sentence “The filter is not disabled during either freeze state or negated GTBE input.” to “The filter is not disabled during freeze state.” — At end of the “Clock Prescaler (CP)” section, changed sentence “The filter is not disabled during either freeze state or negated GTBE input.” to “The filter is not disabled during freeze state.” — At end of the “Global Clock Prescaler Submodule (GCP)” section, changed sentence “The filter is not disabled during either freeze state or negated GTBE input.” to “The filter is not disabled during freeze state.” — At end of the “Channel/Modes Initialization” section, removed final item in numbered list, “15. *[global]* Enable Global Time Base.”
Deserial Serial Peripheral Interface (DSPI)	In the “Clock Polarity Switching between DSPI Transfers” section, added **NOTE:** (ERR003105) In the “Continuous Serial Communications Clock” section, replaced the first bulleted list (ERR000575, ERR001103)
DMA Channel Mux (DMACHMUX)	In “DMACHMUX request assignments” table, set request #53 to “reserved.”
Error Correction Status Module (ECSM)	Corrected “AIPS” to “PBRIDGE.” Corrected “AXBS” to” XBAR.”
Flash Memory	ERR003114: — In “ADR field descriptions” tables (two places), added **NOTE:** to AD bit field
FlexCAN	ERR002360: — Added text to the “Rx Global Mask (RXGMASK)” section above the “Rx Global Mask Register (RXGMASK)” figure describing bit mask mis-alignment. — Added cross-reference in the “Rx 14 Mask (RX14MASK)” section to the new text added in the “Rx Global Mask (RXGMASK)” section — Added cross-reference in the “Rx 15 Mask (RX15MASK)” section to the new text added in the “Rx Global Mask (RXGMASK)” section
IEEE 1149.1 Test Access Port Controller	Added **NOTE:** at the end of the “External signal description” section (ERR003116)
Interrupt Controller (INTC)	Corrected abbreviation of WakeUp Unit module to WKPU.
Mode Entry Module (MC_ME)	In the “Interrupt Status Register (ME_IS) field descriptions” table, added **NOTE:** to I_ICONF bit field: “The I_ICONF bit will not detect that modes that select a PLL to be active also have the FXOSC enabled. Therefore, always ensure that any mode that selects PLL as the system clock also has the associated FXOSC bit set.” (ERR003202) Added **NOTE:** at end of the “SAFE Mode” section (ERR003094) Added **NOTE:** at end of the “HALT Mode” and “STOP Mode” sections (ERR003190)

Table C-2. Changes between revisions 5 and 6 (continued)

Chapter	Description
Nexus Development Interface (NDI)	In the MCKO_DIV bit field description of the “DC1 field descriptions” table, added **NOTE:** “If the Nexus clock divider (NPC_PCR[MCKO_DIV]) is set to 8 and the Nexus clock gating control (NPC_PCR[MCKO_GT]) is enabled, the Nexus clock (MCKO) will be disabled prior to the completion of transmission of the Nexus message data. Do not enable the automatic clock gating mode when the Nexus clock divider is set to 8.” (ERR002161) In the “Read/Write Access Control/Status (RWCS” section, added sentence: “Writes to this register do not begin until one JTAG clock (TCK) cycle after leaving the JTAG Update-DR state.” (ERR00817)
Quad Serial Peripheral Interface (QuadSPI)	Added **NOTE:** below “Connectivity of signals on this device” table (ERR003005)
Reset Generation Module (MC_RGM)	In “Functional Event Status Register (RGM_FES)” section, added **NOTE:** “Clearing each flag in this register requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.” (ERR002958) In “Functional Event Status Register (RGM_FES) field descriptions” table, changed 10 instances of “destructive reset assertion” to “power-on reset.” In “Destructive Event Status Register (RGM_DES)” section, added **NOTE:** “Clearing each flag in this register requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.” (ERR002958) In “Functional Bidirectional Reset Enable Register (RGM_FBRE)” section, added **NOTE:** “It is not possible for a functional event to perform a short reset sequence and also assert the external reset pin. If any functional event is defined to perform a short reset sequence in the RGM_FESS register and also assert the external reset pin, then it will perform a long reset starting with PHASE3 and not assert the external reset pin.” (ERR002977, ERR003049) At end of “STANDBY Entry Sequence” section, added **NOTE:** “If the device is in STANDBY mode and an external reset occurs, the MC_RGM may not assert the external reset for the duration of the reset sequence even when RGM_FBRE[BE_EXR] = 0. This incorrect behavior occurs only if the system releases the external reset before the end of reset sequence PHASE1.” (ERR3086)
Sound Generation Logic (SGL)	Added **NOTE:** below “Truth table for MODE_SEL[SOUND_CTRL]” table describing SOUND_CTRL behavior (ERR003184)
Wakeup Unit (WKPU)	Corrected abbreviation of WakeUp Unit module to WKPU. Corrected “AIPS” to “PBRIDGE.”

C.3 Changes between revisions 4 and 5

Table C-3. Changes between revisions 4 and 5

Chapter	Description
Throughout	Editorial changes and improvements.
Overview	In the “MPC5606S family device comparison” table: • Changed “32 kHz slow external crystal oscillator” to “32 KHz slow external crystal oscillator”. • Revised the entry for “LCD driver”. Added the feature details. Added the “How to use the MPC5606S documents” section. Added the “Using the MPC5606S” section.
Memory Map	Revised the SRAM entries.
Signal Description	In the “Voltage supply pin descriptions” table, revised the entry for VDD12. In the “Pad types” section, changed “registers in the device reference manual” to “registers in the SIUL chapter of the device reference manual”. Renamed the “Functional ports” section. In the “Pad type description” table: • Added a footnote. • Revised the entry for SMD.
Safety	Revised the Overview section. Deleted the text referring to device-specific reset values (all correct values for the device are shown in the register figures). In the SWT section, revised the “Functional description” section.
Analog-to-Digital Converter	In the “Max AD_clk frequency and related configuration settings” table, changed the INPSAMP value associated with AD_clk f_{max} = 32 + 4% (was 6h, is 7h).
Clock Description	Replaced QSPI with QuadSPI. In the “Clock architecture” section, changed “32 kHz” to “32 KHz”. Revised the “Auxiliary clocks” section. Revised the “Clock gating” section. In the CGM_SC_DC0...2 section, added “The divided clock is the reference for the associated peripheral set” to the description. Revised the ‘System clock dividers” section. Revised the “Auxiliary clock dividers” section.
Configurable Enhanced Modular IO Subsystem	In the “Register description” section, changed “and 24-bit wide data registers” to “and 16-bit wide data registers”. In the “Modulus Counter Buffered (MCB) mode” section, changed “which is $ff_ffff for a 24-bit counter” to “which is 0xFFFF for a 16-bit counter”. In the “Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode” section, changed “which is $ff_ffff for a 24-bit counter” to “which is 0xFFFF for a 16-bit counter”. In the “REDC block diagram” figure, changed “24-bit wide” to “16-bit wide”.
Crossbar Switch	In the “XBAR block diagram” figure, changed QSPI to QuadSPI. Deleted the “XBAR Switch Ports” table.
Display Control Unit	In the “Layer configuration and blending” section, revised the blending-algorithm equations.
DMA Channel Mux	Revised the Overview and Features sections.
Error Correction Status Module	Revised the memory map to show correct sizes for PCT and REV. Deleted the IMC register (not available on this device).

Table C-3. Changes between revisions 4 and 5 (continued)

Chapter	Description
Flash Memory	In the “Flash memory setting recommendations” section, changed “Master 3: DCU” to “Master 4: DCU”. Revised the LML[LLK] field description. Revised the SLL[SLK] field description. Revised the LMS[LSL] field description. Changed the reset value of the following registers: • NVPWD0 (was 0xFFFFFFFF, is 0xFEEDFACE) • NVPWD1 (was 0xFFFFFFFF, is 0xCAFEBEEF)
IEEE 1149.1 Test Access Port Controller	Added content to the “External signal description” section.
Internal Static RAM	Changed the chapter title (was “Internal static RAM”, is “Static RAM”).
LIN Controller	Replaced all RAM occurrences with “SRAM”. In the “LIN mode features” section, added the “Peripheral DMA request sources“ item.
Mode Entry Module	In the “Peripheral clock gating” section, changed “each peripheral” to “certain peripherals”.
Quad Serial Peripheral Interface	Revised the “QuadSPI modes of operation” section. Revised the “SCK — Serial Clock” section.
Real-Time Clock	Revised the RTCC[APIVAL] field description. In the RTCC[CLKSEL] field description, changed “32 kHz” to “32 KHz”. In the “API functional description” section, changed “When the counter reaches the offset count” to “When the counter reaches (offset count + 1)”. In the “RTC functional description” section, added the text “RTCC[RTCVAL]=0x000 is invalid.” to the third paragraph.
System Integration Unit Lite	Revised the “SIUL block diagram” figure. Revised the Features section. Revised the “SIUL signal properties” table. Revised the “External interrupt request input pins” section. Revised the ISR, IRER, IREER, IFEER, IFER and PCRx figures. Revised the “External interrupts” section.
Voltage Regulators and Power Supplies	In the “Power domain organization” figure, moved the wakeup pads from PD1 to PD0.
Wakeup Unit	In the IRER figure, revised the presentation of the EIRE field (was read-only, is read/write).
Appendix B	Revised the MC_RGM entries.

C.4 Changes between revisions 3 and 4

Table C-4. Changes between revisions 3 and 4

Chapter	Description
Throughout	Editorial changes and improvements. Rearranged the chapter order (alphabetical by module name).
Overview	In the “System clocks and clock generation modules” section, changed “divider ratio (÷1 to ÷15)” to “divider ratio (÷1 to ÷16)”.

Table C-4. Changes between revisions 3 and 4 (continued)

Chapter	Description
Signal Description	Changed several pin names (primarily those that contained _A, _B, _C, ...; they now contain _0, _1, _2, ...). In the “Port pin summary” table, added a footnote to the “Special function” column.
Clock Description	Revised the “MPC5606S system clock generation” figure. In the “Truth table of crystal oscillator” table, changed “ENABLE” to “OSCON” and “BYP” to “OSCBYP”. Revised the note in the “32 kHz OSC digital interface” OSC_CTL register description. Deleted the S_RC_STDBY field from the RC_CTL register (not present on this device). In the FMPLL section, changed “SSCG” to “FM”. In the FMPLL section, added notes that the maximum value of MODPERIOD is 0x1000.
Mode Entry Module	In the “STANDBY Mode” section, changed “The only parts of the device that are still powered...” to “By default the only parts of the device that are still powered...”. Changed the ME_HALT_MC[MVRON] field from read-only to read/write. Changed the ME_STOP_MC[FIRCON] field from read/write to read-only. Changed the ME_STANDBY_MC[FXOSCON] field from read-only to read/write. Revised the “MC_ME resource control overview” table.
Boot Assist Module	Deleted the FlexRay-related footnotes in the Overview and “Boot Modes” sections.
DMA Channel Mux	In the “DMACHMUX request assignments” table, changed the following: IIC_A_TX to I2C_0_TX IIC_A_RX to I2C_0_RX IIC_B_TX to I2C_1_TX IIC_B_RX to I2C_1_RX IIC_C_TX to I2C_2_TX IIC_C_RX to I2C_2_RX IIC_D_TX to I2C_3_TX IIC_D_RX to I2C_3_RX
Crossbar Switch	Renamed AIPS to PBRIDGE. Revised the “Debug mode” section.
AIPS	Changed the chapter name to “Peripheral Bridge” and the module abbreviation to “PBRIDGE”.

Table C-4. Changes between revisions 3 and 4 (continued)

Chapter	Description
Flash Memory	Added a “Sector” column to the “Flash module sectorization” table. In the “Flash module sectorization” table, changed the code-flash entries B1F0–3 to B2F0–3 and the data-flash entries B0F0–3 to B1F0–3. In the “Program flash memory (Code Flash 0 and Code Flash 1)” section: • Added a note to the BIU0–2 registers that they are available only on code flash 0. • In the “Functional description” > “Reset” section, added the text “Reset and power-off must not be used systematically to terminate a Program or Erase operation”. • Revised the description of the MCR and UT0. • In the “Functional description” > “Programming considerations” section, revised the “Modify Operation” section. (This includes a complete replacement of the “Margin Read” subsection and deletion of the “Read Reset” subsection.) • Deleted the “Register map” section. • Deleted the “Read reset” entry from the “Flash modify operations” table. In the “Data flash memory” section: • Revised the “80 KB flash module sectorization” table. • In the “Functional description” > “Reset” section, added the text “Reset and power-off must not be used systematically to terminate a Program or Erase operation”. • Revised the description of the MCR and UT0. • Revised the NVLML[LLK], NVSLL[SLK], and LMS[LSL] field descriptions. • Deleted the “Register map” section. • Deleted the “Read reset” entry from the “Flash modify operations” table • In the “Functional description” > “Programming considerations” section, revised the “Modify Operation” section. (This includes a complete replacement of the “Margin Read” subsection and deletion of the “Read Reset” subsection.) Changed “Disable Mode” to “Power-Down Mode” and “Sleep Mode” to “Low Power Mode”. In the “Low Address Space configuration” table, changed the entry for 001 (was reserved, is “2×128 KB”). In the “Mid Address Space configuration” table, changed the entry for 0 (was “2 x 128KB”, is “2×128 KB or 0 KB”). Revised the LML[MLK] and LML[LLK] field descriptions. Revised the HBL[HLK] field description. Revised the SLL[SMK] field description. Revised the SLL[SMK] and SLL[SLK] field descriptions. Revised the LMS[MSL] and LMS[LSL] field descriptions. Revised the HBS[HSL] field description. Revised the UT0[AIS] and UT0[AIE] field descriptions. Revised the UT0 values in the “ECC logic check” example. Revised the “Bit manipulation: Double Words with the same ECC value” table. Revised the NVUSR0[WATCHDOG_EN] field description.
Internal Static RAM	Revised the “SRAM memory map” table. In the “General-purpose SRAM” section, changed “plus 16 KB or 40 KB of non-standby SRAM” to “plus 16 KB or 40 KB of SRAM that may or may not be enabled in standby mode”. In the “Low power configuration” section, changed “powered during standby mode” to “powered by default during standby mode”. Added the text “Upper RAM is disabled by the MC_PCU” to the “Low power configuration” table.
Memory Protection Unit	Added information about MPU region descriptors 8–11. Added information about MPU region alternate access controls 8–11.

Table C-4. Changes between revisions 3 and 4 (continued)

Chapter	Description
Interrupt Controller	In the Introduction section, changed “supports 126 interrupt requests” to “supports 122 interrupt requests”. In the note above the interrupt vector table, changed “INTC_PSR0-INTC_PSR211” to “INTC_PSR0_3-INTC_PSR204_206”. In the “Priority management” section, changed “INTC_PSR0_3–INTC_PSR292_293” to “INTC_PSR0_3–INTC_PSR204_206”. In the “Initialization flow” section, changed “INTC_PSR0-INTC_PSR211” to “INTC_PSR0_3 -INTC_PSR204_206”. In the “ISR, RTOS, and task hierarchy” section, changed “INTC_PSR0-INTC_PSR211” to “INTC_PSR0_3 -INTC_PSR204_206”. In the “Scheduling a lower priority portion of an ISR” section, changed “INTC_PSR0_3–INTC_PSR292_293” to “INTC_PSR0_3 -INTC_PSR204_206”. In the note in the “Lowering priority within an ISR” section, changed “INTC_PSR0_3–INTC_PSR292_293” to “INTC_PSR0_3 -INTC_PSR204_206”. In the “Proper setting of interrupt request priority”, changed “INTC_PSR0_3–INTC_PSR292_293” to “INTC_PSR0_3 -INTC_PSR204_206”.
System Integration Unit Lite	Revised the “SIUL memory map” table and adjusted the register descriptions to match its contents.. Revised the MIDR2[PARTNUM] field description.
Error Correction Status Module	Replaced references to “sleep mode” with references to “low-power mode”.
System Timer Module	Added a STM counter stop note. Revised the STM_CIRn figure and field description to show the CIF bit as “w1c” (write 1 to clear). Revised the “Modes of operation” section. Revised the STM_CR[FRZ] field description. In the “Functional description” section, changed “enabled in debug mode” to “the MCU is stopped by a debugger”.
Deserial Serial Peripheral Interface	Revised the “Debug Mode” sections.
LIN Controller	In the “Error calculation for programmed baud rates” table, revised the values for a baud rate of 10417. LINIER field descriptions: Inverted field name and bit number—was 18 BEIE; is BEIE 18.
FlexCAN	Changed register names as follows: • iFLAG1 to IFRL • iFLAG2 to IFRH • iMASK1 to IMRL • iMASK2 to IMRH Revised the MCR[MAXMB] field description. Revised the ESR figure. Revised the “Freeze Mode” description. Revised the MCR[FRZ] field description.
Periodic Interrupt Timer	Removed references to RTI (not present on this device). Revised the block diagram to show 4 timers. Revised the TFLG figure.
Inter-Integrated Circuit Bus Controller Module	Revised the IBSR figure.

Table C-4. Changes between revisions 3 and 4 (continued)

Chapter	Description
Configurable Enhanced Modular IO Subsystem	Revised the “eMIOS clocking configuration” figure. In the “Channel mode selection” table, changed the entry for 10101bb (was reserved, is “Modulus Counter Buffered (Up/Down counter)”). Added a footnote to the block diagram clarifying the channels available on eMIOS200_0 and eMIOS200_1. Revised the EMIOSS[n] section. Revised the “eMIOS clocking configuration” figure. In the “Channel mode selection” table, changed the entry for 10101bb (was reserved, is “Modulus Counter Buffered (Up/Down counter)”). Added a footnote to the block diagram clarifying the channels available on eMIOS200_0 and eMIOS200_1. Revised the “Features” section. In the memory map, changed the entry for 0x020–0x11F (was used, is reserved). Revised the EMIOSMCR section. In the EMIOSUCDIS figures, changed the reset values (were 0/1, are 0). Revised the EMIOSA[n], EMIOSB[n], EMIOSCNT[n], and EMIOSALTA[n] figures. Revised the “UC BSL bits” table.
Analog-to-Digital Converter	In the “Device-specific features” section, changed “PIT channel 3” to “PIT channel 2 (for normal conversion trigger only)”.
Stepper Motor Controller	Editorial changes. Revised the “Low-power modes” section. In the MCCTL0 section, deleted the MCHME field (not available on this device). Deleted the “Operation in Halt Mode” section. Renamed “Operation in Stop Mode” to “Operation in SMC Stop Mode”. Corrected several entries in the “Impact of MCCTL1[RECIRC] and MCDCx[SIGN[4]] Bit on the PWM Output” table.
Display Control Unit	Revised the register description table and figures to show the “write-1-to-clear” (w1c) bits clearly. Revised the CtrlDescL0_1[WIDTH] field description. Revised the CtrlDescL0_4[LUOFFS] field description. Revised the DCU_MODE field descriptions. Revised the SYN_POL field descriptions. Revised the “DCU Mode selection and background color” section. In the “Functional description” section, added the “Proper sequence for enabling and disabling the DCU” section. Revised the “Layer size and positioning” section. In the “Graphics and data format” section, changed “modes” to “formats”. In the “Blend options for BB and AB configurations” table, changed “Mode” to “Format”. Revised the “Tile mode” and “CLUT/Tile RAM” sections. In the “Synchronizing to panel frame rate” section, changed “configuration at the end of the vertical blanking period” to “configuration present one HSYNC before the end of the vertical blanking period”. Revised the “Parallel data interface (camera interface)” section. Added a step to the “DCU initialization” section.
Sound Generation Logic	Revised the SGL_STATUS section to show one field, SDCIF, that is a “write-1-to-clear” (w1c) field.
LCD Driver	Revised the “Information Specific to This Device” section.

Table C-4. Changes between revisions 3 and 4 (continued)

Chapter	Description
Quad Serial Peripheral Interface	Revised the “Debug Mode (SPI Modes Only)” section. Revised the “Signal properties” table. Revised the QSPI_MCR[FRZ] field description. Revised the QSPI_SFAR section.
Safety	Revised the “Modes of operation” section. Changed “AIPS” to “PBRIDGE”. Revised the SWT_CR section. In the SWT section, added the “SWT operation timing diagram” figure and the “SWT operation after reset” figure. In the SWT section, revised the “Features” section.
System Status and Configuration Module	Revised the MEMCONFIG[IVLD] field description.
Wakeup Unit	Revised the “Wakeup vector mapping” table. Added a note about pin termination to the “External signal description” section. In the register figures, changed “[19:0]” to “[20:0]”.
Real-Time Clock	Revised the clock names to be consistent with the rest of the document. Revised the RTCS figure. Restructured the “Modes of operation” section and revised the debug information therein. Revised the RTCC[FRZEN] field description.
Voltage Regulators and Power Supplies	Revised the “Power supply configuration” figure. In the “Power domain organization” figure, changed “RAM Domain 2 (RD2)” to “Power Domain 2 (PD2)”. Revised the “Voltage regulators” section. In the “VDDR” section, changed “This pin should have an external decoupling capacitor of the order of 10 to 100 nF” to “See the device data sheet for details of recommended decoupling capacitance on this pin”. In the “High Power or Main Regulator (HPREG)” section, deleted “The minimum recommended value is 600 nF with low ESR. Limit the series inductance per pad to less than 15 nH.”.
Appendix B	Changed register names as follows: • iFLAG1 to IFRL • iFLAG2 to IFRH • iMASK1 to IMRL • iMASK2 to IMRH

C.5 Changes between revisions 2 and 3

Table C-5. Changes between revisions 2 and 3

Chapter	Description
Throughout	Editorial changes and improvements.
Overview	In the Introduction section, changed “Up to 8 modulus counter” to “Up to 5 modulus counters”. In the Introduction section, changed “Peripheral Interrupt Timer” to “Periodic Interrupt Timer”.

Table C-5. Changes between revisions 2 and 3 (continued)

Chapter	Description
Memory Map	Added the following footnote to the “Region” section: “The contents of memory addresses marked as reserved and individual bits within a memory address that are marked as reserved may return any value when read unless otherwise indicated.”
Signal Description	Renamed the analog pins (were AN..., are ANS...). Added a column for the 208-pin package to the “System pin descriptions” table. Corrected all values in the “Debug pin descriptions” table. Updated the “Port pin summary” table. Added new information on pad types (including splitting up the existing M pads into two categories, M1 and M2). In the “Signal details” table: • Added “ANS[0:15] connect to ATD channels [32:47]” to the ANS signal description. • Added “The available 8 multiplexed channels connect to ATD channels [64:71]” to the MA signal description. • Deleted “when high; otherwise low to allow a subframe display for pixels” from the DCU_DE description. • Changed the descriptions for DCU_TAG, PDI_PCLK, TXD_A, and SSD signals. • Added QuadSPI signals. • Deleted “For valid Pixel Data this is high, otherwise low” from the PDI_DE description.
Clock Description	Updated register access information in the memory map and register descriptions. Changed some clock and signal names for consistency. Revised the system clock generation figure. In the “Peripheral clock generation registers” table, deleted “CGM_AC0_DC0” from the DCU entry. Corrections throughout the Clock Generation Module (MC_CGM) section. Changed the CR[en_pll_sw] field description. Revised the “Progressive clock switching” section. Changed the title of “CMU block diagram” to “CMU component interaction” and, in that figure, changed “4–40 MHz” to “4–16 MHz”.
Mode Entry Module	Corrections throughout chapter. Updated register access information in the memory map and register descriptions. Changed some clock and signal names for consistency.
Boot Assist Module	Changed some clock and signal names for consistency.
Power Control Unit	Removed the trailing “0” from the HALT, STOP, and STANDBY modes. Updated register access information in the memory map and register descriptions.
Reset Generation Module	Updated register access information in the memory map and register descriptions. Changed the RGM_FBRE section to indicate that bits 12 and 13 are reserved.
DMA Channel Mux	Changed the bit order for the channel configuration registers.
Flash Memory	Changed the delivery value of the NVPWD0 and NVPWD1 registers (was 0xXXXXXXXX, is 0xFFFFFFFF). Revised the NVSCI0 and NVSCI1 field descriptions.
Internal Static RAM	Added clarifying information about standby modes to the “Low Power Configuration” table.
Interrupt Controller	Changed some clock and signal names for consistency. In the interrupt vector table, renamed the resources for IRQs 193–198 and 201–206.

Table C-5. Changes between revisions 2 and 3 (continued)

Chapter	Description
System Integration Unit Lite	In the SIU memory map table: • Changed the number of pad configuration registers (was 120, is 132) and adjusted the address offsets accordingly. • Replaced the footnotes. Deleted the “PAD dedicated to ADC” entry from the PCR0–PCR132 section. Revised the description text for the following registers to indicate that they are for external interrupts: • ISR • IRER • IREER • IFEER • IFER In the “Pad Configuration Register (PCR) for the different pad types in MPC5606S” figure, changed bit 12 of the “Pad with slew rate control” entry (was SRC[0], is reserved). Added a note about low-power modes to the PCRx[SRC] field description. Changed the PCRx[WPE] field description.
LINFlex	Changes throughout the chapter.
FlexCAN	Deleted the WAK_MSK and WAK_SRC fields from the MCR register. Deleted the WAK_INT field from the ESR register.
I^2C Module	Changed the bit order for the registers.
Configurable Enhanced Modular IO Subsystem	In the “Unsupported features” section, changed “Channels 8–15” to “Channels 9–15”. In the “eMIOS total channel summary” table, changed the OPWM entry for the 256 KB device (was 12, is 16). Expanded the description of the EMIOSGFLAG, EMIOSOUDIS, and EMIOSUCDIS registers to show the definitions for each of the two EMIOS modules on this device.
Analog-to-Digital Converter	Revised the “ADC implementation” figure. Deleted the “AD_clk phase min. duration” columns from the “Max AD_clk frequency and related configuration settings” table. Deleted the following registers (not present on this device): • CEOCFR[0] • CIMR[0] • DMAR[0] • CTR0 • NMCR0 • JCMR0 Corrected the presentation of the TRCx register.
Stepper Motor Controller	Corrected several entries in the “Impact of MCCTL1[RECIRC] and MCDCx[SIGN[4]] Bit on the PWM Output” table. Added MnCyM and MnCyP columns and a clarifying footnote to the “State of Output Transistors in Various Modes” table. Added a clarifying paragraph to the “Short-circuit detection” section.
Display Control Unit	Revised the INV_PXCK field description. Added RGB-to-PDI mapping information to the “Normal and Narrow Mode” section. Added an explanation of the blending algorithm to the “Blending priority of layers” section.
LCD Driver	Added the “Settings during STANDBY mode” section.
Wakeup Unit	Changed the size of the wakeup configuration registers (were bits 12:31, are bits 11:31).

Table C-5. Changes between revisions 2 and 3 (continued)

Chapter	Description
Real Time Clock	Updated the block diagram. Revised the RTCC[ROVREN] field description.
IEEE 1149.1 Test Access Port Controller	In the “e200z0 OnCE Register Addressing” table, changed the entry for “110 1111” (was for SNC, is reserved).
Nexus Development Interface	In the “Nexus Debug Interface Registers” table, deleted the entry for index 1. Revised the DID figure and field description. Deleted the text “The same values are used for the client select values written to the client select control register” from the “Nexus Messaging” section. Revised the “EVTI Generated Break Request” section.
Appendix B	Corrected the ADC registers.

C.6 Changes between revisions 1 and 2

Table C-6. Changes between revisions 1 and 2

Chapter	Description
Throughout	Editorial changes (improvements in format and style).
Preface	Added a preface to the document.
Overview	Updated the block diagram, feature list, and device comparison table. Emphasized that the block diagram is for the device with 1 MB flash memory. In the “On-chip modules include” list: • Changed the LINFlex and FlexCAN entries. • Changed the maximum number of GPIO entries (was 164, is 133).
Memory Map	Updated the column headings. Inserted the correct names for the Mode Entry Module and Reset Generation Module. Changed “(supports up to 4 [Quad]SPI Serial Flashes)” to “(supports one [Quad]SPI Serial Flash)”.
Signal Description	Updated content throughout the chapter.

Table C-6. Changes between revisions 1 and 2 (continued)

Chapter	Description
Clock Description	In the “Clock architecture” section, changed “FMPLL1 clocked by Main XOSC, only for eMIOSA, eMIOSB, and DCU clock” to “FMPLL1 clocked by Main XOSC, only for eMIOSA, eMIOSB, QuadSPI, and DCU clock”. Renamed eMIOS_A and eMIOS_B to eMIOS_0 and eMIOS_1, respectively. Replaced the “System Clock Generation” diagram. Added an “Auxiliary clocks” section. Added the “Peripheral clock generation registers” table to the “Clock gating” section. Removed references to “Cut1” and “Cut2”. Replaced the second paragraph of the “Clock Gating” section. Replaced the entire CGM section. In the “Main features” list of the XOSC external oscillator, deleted “Oscillator powerdown control and status” (it is a feature of the MC_ME, not the XOSC). In the “Functional description” of the XOSC external oscillator, deleted the information on the unsupported “Standalone control by OSC_CTL register”. Changed bits 30 and 31 of the OSC_CTL register to Reserved. In the “32 kHz OSC digital interface” section, deleted “The implementation of OSCON, S_OSC bits in OSC_CTL register depends on the parameter OSC_ON_STAT value” (the bits are implemented on this device). Revised the truth table In the “32 kHz OSC digital interface” section. Revised the “Low Power RC Oscillator (128 kHz)” section to reflect the fact that this oscillator remains on in all modes. Deleted the LPRC_CTL[LPRCON_STDBY] bit. Deleted the second paragraph of the FMPLL0/1 Overview section. In the FMPLL0/1 section, changed the reset value of CR[ODF] (was 0, is 1). In the FMPLL0/1 section, added text to clarify that 1:1 mode is for FMPLL0 only. In the CMU memory map, renamed CMU_FDISP to CMU_FDR. Revised the “PLL clock monitor” section.
Mode Entry Module	Replaced the entire chapter.
Boot Assist Module	Added “System can recover from Static mode only by Reset” to the Features section. Changed “LINFlex_A” to “LINFlex 0”. In the RCHW section, changed “Each boot sector contains at offset 0x02” to “Each boot sector contains at offset 0x00”. Changed the position of the RCHW in the “Flash memory partitioning and RCHW search” figure. Added “SWT (the BAM disables it)” to the “BAM resources” section.
Power Control Unit	Replaced the entire chapter.
Reset Generation Module	Replaced the entire chapter.
e200z0h Core	Renamed the chapter (was “e200z0 Core”, is “e200z0h Core”). Added a module boundary to the block diagram. In the block diagram, changed “MEMORY MANAGEMENT UNIT” to “NEXUS DEBUG UNIT”. Removed erroneous references to the z1 core.
Enhanced Direct Memory Access	Corrected the bit ordering in the register diagrams.
DMA Channel Mux	Added this chapter to the document.
Crossbar Switch	Revised the block diagram.

Table C-6. Changes between revisions 1 and 2 (continued)

Chapter	Description
Internal Static RAM	Updated the column headings in the SRAM memory map table.
Memory Protection Unit	In the overview, added the QuadSPI module to the list of slave ports.
Interrupt Controller	Renumbered the eMIOS_GFR entries in the interrupt vector table. In the “Interrupt Vector Table”, changed IRQs 67 and 87 to “Reserved”.
Flash Memory	Replaced the “Platform Flash Controller” section. Added the “Initialization/application information” section.
System Integration Unit Lite	Corrected content throughout to reflect the fact that this device has 133 GPIO ports and is not available in a 100-pin package. Deleted the entry for PADSEL31 from the “Peripheral input pin selection” table. Replaced the “Peripheral input pin selection” table.
Error Correction Status Module	Added a note about bitwise operators to the “ECC Status Register (ESR)” section. Revised the ESR figure. Deleted the first paragraph in the “High Priority Enables” section.
System Timer Module	Modified the structure of the chapter.
Deserial Serial Peripheral Interface	Revised the list in the “Features” section. Added information about the External Stop Mode. Updated content throughout to reflect that this device has three chip selects and five FIFO registers. Changed the definition of DSPIx_MCR. Added missing field DSPIx_CTAR*n* field descriptions and associated tables. Added information to the “Delay after Transfer” and “Continuous Serial Communications Clock” sections. Removed information relating to DSI and CSI modes. Updated the “Continuous SCK Timing Diagram (CONT=1)” figure. Added “with DBR = 0” to the first paragraph of the “Baud Rate Settings” section.
LIN Controller	Replaced the entire chapter. Added a note to the Introduction section indicating that only LINFlex_0 supports slave mode.
FlexCAN	Deleted the note in the “FlexCAN module features” section. Deleted the note in the “FlexCAN memory mapping” section. Deleted information about the Doze and Stop modes (not implemented on this device). Deleted the MCR[SLF_WAK] field (is reserved). Fixed the figure for the “Rx Individual Mask Registers”. Deleted the note in the “Rx Individual Mask Registers” section. Deleted the note in the “Matching Process” section. Added information about MB grouping to the “Interrupts” section.
CAN Sampler	Changed content throughout to reflect that Rx2–4 are present on this device. Changed the contents of the “Internal multiplexer correspondence” table. In the “Enabling/Disabling the CAN Sampler” section, added the text: When the CAN sampler is enabled, the A, D, WEN, CSN and CK to the (12 x 32) block of registers are switched to those generated by the kernel of the sampler. You can monitor CR[Active_CK] to check which is the active clock to the registers.
Periodic Interrupt Timer	Modified the structure of the chapter.

Table C-6. Changes between revisions 1 and 2 (continued)

Chapter	Description
I^2C Module	Added the “Modes of operation” section. Added the IBCR[DMAEN] field. Added the “DMA application information” section.
Configurable Enhanced Modular IO Subsystem	Changed “eMIOS200_A” to “eMIOS200_0” and “eMIOS200_B” to “eMIOS200_1”. Added device-specific information. Changed some clock and signal names for consistency. In the block diagram: • Changed CH[27] to CH[23] • Changed ipp_obe_emios_ch[27] to ipp_obe_emios_ch[23] • Changed EMIOSO[27] to EMIOSO[23] • Changed EMIOSI[27] to EMIOSI[23] • Changed emios_flag_out[27] to emios_flag_out[23] • Changed the interface signal names. Removed references to non-existent counter bus E. Revised the “UC ODISSL selection” table. Changed “Shadow FLAG register” to “Global flag register”. Deleted information about the following non-existent modes: • Input Pulse Width Measurement • Input Period Measurement • Double Action Output Compare • Pulse/Edge Accumulation • Pulse/Edge Counting • Quadrature Decode • Windowed Programmable Time Accumulation • Modulus Counter • Output Pulse Width and Frequency Modulation • Center Aligned Output Pulse Width Modulation with Dead Time Insertion (normal and buffered) • Output Pulse Width Modulation • Output Pulse Width Modulation with Trigger Deleted information about the non-existent Wheel Speed Channel. Changed bit 0 in the EMIOSMCR (was DOZEEN, is reserved). In the “Functional Description” sections, deleted all channels above 23.
Analog-to-Digital Converter	Replaced the entire chapter.
Display Control Unit	Replaced the entire chapter.
Sound Generation Logic	Changed the description of the “pwm_ch0 to pwm_ch15” signals. Revised the SGL_STATUS register figure. In the “Monophonic sound” section, changed “fixed pulse width” to “fixed duty cycle”. In the “Polyphonic sound” section, changed “a given frequency” to “the PCM sample frequency” and added the following text: The output must be filtered by a low-pass filter. The quality and the order of the low-pass filter will have a direct impact on the quality of the produced sound.
LCD Driver	Corrected the number of front and back planes.
Quad Serial Peripheral Interface	Added information about the QuadSPI module.
Safety	Updated the SWT_CR figure to reflect that this device has only 4 masters.

Table C-6. Changes between revisions 1 and 2 (continued)

Chapter	Description
System Status and Configuration Module	Added the PUB and SEC bits to the STATUS register.
Wakeup Unit	Revised the block diagram. Updated the “Wakeup Vector Mapping” table. Updated several sections to indicate that the WKPU supports 19 interrupt sources (including NMI) and three interrupt vectors. Changed the field widths in the WISR, IRER, WRER, WIREER, WIFEER, WIFER, and WIPUER registers (were 17:0, are 19:0). Revised the pin numbers in the “Non-Maskable Interrupts” section.
Real-Time Clock	Changed some clock and signal names for consistency. Revised the clock names in the RTCC[CLKSEL] field descriptions. Added the note “RTCVAL should not be set to 0” to the RTCC[RTCVAL] field description. Added the note “RTCF is not set if RTCVAL is 12'b0” to the RTCS[RTCF] field description. In the “Functional description” section, changed “A rollover wakeup and/or interrupt can be generated” to “A rollover interrupt can be generated”. In the “Functional description” section, deleted the sentence “An RTC counter rollover with this bit will cause a wakeup from low power mode”.
Voltage Regulators and Power Supplies	Changed some clock and signal names for consistency. Revised several port numbers in the “GPIO Power Bank Supplies and Functionality” section. Deleted the non-existent CGL block from the “Power Domain Organization” diagram. Revised the “Power Domain Organization” diagram.
IEEE 1149.1 Test Access Port Controller	Modified the “External signal description” section. Changed the reset value for IDCODE[PIN] (was 0b1001000000, is 0b1001100001). In the “JTAG Instructions” table, changed the codes for ACCESS_AUX_TAP_TCU and ACCESS_AUX_TAP_NPC.
Nexus Development Interface	Added the DS[LPS] field description. Deleted the RWCS[BST] field (not present on this device). Added information about the NPC_HNDSHK module.
Appendix B: Register Map	Changed some clock and signal names for consistency.

C.7 Changes between revisions 0 and 1

Table C-7. Changes between revisions 0 and 1

Chapter	Description
Throughout	Major reorganization of all content in this document.